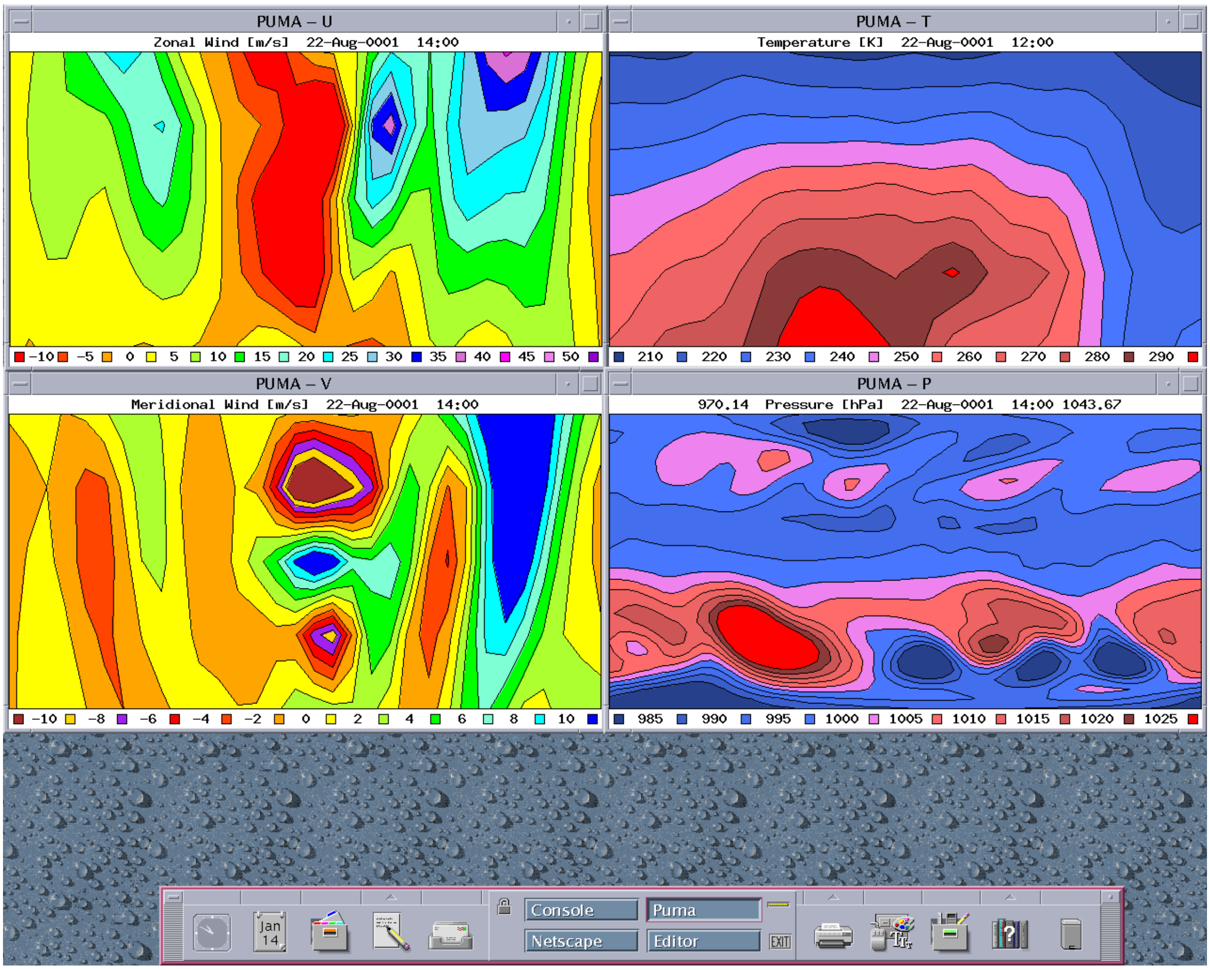
PUMA

Portable **U**niversity **M**odel of the **A**tmosphere

Klaus Fraedrich, Edilbert Kirk, Frank Lunkeit



Meteorologisches Institut
Universität Hamburg
Bundesstraße 55
20146 Hamburg

Contents

1 Introduction

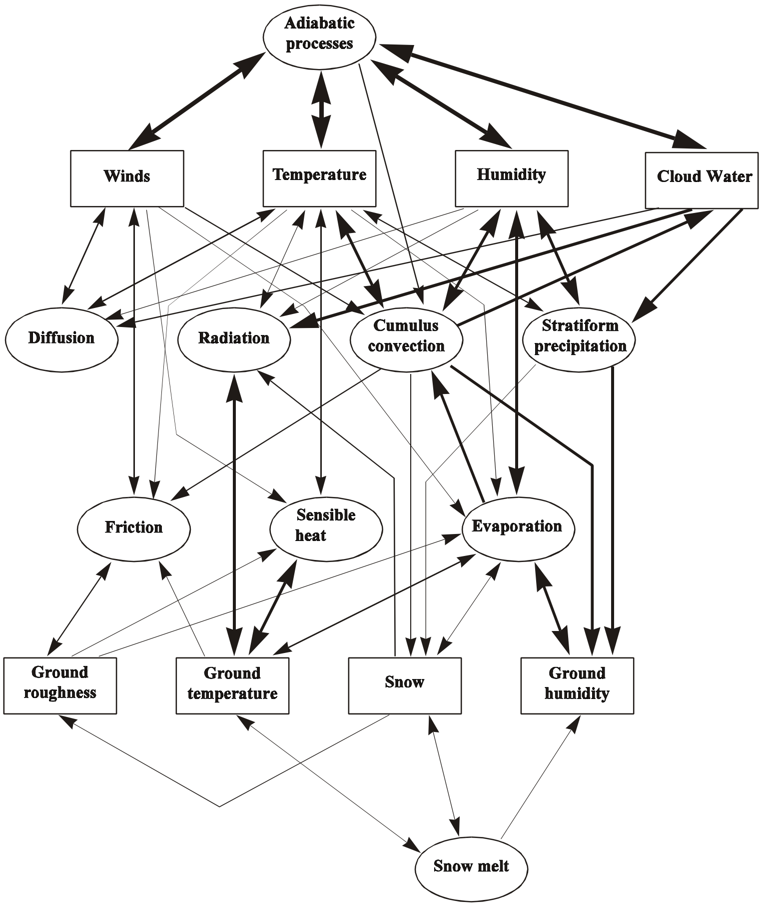
The Portable University Model of the Atmosphere (PUMA) is based on the Reading multi-level spectral model SGCM (Simple Global Circulation Model) described by Hoskins and Simmons (1975) and James and Gray (1986). Originally developed as a numerical prediction model, it was changed to perform as a circulation model. For example, James and Gray (1986) to 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. (1996) simulated an idealized storm track by embedding a dipole structure in a zonally symmetric forcing field, Lunkeit et al. (1998) studied the sensitivity of GCM scenarios.

PUMA is introduced with following two aims in mind:

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 few weeks, whereas full size GCMs require a team of specialists for maintenance, experiment design and diagnostic.

Compatibility with the ECHAM model: PUMA is designed to be as compatible as possible with the comprehensive GCM ECHAM. Therefore the same triangular truncation is employed, and analogous transformation techniques like the Legendre- and fast Fourier transformation are used. Data from numerical experiments can be processed with the ECHAM postprocessor "afterburner", and all other common tools, statistical and graphical packages, can be applied on PUMA data in the same manner as on ECHAM data. This allows PUMA to be used as a learning toolkit for potential ECHAM users.

Scientific applications: The PUMA code is the dynamical core of a GCM forced by Newtonian cooling and Rayleigh friction, such as proposed by Held and Suares (1994) to evaluate the dynamical cores of GCMs. It forms the basis for various applications: (a) The code can be utilized to build and test new numerical algorithms (like semi-Langrangian); (b) Idealised experiments can be performed to analyse nonlinear processes arising from internal atmospheric systems (life cycles, etc.) (c) Data assimilation techniques can be incorporated to interpret results from GCM simulations or obsdervations.



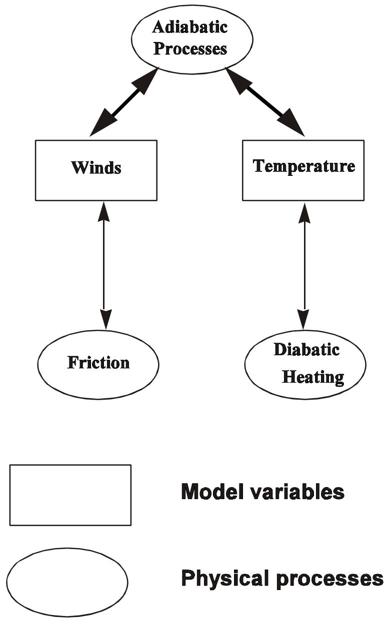


Figure 1: Processes in ECHAM

Figure 2: Processes in PUMA

Figure 1 demonstrates the complexity of interactions in a full size climate model, which leads to equally complex response patterns even on simple parameter changes. The same diagram for PUMA in figure 2 shows the direct paths, which allow the easy identification of effects.

Modes of cooperation

The use of PUMA is not restricted to a limited number of research groups. Although it is not a public domain program, anyone who is interested in atmospheric circulation model experiments, may obtain a copy of the source code including this documentation and use it freely. The only thing we ask for, is to give credit to the authors.

The PUMA version at the University of Hamburg originates from the Simmons and Hoskins SGCM version. The main differences between PUMA and its predecessor SGCM are as follows:

The code is rewritten in portable FORTRAN-90 code, which removes any problems associated with machine-specific properties like word lengths, floating point precision, output, etc. PUMA is a stand alone program, which does not use any external libraries. All necessary routines are in the source code, even the FFT (Fast Fourier Transform) and the matrix inversion. The model can now be run on any computer with a standard FORTRAN-90 compiler.

The truncation scheme is changed from jagged triangular truncation to standard triangular truncation that is compatible to other T-models like ECHAM. The model output data is written in a format compatible to the ECHAM-postprocessor “afterburner”. Thus all other diagnostic software can be used on PUMA data.

2 Model Dynamics

The model is based on the primitive equations, conveniently formulated in terms of the vertical component of absolute vorticity ζ and the horizontal divergence *D*. Terrain following σ - coordinates, $\sigma = \frac{p}{p_s}$ are used in the vertical, so that continuity is expressed as a prognostic equation for the logarithm of surface pressure, $\ln(p_s)$.

vorticity equation

$$\frac{\partial\zeta}{\partial t} = \frac{1}{1-\mu^2}\frac{\partial}{\partial\lambda}\mathscr{F}_v - \frac{\partial}{\partial\mu}\mathscr{F}_u - \frac{\xi}{\tau_F} + K(-1)^{\frac{p}{2}}\nabla^p\xi \qquad \textbf{(1)}$$

divergence equation

$$\frac{\partial D}{\partial t} = \frac{1}{1-\mu^2}\frac{\partial}{\partial\lambda}\mathscr{F}_u + \frac{\partial}{\partial\mu}\mathscr{F}_v - \nabla^2\left(\frac{U^2+V^2}{2(1-\mu^2)} + \Phi + T_R\ln p_s\right) - \frac{D}{\tau_F} + K(-1)^{\frac{p}{2}}\nabla^p D \qquad \textbf{(2)}$$

thermodynamic equation

$$\frac{\partial T'}{\partial t} = -\frac{1}{(1-\mu^2)}\frac{\partial}{\partial\lambda}(UT') - \frac{\partial}{\partial\mu}(VT') + DT' - \dot{\sigma}\frac{\partial T}{\partial\sigma} + \kappa\frac{T\omega}{p} + \frac{T_E-T}{\tau_E} + K(-1)^{\frac{p}{2}}\nabla^p T' \qquad \textbf{(3)}$$

continuity equation

$$\frac{\partial(\ln p_s)}{\partial t} = -\frac{U}{1-\mu^2}\frac{\partial(\ln p_s)}{\partial\lambda} - V\frac{\partial(\ln p_s)}{\partial\mu} - D - \frac{\partial\dot{\sigma}}{\partial\sigma} \qquad \textbf{(4)}$$

hydrostatic equation

$$\frac{\partial\Phi}{\partial(\ln\sigma)} = -T \qquad \textbf{(5)}$$

with

$$U = u \cos\phi = u\sqrt{1-\mu^2} \qquad \mathcal{F}_u = V\zeta - \dot{\sigma}\frac{\partial U}{\partial \sigma} - T'\frac{\partial(\ln p_s)}{\partial \lambda}$$

$$V = v \cos\phi = v\sqrt{1-\mu^2} \qquad \mathcal{F}_v = -U\zeta - \dot{\sigma}\frac{\partial V}{\partial \sigma} - T'(1-\mu^2)\frac{\partial(\ln p_s)}{\partial \mu}$$

ζ	absolute vorticity	λ	longitude
ξ	relative vorticity	ϕ	latitude
D	divergence	μ	sin(ϕ)
Φ	geopotential	κ	adiabatic coefficient
ω	vertical velocity	τ_E	timescale Newtonian cooling
p	pressure	τ_F	timescale Rayleigh friction
p_s	surface pressure	σ	p/p_s vertical coordinate
K	hyper diffusion	$\dot{\sigma}$	$d\sigma / dt$ vertical velocity
u	zonal wind	v	meridional wind

The temperature is split into a reference part, $T_R(\sigma)$, (usually set to be constant for the earth 250 K on all levels), and an anomaly, T'.

Parameterizations

Hyperdiffusion is represented by $K(-1)^{\frac{p}{2}}\,\nabla^p\, Q$, where p is an even integer and K a diffusion coefficient. The hyperdiffusion is included for computational reasons in the vorticity (1), divergence (2), and temperature equation (3) in order to parameterize the energy cascading into subgrid scales and its subsequent dissipation.

Rayleigh friction is included in the vorticity (1) and the divergence equation (2):

$$\frac{\partial \zeta}{\partial t} + \mathfrak{L}_{\xi} + N_{\xi} = -\frac{\xi}{\tau_F} \tag{6}$$

$$\frac{\partial D}{\partial t} + \mathfrak{L}_{D} + N_{D} = -\frac{D}{\tau_F} \tag{7}$$

The timescale, τ_F , can be individually defined for every horizontal layer. Its vertical distribution has a maximum value at the surface decreasing upwards to zero. This is a simple linear approximation for surface drag and turbulent exchange of momentum in the boundary layer.

Newtonian cooling parameterizes the diabatic processes and is included in the thermodynamic equation (3) as a linear term:

$$\frac{\partial T'}{\partial t} + \mathfrak{L}_{T} + N_{T} = \frac{(T_E - T)}{\tau_E} \tag{8}$$

It represents the process of radiative-convective heating in the free atmosphere. The model is set into motion as the model temperature T relaxes towards the restoration temperature, T_E, which, usually, has a large equator to pole gradient; τ_E is the e-folding timescale of the Newtonian cooling.

Dimensions

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	101325 hPa	PSURF = constant global mean pressure
Orography	$(a^2\ \Omega^2) / g$	g = gravity constant

Solution

The equations are solved using the spectral transform method (Orszag 1970, Eliasen et al. 1970). A variable Q is represented by a truncated series of spectral harmonics:

$$Q\ (\lambda,\mu)\ =\ \sum_{n,m}\ Q_n^m\ P_n^m\ (\mu)\ e^{im\lambda} \tag{10}$$

where n is the total wavenumber and m the zonal wavenumber with m $\leq$ n; the $P_n^m(\mu)$ are the associated Legendre functions. The spectral transformation method relies on a transformation between Q and the spectral coefficients Q_n^m, and its inverse. Each timestep involves a transformation of the variables from spectral to gridpoint representation and back again. Linear terms are evaluated in spectral space and nonlinear products (such as $U\zeta$) are evaluated in gridpoint space. The gridpoint phase of the step provides the opportunity to introduce local parameterizations of radiation, convection adjustments, friction and so on. All of these processes are represented by linear terms (Newtonian cooling and Rayleigh friction) in PUMA and can be solved very fast in the spectral domain.

The <u>F</u>ast <u>F</u>ourier <u>T</u>ransformation (FFT) provides an extremely fast transform in the zonal direction. The transforms Q^m (µ), which are the Fourier coefficients at each latitude, are calculated in this way. From these, the spectral coefficients are obtained by integration with respect to µ, using the orthogonally of the Q_n^m (µ):

$$Q_n^m\ =\ \int_{-1}^{+1}\ Q^m\ (\mu)\ P_n^m\ (\mu)\ d\mu \tag{10}$$

The transformation in µ is carried out by Gaussian quadrature (Press et al. 1989).

Vertical Discretization

Level	σ	Variables
0.5	0.0	$p=0 \quad \dot\sigma=0$
1	0.1	ζ, D, T'
1.5	0.2	$\dot\sigma$
2	0.3	ζ, D, T'
2.5	0.4	$\dot\sigma$
3	0.5	ζ, D, T'
3.5	0.6	$\dot\sigma$
4	0.7	ζ, D, T'
4.5	0.8	$\dot\sigma$
5	0.9	ζ, D, T'
5.5	1.0	$p=p_s \quad \dot\sigma=0$

Figure 3: Vertical geometry of PUMA with associated variables (5 level version)

The model is represented by finite differences in the vertical. The number of the levels in the vertical is variable with σ = p/p_s. The prognostic variables ζ, *D*, and *T'* are calculated at full-levels. At the half levels σ = 0 (upper boundary) and σ = 1 (lower boundary) the vertical velocity is set to zero. The vertical advection at level r is approximated as follows:

$$\left(\overline{\dot\sigma\delta_\sigma Q^\sigma}\right)_r = \frac{1}{2}\left(\dot\sigma_{r+\frac{1}{2}}\frac{Q_{r+1}-Q_r}{\Delta\sigma} + \dot\sigma_{r-\frac{1}{2}}\frac{Q_r-Q_{r-1}}{\Delta\sigma}\right) \qquad (11)$$

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).

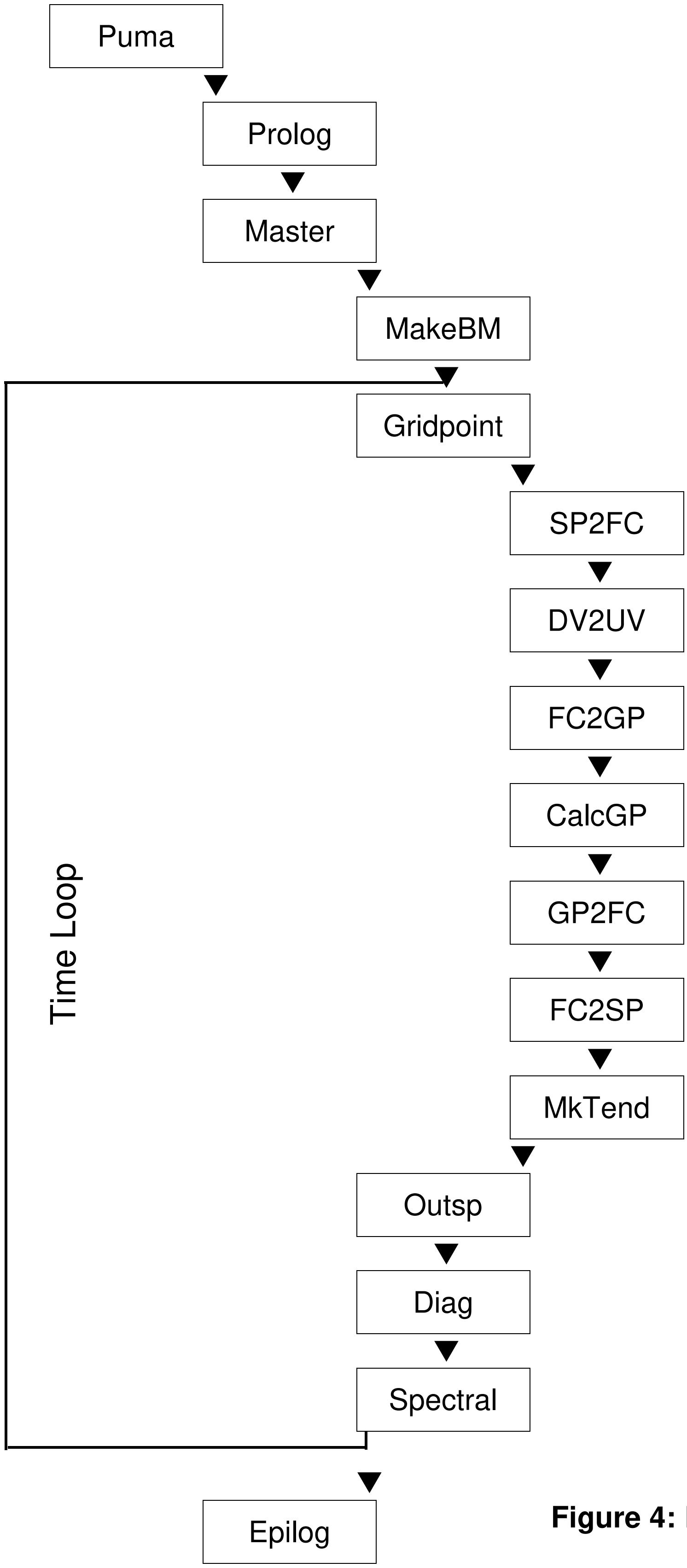


Figure 4: Flow diagram of main routines

3 Model Structure

The diagram (Fig. 4) shows the route through the main program PUMA, with the names of the most important subroutines.

PumaMod is a module, that defines all global parameters, variables and arrays.
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 of a previous run, if selected.
- **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 On initial runs **master** does some initial timesteps, 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 grid point domain.
  - **sp2fc** spectral to Fourier coefficients (inverse Legendre transf.)
  - **dv2uv** divergence, 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 make tendencies (implies fourier to spectral)

spectral does all calculations in the spectral domain.

outsp writes spectral fields in physical dimensions on an output file.

diag writes selected fields and parameters to standard output.

Epilog writes the restart file.

Initialization

The model starts either from a restart file or with an 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 horizontal constant 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, the model runs zonally constant without eddies. **kick**=1 generates random white noise, **kick**=2 generates random white noise that is symmetric to the equator. Runs started with **kick**=1 or 2 are irreproducible due to the randomization. For reproducible runs with eddies use **kick**=3 which initializes only 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 $T_{ER}(\sigma)$ is defined. A hyperbolic function of height is used to provide T_{ER}, as illustrated in Fig. 4. With $z \rightarrow -\infty$ the profile tends to a uniform laps rate, (**alr**), passing through the temperature (**tgr**) at $z = 0$. With $z \rightarrow +\infty$ the profile becomes isothermal. The transition takes place at a 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.

The radiative equilibrium temperature is then set to:

$$T_E\,(\phi,\sigma) = T_{ER}\,(\sigma) + f\,(\sigma)\left[\Delta T_{NS}\,\frac{\mu}{2} - \Delta T_{EP}\left[\mu^2 - \frac{1}{3}\right]\right] \tag{13}$$

The function $f\,(\sigma)$ is set to become small near the upper boundary. In the code supplied, it has the form:

$$\begin{aligned} f(\sigma) &= \sin\left(\frac{\pi\ (\sigma-\sigma_T)}{2\ (1-\sigma_T)}\right) && \text{for } \sigma \geq \sigma_T \\ f(\sigma) &= 0 && \text{for } \sigma < \sigma_T \end{aligned} \tag{14}$$

(where σ_T is the value of σ at height (ztrop), calculated within the subroutine). ΔT_{NS} defines the Northpole-Southpole temperature difference, and ΔT_{EP} the equator-pole temperature difference. Note that ΔT_{EP} should be positive for the usual case where the poles are colder than the equator.

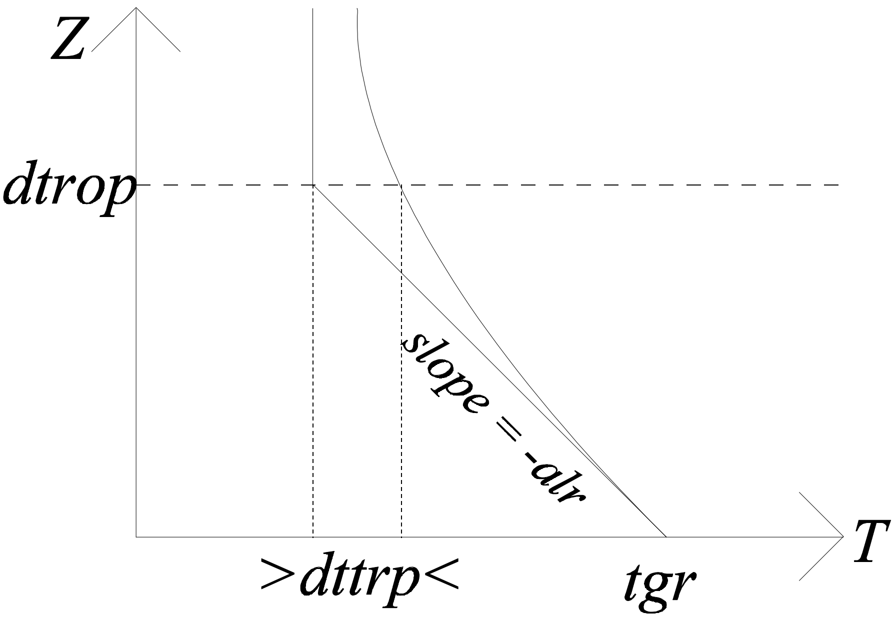


Figure 4: Specifying the tropopause and the equilibrium temperature T_{ER} from input parameters.

Computations in spectral domain

The subroutine **spectral** performs one timestep. Details of the time stepping scheme are given in Hoskins and Simmons (1978). The adiabatic tendencies (advection, etc.) are used. The normal timestep is centered in time, and includes a Robert time filter to control time splitting. For the first **nkits** timesteps, short initial timesteps, ie, an initial forward timestep followed 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:

The Newtonian cooling is the relaxation of the temperature field ST towards the equilibrium-radiative temperature (T_{ER}) fields **sr**, which is calculated in subroutine **setzt** or **setztex**. The relaxation is driven by the timescale parameter **damp**, set by the namelist parameter **restim**. The Newtonian cooling is added to the spectral temperature tendency **stt**.

The Rayleigh friction is the damping of the spectral divergence **sd** and vorticity **sz** arrays by the timescale **tfrc**, set as namelist parameter. Rayleigh friction is added to the spectral tendencies **sdt** and **szt**.

Hyperdiffusion: The temperature, divergence and vorticity (**st, sd, sz**) are multiplied by the hyperdiffusion term, the order is defined by the namelist parameter **ndel**. Hyperdiffusion is added to the according spectral tendencies (**stt, sdt, szt**).

Files

Subroutine **readnl** reads the namelist from **stdin**.

Subroutine **diag** writes some diagnostics to **stdout**.

Subroutine **restart** reads restart data from **fort.10**.

Subroutine **epilog** writes restart data to **fort.12**, containing nstep, svo, sd, st, spn, svomi, sdmi, stmin, spmin, sgs and stres.

Subroutine **outsp** writes at selected time intervals data records to **fort.40**, containing the variables divergence, vorticity, temperature and logarithm of surface pressure.

4 User Manual

Compiling

PUMA may be compiled with any standard FORTRAN-90 compiler.

Sample compile command:

```
f90 -o puma puma.f
```

Use the "-g" option for debugging and the "-O" option for performance.

Sample run command:

```
puma <namelist >output
```

The namelist is read from file "namelist" the output is written to file "output".

Sample CRAY job:

```
#QSUB -eo -lM 2000kw -lT 600
cd $TMPDIR
set -ex
PUMADIR=$HOME/puma           # SOURCE DIRECTORY
EXP=Control                  # EXPERIMENT IDENTIFIER
PUMADAT=$MFHOME/puma/${EXP} # EXPERIMENT DIRECTORIES /mf/...
cp $PUMADIR/puma .
cat > NAMELIST << EOF
 &INP
 NKITS=3, KICK=1, DTEP=80., DTNS=40.,
 RESTIM=30.,30.,30.,30.,30.,
 TFRC=4*0.,1., TDISS=0.25, NDEL=8, NRUN=8640, NDIAG=720,
 NAFTER=24, TSPD=48., LRSTRT=${1}, NCOEFF=10,
 LSPO=4*.F.,.T.
 &END
EOF
```

```
#
cp $EXPDIR/model.x model.x
assign -s unblocked u:40
#
model.x < NAMELIST
#
assign -R
#
# history and restart saved for further diagnostics
mv fort.40 ${EXP}PUMA\_0101
exit
```

The model output can be analysed by using the AFTERBURNER, a program which process model data represented in T21, T42, T63 and T106 resolution in gridpoint and spectral space.

The AFTERBURNER is documented in:
/pf/k/k204004/burn/mod.doc

and the program is found in:
/pf/k/k204004/burn/after

5 Literature

Frisius T., F. Lunkeit, K. Fraedrich and I. N. James, 1998: Storm track simulation in a simplified global circulation model (SGCM). *Q J Roy Meteor Soc*, in press.

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

James I. N. and L. J. Gray, 1986: Concerning the effect of surface drag on the circulation of a planetary atmosphere. *Q J Roy Meteor Soc* **112**: 1231-1250.

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

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

Lunkeit F., K. Fraedrich and S. E. Bauer, 1998: Storm tracks in a warmer climate: Sensitivity studies with a simplified global circulation model. *Clim Dyn* submitted.

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

A Global Constants and Variables

All global constants and variables are declared in the module PUMAMOD

```
      module pumamod

!     ********************
!     * Global Constants *
!     ********************

      parameter(NTRU = 21)                ! Truncation
      parameter(NLAT = 32)                ! Latitudes
      parameter(NLEV =  5)                ! Number of levels
      parameter(NLON = NLAT + NLAT)       ! Number of longitudes
      parameter(NHOR = NLON * NLAT)       ! Horizontal part
      parameter(NLEM = NLEV - 1)          ! Levels - 1
      parameter(NLEP = NLEV + 1)          ! Levels + 1
      parameter(NLSQ = NLEV * NLEV)       ! Levels squared
      parameter(NTP1 = NTRU + 1)          ! Truncation + 1
      parameter(NRSP =(NTRU+1)*(NTRU+2))  ! No of real global    modes
      parameter(NCSP = NRSP / 2)          ! No of complex global modes
      parameter(NVCT = 2 * (NLEV+1))      ! Dim of Vert. Coord. Tab
      parameter(NROOT = 0)                ! Master node
      parameter(AKAP   = 0.286)           ! Kappa
      parameter(ALR    = 0.0065)          ! Lapse rate
      parameter(EZ     = 1.63299310207)   ! ez = 1 / sqrt(3/8)
      parameter(GA     = 9.81)            ! Gravity
      parameter(GASCON = 287.0)           ! Gas constant
      parameter(PI     = 3.14159265359)   ! Pi
      parameter(TWOPI  = PI + PI)         ! 2 Pi
      parameter(PLARAD = 6371000.0)       ! Planet radius
      parameter(PNU    = 0.02)            ! Time filter
      parameter(PNU21  = 1.0 - 2.0*PNU)   ! Time filter 2
      parameter(PSURF  = 101325.0)        ! Surface pressure [Pa]
      parameter(WW     = 0.00007292)      ! Rotation speed
      parameter(CV     = PLARAD * WW)     ! velocity scale
      parameter(CT     = CV*CV/GASCON)    ! temperature scale
```

```
!     *************************
!     * Global Integer Scalars *
!     *************************

      integer :: kick     =  1 ! add noise for kick > 0
      integer :: nafter   = 12 ! write data interval
      integer :: ncoeff   =  0 ! number of modes to print
      integer :: ndel     =  8 ! order of n - diffusion
      integer :: ndiag    = 12 ! write diagnostics interval
      integer :: nexp     =  0 ! experiment number
      integer :: nexper   =  0 ! 1: dipole experiment
      integer :: nkits    =  3 ! number of initial timesteps
      integer :: nrestart =  0 ! 1: read restart file 0: initial run
      integer :: nrun     =  0 ! if (nstop == 0) nstop = nstep + nrun
      integer :: nstep    =  0 ! current timestep
      integer :: nstop    =  0 ! finishing timestep
      integer :: ntspd    = 24 ! number of timesteps per day

!     **********************
!     * Global Real Scalars *
!     **********************

      real :: amco  =    50.0
      real :: amwo  =    50.0
      real :: ascl  =     0.0
      real :: ascn  =    10.0
      real :: ascr  =     8.0
      real :: ascs  =     0.0
      real :: aswl  =     8.0
      real :: aswn  =     0.0
      real :: aswr  =     0.0
      real :: asws  =    10.0
      real :: delt            ! 2 pi / ntspd timestep interval
      real :: delt2           ! 2 * delt
      real :: dtep  =    60.0
      real :: dtns  =     0.0
      real :: dtrop = 12000.0
      real :: dttrp =     2.0
      real :: hco   =    10.0
      real :: hwo   =    10.0
```

```
      real :: pclam =   145.0
      real :: pcphi =    50.0
      real :: pwlam =   180.0
      real :: pwphi =    40.0
      real :: tdiss =     0.25 ! diffusion time scale [days]
      real :: tgr   =   288.0  ! Temperature ground in mean profile
      real :: wco   =    10.0
      real :: wwo   =    10.0

!     ***********************
!     * Legendre Polynomials *
!     ***********************

!     polix : indirect transformation (spectral to fourier )
!     poldx :   direct transformation (fourier  to spectral)
!       x=p : polynomials
!       x=d : deviation polynomials

      real :: plip(NLAT,NCSP),plid(NLAT,NCSP)
      real :: pliu(NCSP,NLAT),pliv(NCSP,NLAT)
      real :: pldp(NLAT,NCSP),pldd(NLAT,NCSP)
      real :: pldc(NLAT,NCSP),pldq(NLAT,NCSP)
      real trig(NLON)

!     *************************
!     * Global Spectral Arrays *
!     *************************

      real ::  sd(NRSP,NLEV) = 0.0 ! Spectral Divergence
      real ::  st(NRSP,NLEV) = 0.0 ! Spectral Temperature
      real ::  sz(NRSP,NLEV) = 0.0 ! Spectral Vorticity
      real ::  sp(NRSP)      = 0.0 ! Spectral Pressure (ln Ps)
      real ::  so(NRSP)      = 0.0 ! Spectral Orography
      real ::  sr(NRSP,NLEV) = 0.0 ! Spectral Restoration Temperature
      real :: sdt(NRSP,NLEV) = 0.0 ! Spectral Divergence  Tendency
      real :: stt(NRSP,NLEV) = 0.0 ! Spectral Temperature Tendency
      real :: szt(NRSP,NLEV) = 0.0 ! Spectral Vorticity   Tendency
      real :: spt(NRSP)      = 0.0 ! Spectral Pressure    Tendency
      real :: sdm(NRSP,NLEV) = 0.0 ! Spectral Divergence  Minus
      real :: stm(NRSP,NLEV) = 0.0 ! Spectral Temperature Minus
      real :: szm(NRSP,NLEV) = 0.0 ! Spectral Vorticity   Minus
```

```
      real :: spm(NRSP)      = 0.0 ! Spectral Pressure    Minus
      real :: sak(NRSP)      = 0.0 !
      real :: span(NRSP)     = 0.0 ! Pressure for diagnostics
      real :: spnorm(NRSP)   = 0.0 ! Factors for output normalization
      integer :: nindex(NRSP) = NTRU ! Holds wavenumber

!     **************************
!     * Global Gridpoint Arrays *
!     **************************

      real gd(NHOR,NLEV),gt(NHOR,NLEV),gz(NHOR,NLEV)
      real gu(NHOR,NLEV),gv(NHOR,NLEV),gp(NHOR)
      real rcsq(NHOR)

!     *********************
!     * Diagnostic Arrays *
!     *********************

      integer :: ndl(NLEV) = 0
      real csu(NLAT,NLEV),csv(NLAT,NLEV),cst(NLAT,NLEV)

!     *******************
!     * Latitude Arrays *
!     *******************

      character *3 chlat(NLAT)
      real csq(NLAT),si(NLAT),gw(NLAT)

!     ****************
!     * Level Arrays *
!     ****************

      real :: restim(NLEV) =  15.0
      real :: t0(NLEV)     = 250.0
      real :: tfrc(NLEV)   = (/ (0.0,i=1,NLEM), 1.0 /)
      real :: vct(NVCT)    = 0.0
      real damp(NLEV)
      real dsigma(NLEV)
      real rdsig(NLEV)
      real sigma(NLEV)
```

```
real sigmah(NLEV)
real t01s2(NLEV)
real tkp(NLEV)
real c(NLEV,NLEV)
real g(NLEV,NLEV)
real tau(NLEV,NLEV)
real bm1(NLEV,NLEV,NTRU)

end module pumamod
```

B Namelist

The characteristics of the run can be set by means of a namelist INP which is read in by subroutine **readnl**. Usable defaults have been set for these variables.

Variable	Defaults	Meaning
`kick`	1	0: no noise intialization (p_s = const.) 1: random white noise 2: equator symmetric random white noise. 3: mode (1,2) no random initialization
`nafter`	12	output intervall for data [timesteps]
`ncoeff`	0	number of coefficients to print in **wrspam**
`ndel`	8	order of hyperdiffusion (must be even)
`ndiag`	12	output intervall for diagnostics [timesteps]
`nexp`	0	experiment identifier
`nexper`	0	1: dipole added to T_E
`nkits`	3	number of short initial timesteps
`nrestart`	0	1: perform a restart run
`nrun`	0	number of timesteps to run (excl. initial ones)
`nstep`	0	current timestep
`nstop`	0	stop at timestep "nstop
`ntspd`	24	number of timesteps per day

Variable	Defaults	Meaning
dtep	60.0	Equator - pole temperature difference at surface for T_E.
dtns	0.0	Pole to pole temperature difference at surface for T_E, can simulate winter / summmer hemispheres.
ztrop	2.0	A temperature increment which controls the sharpness of the tropopause in T_E.
dttrp	12000.0	Height of tropopause in T_E.
tdiss	0.25	Dissipation time in sidereal days for wavenumber.
tgr	288.0	Global mean temperature of ground used to set T_E.
tspd	24.0	Number of time-steps per day.
ndl(NLEV)	NLEV * 0	1: activate spectral printouts for this level
restim(NLEV)	NLEV * 15.0	Radiative equilibrium timescale τ_E for each level.
t0(NLEV)	NLEV * 250.0	Reference temperature profile.
tfrc(NLEV)	0,0,0,.. ,1	Rayleigh friction timescale τ_F in days for each levels.