User Manual for the LMD Martian Atmospheric General Circulation Model

François FORGET, Ehouarn MILLOUR
Contributors to earlier versions: Karine Dassas, Christophe HOURDIN, Frédéric HOURDIN, and Yann WANHERDRICK
initial version translated by Gwen Davis (LMD)

December 3, 2009

Draft

Contents

Chapter 1

Introduction

This document is a user manual for the General Circulation Model of the Martian atmosphere developed by the Laboratoire de Météorologie Dynamique of the CNRS in Paris in collaboration with the Atmospheric and Oceanic Planetary Physics sub-department in Oxford. It corresponds to the version of the model available since November 2002, that includes the new dynamic code lmdz3.3 and the input and output data in NetCDF format. The physical part has been available since June 2001, including the NLTE radiative transfer code valid at up to 120 km, tracer transport, the water cycle with water vapour and ice, the "double mode" dust transport model, and with optional photochemistry and extension in the thermosphere up to 250km.

A more general, scientific description of the model without tracers can be found in *Forget et al.* [1999].

Chapter 2 of this document, to be read before any of the others, describes the main features of the model. The model is divided into two relatively independent parts: (1) The hydrodynamic code, that is shared by all atmospheres (Earth, Mars, etc.) that integrates the fluid mechanics equations in time and on the globe, and (2) a set of Martian physical parameterizations, including, for example, the radiative transfer calculation in the atmosphere and the turbulence mix in the upper layer.

It is followed by a list of references for anyone requiring a detailed description of the physics and the numerical formulation of the parameterizations of the Martian physical part (Chapter 3).

For your **first contact with the model**, chapter 4 guides the user through a practice simulation (choosing the initial states and parameters and visualizing the output files).

The document then describes the programming code for the model, including a user computer manual for compiling and running the model (Chapter 5).

Chapter 6 describes the input/output data of the model. The input files are the files needed to initialize the model (state of the atmosphere at instant $t0$ as well as a dataset of boundary conditions) and the output files are "historical files", archives of the atmospheric flow history as simulated by the model, the "diagfi files", the "stats files", the daily averages etc. The means to edit or visualize these files (editor "ncdump" and the graphics software "grads") are also explained.

Chapter 8 explains how to run a simulation including the water cycle. Chapter 9 illustrates how to run the model with the photochemical module.

Finally, chapter 10 will help you to use a 1-dimensional version of the model, which may be a simplier tool for some analysis work.

Chapter 2

Main features of the model

2.1 Basic principles

The General Circulation Model (GCM) calculates the temporal evolution of the different **variables** (listed below) that control or describe the Martian meteorology and climate at different points of a **3D "grid"** (see below) that covers the entire atmosphere.

From an initial state, the model calculates the evolution of these variables, timestep by timestep:

- At instant t, we know variable X_t (temperature for example) at one point in the atmosphere.
- We calculate the evolution (the **tendencies**) $(\frac{\partial X}{\partial t})_1$, $(\frac{\partial X}{\partial t})_2$, etc. arising from each physical phenomenon, calculated by a **parameterization** of each of these phenomenon (for example, heating due to absorption of solar radiation).
- At the next time step $t + \delta t$, we can calculate $X_{t+\delta t}$ from X_t and $(\frac{\partial X}{\partial t})$. This is the **"integration"** of the variables in time. (For example, $X_{t+\delta t} = X_t + \delta t(\frac{\partial X}{\partial t})_1 + \delta t(\frac{\partial X}{\partial t})_2 + ...$)

The main task of the model is to calculate these tendencies $(\frac{\partial X}{\partial t})$ arising from the different parameterized phenomenon.

2.2 Dynamical-Physical separation

In practice, the 3D model operates in two parts:
- one **dynamical part** containing the numerical solution of the general equations for atmospheric circulation. This part (including the programming) is common to the Earth and Martian model, and in general for all atmospheres of the terrestrial type.
- a second **physical part** that is specific to the planet in question and which calculates the forced circulation and the climate details at each point.

The calculations for the dynamical part are made on a 3D grid with horizontal exchanges between the grid boxes, whereas the physical part can be seen as a juxtaposition of atmosphere "columns" that do not interact with each other (see diagram 2.1).

The dynamical and physical parts deal with variables of different natures, and operate on grids that are differently constructed. The temporal integration of the variables is based on different numerical schemes (simple, such as the one above for the physical part, and more complicated, the "Matsuno-Leapfrog" scheme for the dynamical part). The timesteps are also different. The physical timestep is `iphysiq` times longer than the dynamical

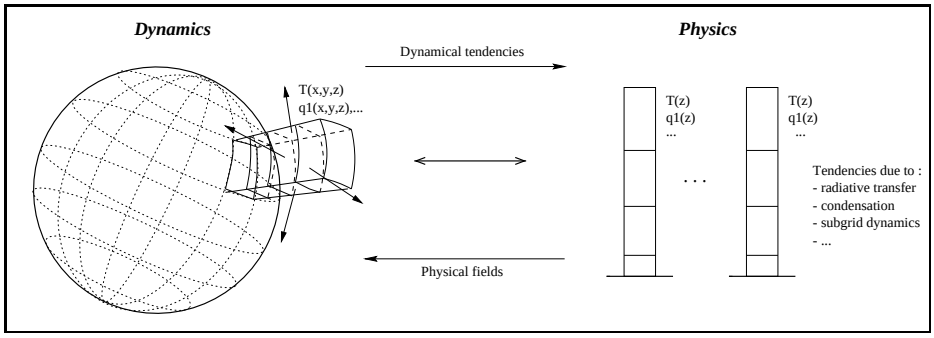

Figure 2.1: Physical/dynamical interface

timestep, as the solution of the dynamic equations requires a shorter timestep than the forced calculation for the physical part.

In practice, the main program that handles the whole model (`gcm.F`) is located in the dynamical part. When the temporal evolution is being calculated, at each timestep the program calls the following:

1. Call to the subroutine that handles the total tendency calculation ($\frac{\partial X}{\partial t}$) arising from the dynamical part (`caldyn.F`)
2. Integration of these dynamical tendencies to calculate the evolution of the variables at the following timesteps (subroutine `integrd.F`)
3. Every `iphysiq` dynamical timestep, a call to the interface subroutine (`calfis.F`) with the physical model (`physiq.F`), that calculates the evolution of some of the purely physical variables (e.g: surface temperature `tsurf`) and returns the tendencies ($\frac{\partial X}{\partial t}$) arising from the physical part.
4. Integration of the physical variables (subroutine `addfi.F`)
5. Similarly, calculation and integration of tendencies due to the horizontal dissipation and the "sponge layer" is done every `idissip` dynamical time step.

Remark: The physical part can be run separately for a 1-D calculation for a single column using program `testphys1d.F`.

2.3 Grid boxes:

Examples of typical grid values are 64x48x25, 64x48x32 or 32x24x25 in longitudexlatitudexaltitude. For Mars (radius$\sim$3400 km), a 64x48 horizontal grid corresponds to grid boxes of the order of 330x220 kilometers near the equator.

2.3.1 Horizontal grids

Dynamics and physics use different grids. Figure 2.2 shows the correspondance and indexing of the physical and dynamical grids as well as the different locations of variables on these grids. To identify the coordinates of a variable (at one grid point up, down, right or left) we use coordinates `rlonu`, `rlatu`, `rlonv`, `rlatv` (longitudes and latitudes, in **radians**).

On the dynamical grid, values at `i=1` are the same as at `i=IM+1` as the latter node is a redundant point (due to the periodicity in longitude, these two nodes are actualy located

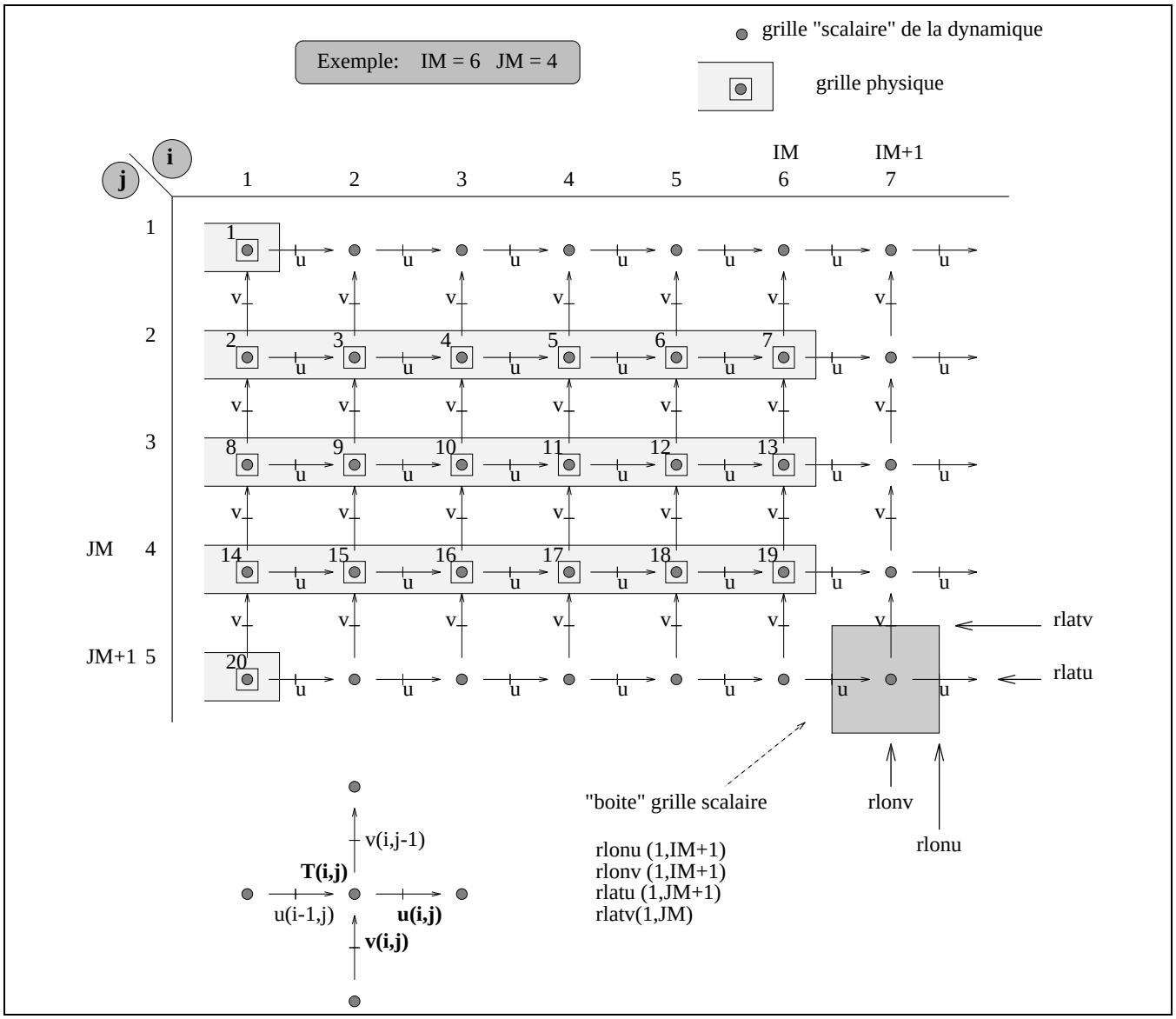

Figure 2.2: Dynamical and physical grids for a 6×7 horizontal resolution. In the dynamics (but not in the physics) winds u and v are on specific staggered grids. Other dynamical variables are on the dynamical "scalar" grid. The physics uses the same "scalar" grid for all the variables, except that nodes are indexed in a single vector containing NGRID=2+(JM-1)$\times$IM points when counting from the north pole.
N.B.: In the Fortran program, the following variables are used: `iim=IM , iip1=IM+1, jjm=JM , jjp1=JM+1`.

at the same place). Similarly, the extreme `j=1` and `j=JM+1` nodes on the dynamical grid (respectively corresponding to North and South poles) are duplicated IM+1 times.
In contrast, the physical grid does not contain redundant points (only one value for each pole and no extra point along longitudes), as shown in figure 2.2. In practice, computations relative to the physics are made for a series of `ngrid` atmospheric columns, where `NGRID`=`IM`x(`JM`-1)+2.

2.3.2 Vertical grids

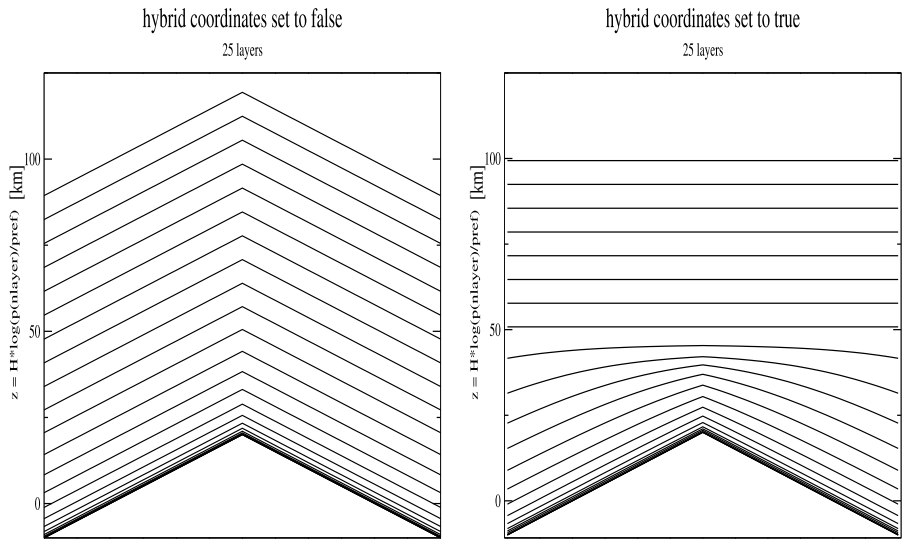

Figure 2.3: Sketch illustrating the difference between hybrid and non-hybrid coordinates

The GCM was initially programmed using sigma coordinates $\sigma = p/ps$ (atmospheric pressure over surface pressure ratio) which had the advantage of using a constant domain ($\sigma = 1$ at the surface and $\sigma = 0$ at the top of the atmosphere) whatever the underlying topography. However, it is obvious that these coordinates significantly disturb the stratospheric dynamical representation as the topography is propagated to the top of the model by the coordinate system. This problem can elegantly be solved by using a hybrid sigma-P (sima-pressure) hybrid coordinate which is equivalent to using σ coordinates near the surface and gradualy shifting to purely pressure p coordinates with increasing altitude. Figure 2.3 illustrates the importance of using these hybrid coordinates compared to simple σ coordinates. The distribution of the vertical layers is irregular, to enable greater precision at ground level. In general we use 25 levels to describe the atmosphere to a height of 80 km, 32 levels for simulations up to 120 km, or 50 levels to rise up to thermosphere. The first layer describes the first few meters above the ground, whereas the upper layers span several kilometers. Figure 2.4 describes the vertical grid representation and associated variables.

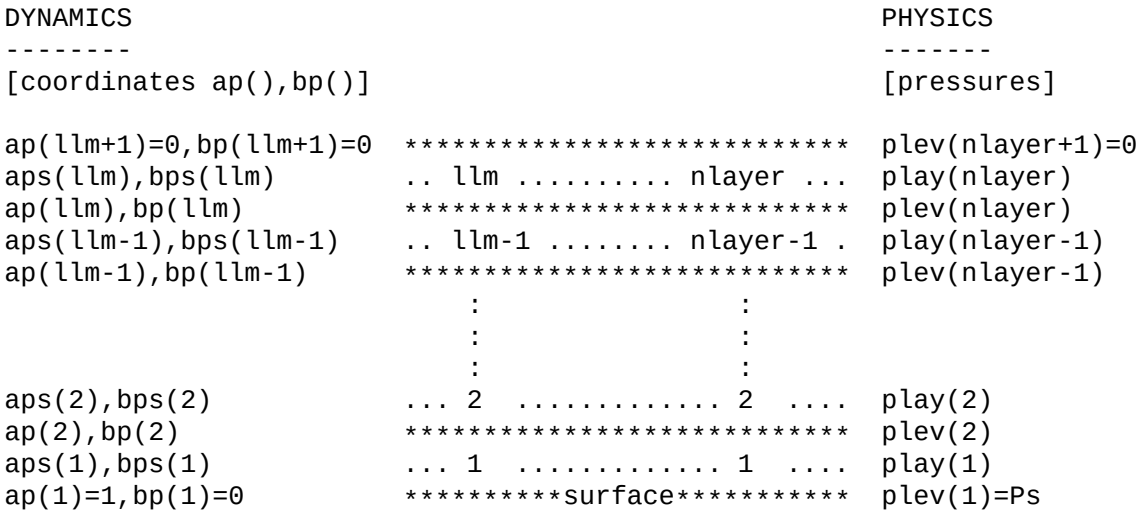

Figure 2.4: Vertical grid description of the `llm (or nlayer)` atmospheric layers in the programming code (`llm` is the variable used in the dynamical part, and `nlayer` is used in the physical part). Variables `ap, bp` and `aps, bps` indicate the hybrid levels at the interlayer levels and at middle of the layers respectively. Pressure at the interlayer is $Plev(l) = ap(l) + bp(l) \times Ps$ and pressure in the middle of the layer is defined by $Play(l) = aps(l) + bps(l) \times Ps$, (where Ps is surface pressure). Sigma coordinates are merely a specific case of hybrid coordinates such that $aps = 0$ and $bps = P/Ps$. Note that for the hybrid coordinates, $bps = 0$ above ~ 50 km, leading to purely pressure levels. The user can choose whether to run the model using hybrid coordinates or not by setting variable `hybrid` in run.def to True or False.

2.4 Variables used in the model

2.4.1 Dynamical variables

The dynamical state variables are the atmospheric temperature, surface pressure, winds and tracer concentrations. In practice, the formulation selected to solve the equations in the dynamics is optimised using the following less "natural" variables:

- **potential temperature** θ (`teta` in the code), linked to temperature **T** by $\theta = T(P/Pref)^{-\kappa}$ with $\kappa = R/C_p$ (note that κ is called `kappa` in the dynamical code, and `rcp` in the physical code). We take $Pref = 610$ Pa on Mars.

- **surface pressure** (`ps` in the code).

- **mass** the atmosphere mass in each grid box (`masse` in the code).

- **the covariant meridional and zonal winds** `ucov` and `vcov`. These variables are linked to the "natural" winds by `ucov = cu * u` and `vcov = cv * v`, where `cu` and `cv` are constants that only depend on the latitude.

- **mixing ratio of tracers** in the atmosphere, typically expressed in kg/kg (array `q` in the code).

`ucov` and `vcov`, "vectorial" variables, are stored on "scalari" grids u and v respectively, in the dynamics (see section 2.2). `teta`, `q`, `ps`, `masse`, "scalar variables", are stored on the "scalar" grid of the dynamics.

2.4.2 Physical variables

In the physics, the state variables of the dynamics are transmitted via an interface that interpolates the winds on the scalar grid (that corresponds to the physical grid) and transforms the dynamical variables into more "natural" variables. Thus we have winds **u** and **v** (m.s^{-1}), temperature **T** (K), pressure at the middle of the layers **play** (Pa) and at interlayers **plev** (Pa), tracers **q**, etc. (kg/kg) on the same grid.

Furthermore, the physics also handle the evolution of the purely physical state variables:

- **co2ice** CO_2 ice on the surface (kg.m^{-2})

- **tsurf** surface temperature (K),

- **tsoil** temperature at different layers under the surface (K),

- **emis** surface emissivity,

- **q2** wind variance, or more precisely the square root of the turbulent kinetic energy.

- **qsurf** tracer on the surface (kg.m^{-2}).

2.4.3 Tracers

The model may include different types of tracers:

- dust particles, which may have several modes

- chemical species which depict the chemical composition of the atmosphere

- water vapor and water ice particles

- ...

In the code, all tracers are stored in one three-dimensional array `q`, the third index of which corresponds to each individual tracer. In input and output files ("start.nc", "startfi.nc", see Section 4) tracers are stored seperately using their individual names. Loading specific tracers requires that the approriate tracer names are set in the `traceur.def` file (see Section 6.2.3), and specific computations for given tracers (e.g.: computing the water cycle, chemistry in the upper atmosphere, ...) is controled by setting coresponding options in the `callphys.def` file (see Section 6.2.2).

Chapter 3

The physical parameterizations of the Martian model: some references

3.1 General

The Martian General Circulation Model uses a large number of physical parameterizations based on various scientific theories and some generated using specific numerical methods.

A list of these parameterizations is given below, along with the most appropriate references for each one. Most of these documents can be consulted at: `http://www.lmd.jussieu.fr/mars.html`.

General references:

Two documents attempt to give a complete scientific description of the current version of the GCM (a version without tracers):

- *Forget et al.* [1999] (article published in the JGR)
- "Updated Detailed Design Document for the Model" (ESA contract, Work Package 6, 1999, available on the web) which is simply a compilation of the preceding article with a few additions that were published separately.

3.2 Radiative transfer

The radiative transfer parameterizations are used to calculate the heating and cooling ratios in the atmosphere and the radiative flux at the surface.

3.2.1 CO_2 gas absorption/emission:

Thermal IR radiation

(`lwmain`)

- New numerical method, solution for the radiative transfer equation: *Dufresne et al.* [2005].
- Model validation and inclusion of the "Doppler" effect (but using an old numerical formulation): *Hourdin* [1992] (article).

- At high altitudes, parameterization of the thermal radiative transfer (`nltecool`) when the local thermodynamic balance is no longer valid (e.g. within 0.1 Pa) : Lopez-Valverde et al. [2001] : Report for the ESA available on the web as: "CO2 non-LTE cooling rate at 15-um and its parameterization for the Mars atmosphere".

Absorption of near-infrared radiation

(`nirco2abs`)

- *Forget et al.* [1999]

3.2.2 Absorption/emission and diffusion by dust:

Dust spatial distribution

(`dustopacity`)

- Vertical distribution and description of "MGS" and "Viking" scenarios in the ESA report *Mars Climate Database V3.0 Detailed Design Document* by Lewis et al. (2001), available on the web.
- For the "MY24" scenario, dust distribution obtained from assimilation of TES data is used (and read via the `readtesassim` routine).

Thermal IR radiation

(`lwmain`)

- Numerical method: *Toon et al.* [1989]
- Optical properties of dust: *Forget* [1998]

Solar radiation

(`swmain`)

- Numerical method: *Fouquart and Bonel* [1980]
- Optical properties of dust: see the discussion in *Forget et al.* [1999], which quotes *Ockert-Bell et al.* [1997] and *Clancy and Lee* [1991].

3.3 Subgrid atmospheric dynamical processes

3.3.1 Turbulent diffusion in the upper layer

(`vdifc`)

- Implicit numerical scheme in the vertical: see the thesis of Laurent Li (LMD, Université Paris 7, 1990), Appendix C2.
- Calculation of the turbulent diffusion coefficients: *Forget et al.* [1999].

3.3.2 Convection

(`convadj`)
See *Hourdin et al.* [1993]

3.3.3 Effects of subgrid orography and gravity waves

(`calldrag_noro` , `drag_noro`)
See *Forget et al.* [1999] and *Lott and Miller* [1997]

3.4 Surface thermal conduction

(`soil`)
Thesis of Frédéric Hourdin (LMD, Université Paris 7, 1992) : section 3.3 (equations) and Appendix A (Numerical scheme).

3.5 CO_2 Condensation

In *Forget et al.* [1998] (article published in Icarus):
- Numerical method for calculating the condensation and sublimation levels at the surface and in the atmosphere (`newcondens`) explained in the appendix.
- Description of the numerical scheme for calculating the evolution of CO_2 snow emissivity (`co2snow`) explained in section 4.1

3.6 Tracer transport and sources

- "Van-Leer" transport scheme used in the dynamical part (`tracvl` and `vlsplt` in the dynamical part): *Hourdin and Armengaud* [1999]
- Transport by turbulent diffusion (in `vdifc`), convection (in `convadj`), sedimentation (`sedim`), dust lifting by winds (`dustlift`) : see note "Preliminary design of dust lifting and transport in the Model" (ESA contract, Work Package 4, 1998, available on the web).
- Dust lifting by "Dust devils" (`dustdevil`) *Rennò et al.* [1998].
- Dust transport by the "Mass mixing ratio / Number mixing ratio" method for grain size evolution: article by F. Forget in progress
- **Watercycle**, see *Montmessin et al.* [2004]
- **Chemistry**, see *Lefèvre et al.* [2004]

3.7 Thermosphere

- See *Angelats i Coll et al.* [2005] and *González-Galindo et al.* [2005]

Chapter 4

Running the model: a practice simulation

This chapter is meant for first time users of the LMD model. As the best introduction to the model is surely to run a simulation, here we explain how to go about it. All you will need are files necessary to build the GCM (all are in the `LMDZ.MARS` directory) as well as some initial states to initiate simulations (see below).
Once you have followed the example given below, you can then go on to change the control parameters and the initial states as you wish. A more detailed description of the model's organization as well as associated inputs and outputs are given in sections 5 and 6.

4.1 Installing the model

- Copy the basic model directory LMDZ.MARS to your account (the contents of this directory are described in chapter 5).

- Set some environment variables needed for the compilation of the model:

  LMDGCM : Path to the directory where you have put the model (full path).
  If using Csh:

  ```
  setenv  LMDGCM /where/you/put/the/model/LMDZ.MARS
  ```

  If using Bash:

  ```
  export  LMDGCM=/where/you/put/the/model/LMDZ.MARS
  ```

  LIBOGCM : Path to the directory (`libo` for example) where intermediate objects will be stored during the compilation of the model with the `makegcm` script (if that directory does not exist then `makegcm` will create it).
  If using Csh:

  ```
  setenv  LIBOGCM /where/you/want/objects/to/go/libo
  ```

  If using Bash:

  ```
  export  LIBOGCM=/where/you/want/objects/to/go/libo
  ```

- Install NetCDF and set environment variables **NCDFINC** and **NCDFLIB**:

The latest version of the NetCDF package is available on the web at the following address:

```
http://www.unidata.ucar.edu/software/netcdf
```

along with instructions for building (or downloading precompiled binaries of) the library.

Once the NetCDF library has been compiled (or downloaded), you should have access to the library `libnetcdf.a` itself, the various files (`netcdf.inc`, `netcdf.mod`, ...) to include in programs, and basic NetCDF software (*ncdump* and *ncgen*).

To ensure that during compilation, the model can find the NetCDF library and include files, you must declare environment variables **NCDFLIB** and **NCDFINC**.

NCDFLIB must contain the path to the directory containing the object library `libnetcdf.a` and **NCDFINC** must contain the path to the directory containing the include files (`netcdf.inc`,...)
If using Csh:

```
setenv NCDFINC /wherever/is/netcdf/include
setenv NCDFLIB /wherever/is/netcdf/lib
```

If using Bash:

```
export NCDFINC=/wherever/is/netcdf/include
export NCDFLIB=/wherever/is/netcdf/lib
```

- Install software with which can load and display NetCDF files such as GrAdS or Ferret

For people working at LMD, thanks to the brilliant Laurent Fairhead, Grads and Ferret are installed and ready to go.

- Create an alias so that the compilation script **makegcm** is available from anywhere (more convinient than having to type the full path to the script, or copying it over where you want to run it). The `makegcm` script is in the LMDZ.MARS directory, which is referenced by the **LMDGCM** variable, so:
If using Csh:

```
alias makegcm $LMDGCM'/makegcm'
```

if using Bash:

```
alias makegcm=$LMDGCM/makegcm
```

- Finally, make sure that you have access to all the executables needed for building and using the model and remember to set environment variables to the correct corresponding pathes (note that if you do not want to have to redefine these every session, you should put the definitions in the corresponding `.cshrc` or `.bashrc` files).

  - UNIX function ımake
  - a Fortran compiler
  - ncdump
  - grads (or ferret)

4.2 Compiling the model

- Example 1: Compiling the Martian model at grid resolution 64x48x25 for example, type (in compliance with the manual for the makegcm function given in section 5.4)

```
makegcm -d 64x48x25 -p mars gcm
```

You can find executable **gcm.e** (the compiled model) in the directory where you ran the makegcm command.

- Example 2: Compiling the Martian model with 2 tracers (e.g. water vapour and ice to simulate the water cycle):

```
makegcm -d 64x48x25 -t 2 -p mars gcm
```

- Example 3: Compiling the the Martian model to check for array overflow (useful for debugging: warning, the model then runs very slowly!):

```
makegcm -d 64x48x25 -p mars -O "-C" gcm
```

4.3 Input files (initial states and def files)

- In directory `LMDZ.MARS/deftank` you will find some examples of run parameter files (`.def` files) which the model needs at runtime. The four files the model requires (they must be in the same directory as the executable `gcm.e`) are: **run.def** (described in section 6.2) **callphys.def** (see section 6.2.2), **callphys.def**, **z2sig.def** and **traceur.def**.

The example `.def` files given in the `deftank` directory are for various configurations (e.g. model resolution), copy (and eventually rename these files to match the generic names) to the directory where you will run the model.

- Copy initial condition files **start.nc** and startfi.nc (described in section 6.2) to the same directory.
You can extract such files from **start_archive** 'banks of initial states' (i.e. files which contain collections of initial states from stndard scenarios and which can thus be used to check if the model is installed correctly) stored on the LMD website at `http://www.lmd.jussieu.fr/~forget/datagcm/Starts`. See section 4.8 for a description of how to proceed to extract **start** files from **start_archives**.

4.4 Running the model

Once you have the program **gcm.e**, input files **start.nc startfi.nc**, and parameter files **run.def callphys.def traceur.def z2sig.def** in the same directory, simply execute the program to run a simulation:

```
gcm.e
```

You might also want to keep all messages and diagnotics written to standard output (i.e. the screen). You should then redirect the standard output (and error) to some file, e.g. `gcm.out`:
If using Csh:

```
gcm.e >! gcm.out
```

If using Bash:

```
gcm.e > gcm.out 2>&1
```

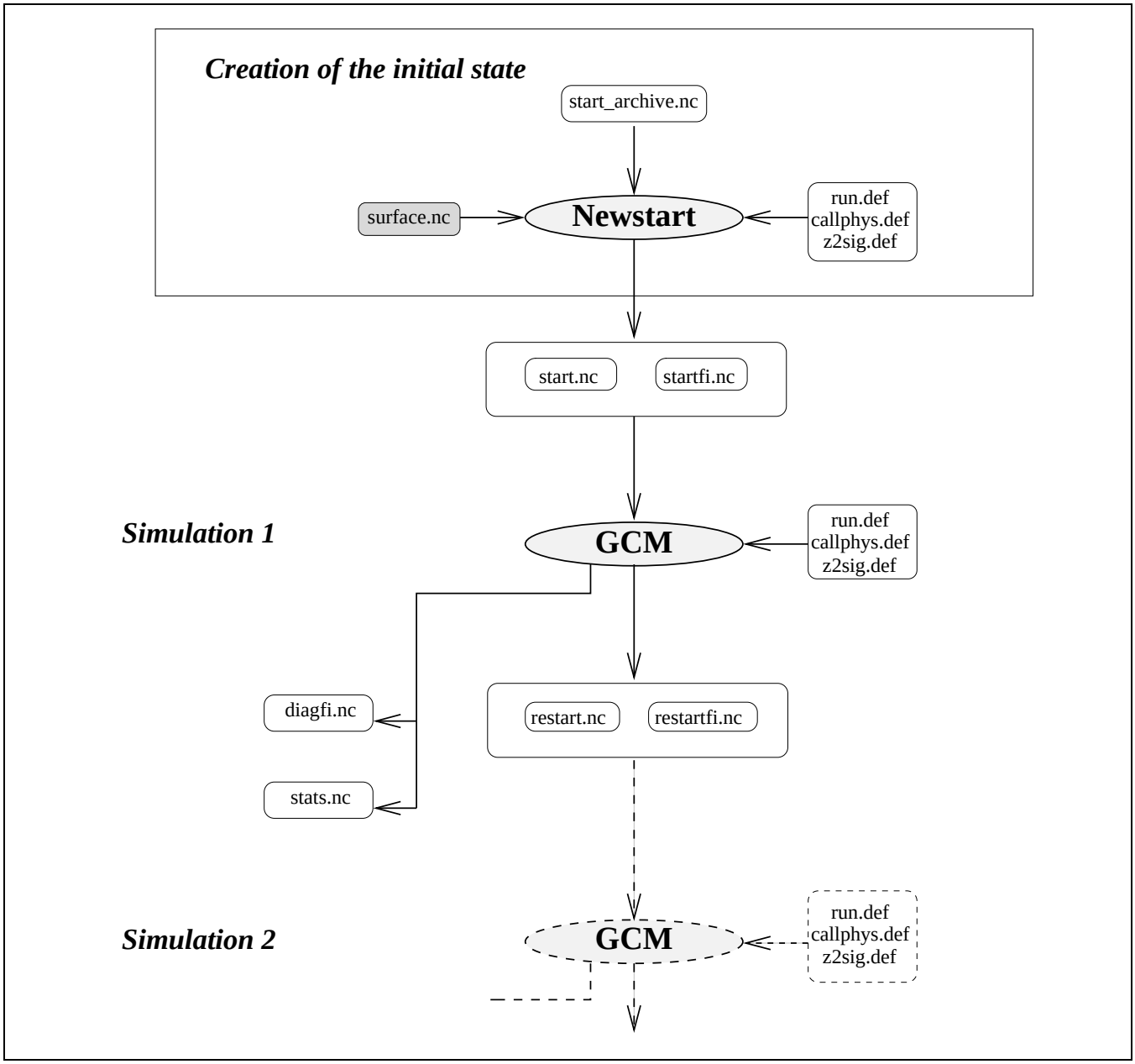

Figure 4.1: Input/output data

4.5 Visualizing the output files

As the model runs it generates output files **diagfi.nc** and **stats.nc** files. The former contains instantaneous values of various fields and the later statistics (over the whole run) of some variables.

4.5.1 Using GrAds to visualize outputs

If you have never used the graphic software **GrAds**, we strongly recommend spending half an hour to familiarize yourself with it by following the demonstration provided for that purpose. The demo is fast and easy to follow and you will learn the basic commands. To do this read file

`/distrib/local/grads/sample`

For example, to visualize files `diagfi.nc` and `stats.nc`

NetCDF files `diagfi.nc` and `stats.nc` can be accessed directly using GrAdS thanks to utility program gradsnc, (the user does not need to intervene).

To visualize the temperature in the 5th layer using file `diagfi.nc` for example:

- GrAdS session:

  `grads` *return*

  return (opens a landscape window)

  `ga-> sdfopen diagfi.nc`

  `ga-> query file` (displays info about the open file, including the name of the stored variables. Shortcut: *q file*)

  `ga-> set z 5` (fixes the altitude to the 5th layer)

  `ga-> set t 1` (fixes the time to the first stored value)

  `ga-> query dims` (indicates the fixed values for the 4 dimensions. Shortcut: *q dims*)

  `ga-> display temp` (displays the temperature card for the 5th layer and for the first time value stored. Shortcut: *d T*)

  `ga-> clear` (clears the display. Shortcut: *c*)

  `ga-> set gxout shaded` (not a contour plot, but a shaded one)

  `ga-> display temp`

  `ga-> set gxout contour` (returns to contour mode to display the levels)

  `ga-> display temp` (superimposes the contours if the clear command is not used)

4.6 Resuming a simulation

At the end of a simulation, the model generates **restart** files (files `restart.nc` and `restartfi.nc`) which contain the final state of the model. As shown in figure 4.1, these files (which are of the same format as the start files) can later be used as initial states for a new simulation.

The **restart** files just need to be renamed:

```
mv restart.nc start.nc
mv restartfi.nc startfi.nc
```

and running a simulation with these will in fact resume the simulation from where the previous run ended.

4.7 Chain simulations

In practice, we recommend running a chain of simulations lasting several days or longer (or hundreds of days at low resolution).

To do this, a script named `run0` is available in `LMDZ.MARS/deftank`, which should be used as follows:

- Set the length of each simulation in `run.def` (i.e. set the value of `nday`)
- Set the maximum number of simulations at the beginning of the `run0` script (i.e. set the value of `nummax`)
- Copy start files `start.nc startfi.nc` over and rename them `start0.nc startfi0.nc`.
- Run script `run0`

`run0` runs a series of simulations that generate the indexed output files (e.g. `start1, startfi1, diagfi1`, etc.) including files `lrun1, lrun2`, etc. containing the redirection of the display and the information about the run.

NOTE: to restart a series of simulations after a first series (for example, starting from `start5 and startfi5`), just write the index of the initial files (e.g. 5) in the file named `num_run`. If `num_run` exists, the model will start from the index written in `num_run`. If not it will start from, `start0 and startfi0`.

NOTE: A script is available for performing annual runs with 12 seasons at 30^o solar longitude as it is in the database (script `run_mcd`, also found in directory `deftank`). This script functions with script run0. Just set the number of simulations to 1 in run0. Then copy run.def into run.def.ref and set nday to 9999 in this file. To start from startN.c, edit the file run_mcd and comment (with a #) the N months already created and describe N in `num_run`. Then run `run_mcd`.

4.8 Creating and modifying initial states

4.8.1 Using program "newstart"

Several model parameters (for example, the dust optical depth) are stored in the initial states (NetCDF files `start.nc` and `startfi.nc`). To change these parameters, or to generally change the model resolution, use program **newstart**.

This program is also used to create an initial state. In practice, we usually reuse an old initial state, and modify it using **newstart**.

Like the GCM, program **newstart** must be compiled (using the `makegcm` script) to the required grid resolution. For example:

```
makegcm -d 64x48x25 -p mars newstart
```

Then run

```
newstart.e
```

The program then gives you two options:

```
 A partir de quoi souhaitez vous creer vos etats initiaux ?
     0 - d un fichier start_archive
     1 - d un fichier start
```

- - Option "1" allows you to read and modify the information needed to create a new initial state from the files `start.nc, startfi.nc`

- - Option “0” allows you to read and modify the information needed to create a new initial state from file `start_archive.nc` (whatever the `start_archive.nc` grid resolution is).

If you use tracers, make sure that they are taken into account in your start files (either start or start_archive).

Then answer to the various questions in the scroll menu. These questions allow you to modify the initial state for the following parameters.

```
First set of questions:
Modifications of variables in tab_cntrl:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  day_ini : Jour initial (=0 a Ls=0)
  z0 :  surface roughness (m)
  emin_turb :  energie minimale
  lmixmin : longueur de melange
  emissiv : Emissivite du sol martien
  emisice : Emissivite des calottes
  albedice : Albedo des calotte
  iceradius : mean scat radius of CO2 snow
  dtemisice : time scale for snow,
    . ' metamorphism
       tauvis : profondeur optique visible,
    . ' moyenne
  obliquit : planet obliquity (deg)
  peri_day : perihelion date (sol since Ls=0)
  periheli : min. sun-mars dist (Mkm)
  aphelie  : max. sun-mars dist (Mkm)

Second set of questions :
  flat : no topography ("aquaplanet")
  bilball : albedo, inertie thermique uniforme
  coldspole : sous sol froid et haut albedo au pole sud
  q=0 : traceurs a zero
  ini_q : traceurs initialises pour la chimie
  ini_q-H2O : idem, sauf le dernier traceur (H2O)
  ini_q-iceH2O : idem, sauf ice et H2O
  watercapn : H20 ice sur la calotte permanente nord
  watercaps : H20 ice sur la calotte permanente sud
  wetstart  : start with a wet atmosphere
  iqset     : give a specific value to tracer iq
  isotherm : Temperatures isothermes et vents nuls
  co2ice=0 : elimination des calottes polaires de CO2
  ptot : pression totale
```

Program **newstart.e** creates files `restart.nc` and `restartfi.nc` that you generally need to rename (for instance rename them in start0.nc and startfi0.nc if you want to use run0 or run_mcd, starting with season 0; rename them `start.nc` and `startfi.nc` if you just want to perform one run with `gcm.e`).

4.8.2 Creating the initial start_archive.nc file

Archive file `start_archive.nc` is created from files `start.nc` and `startfi.nc` by program **start2archive**. Program **start2archive** compiles to the same grid resolution as the `start.nc` and `startfi.nc` grid resolution. For example:

```
makegcm -d 64x48x25 -p mars start2archive
```

Then run `start2archive.e`

You now have a `start_archive.nc` file for one season that you can use with newstart. If you want to gather other states obtained at other times of year, rerun `start2archive.e` with the `start.nc` and `startfi.nc` corresponding to these. These additional initial states will automatically be added to the `start_archive.nc` file present in the directory.

4.8.3 Changing the horizontal or vertical grid resolution

To run at a different grid resolution than available initial conditions files, one needs to use tools **newstart** and **start2archive**

For example, to create initial states at grid resolution $32\times24\times25$ from NetCDF files `start` and `startfi` at grid resolution $64\times48\times32$:

- Create file `start_archive.nc` with **start2archive.e** compiled at grid resolution $64\times48\times25$ using **old file** `z2sig.def` **used previously**
- Create files `newstart.nc` and `newstartfi.nc` with **newstart.e** compiled at grid resolution $32\times24\times25$, using **new file** `z2sig.def`

If you want to create starts files with tarcers for 50 layers using a `start_archive.nc` obtained for 32 layers, do not forget to use the `ini_q` option in newstart in order to correctly initialize tracers value for layer 33 to layer 50. You just have to answer yes to the question on thermosphere initialization if you want to initialize the thermosphere part only (l=33 to l=50), and no if you want to initialize tracers for all layers (l=0 to l=50).

Chapter 5

Program organization and compilation script

All the elements of the LMD model are in the **LMDZ.MARS** directory (and subdirectories). As explained in Section 4, this directory should be associated with environment variable **LMDGCM**:
If using Csh:

```
setenv  LMDGCM /where/you/put/the/model/LMDZ.MARS
```

If using Bash:

```
export  LMDGCM=/where/you/put/the/model/LMDZ.MARS
```

is a brief description of the **LMDZ.MARS** directory contents:

```
libf/  All the model FORTRAN Sources (.F or .F90)
       and  include files (.h) organised in sub-directories
      (physics (phymars), dynamics (dyn3d), filters (filtrez)...)

deftank/   A collection of examples of parameter files required
              to run the GCM (run.def, callphys.def, ...)

makegcm  Script that should be used to compile the GCM as well
          as related utilities (newstart, start2archive, testphys1d)

create_make_gcm   Executable used to create the makefile.
                  This command is run automatically by
                  "makegcm" (see below).
```

5.1 Organization of the model source files

The model source files are stored in various sub directories in directory **libf**. These subdirectories correspond to the different parts of the model:

grid: mainly made up of ”dimensions.h” file, which contains the parameters that define the model grid, i.e. the number of points in longitude (IIM), latitude (JJM) and altitude (LLM), as well as the number of tracers (NQMX).

dyn3d: contains the dynamical subroutines.

bibio: contains some generic subroutines not specifically related to physics or dynamics but used by either or both.

phymars: contains the Martian physics routines.

filtrez: contains the longitudinal filter sources applied in the upper latitudes, where the Courant-Friedrich-Levy stability criterion is violated.

aeronomars: contains the Martian chemistry and thermosphere routines.

5.2 Programming

The model is written in **Fortran-77** and **Fortran-90**.

- The program sources are written in **"file.F"** or **"file.F90"** files. The extension .F is the standard extension for fixed-form Fortran and the extension .F90 is for free-form Fortran. These files must be preprocessed (by a**C preprocessor** such as (cpp)) before compilation (this behaviour is, for most compilers, implicitly obtained but using a capital F in the extention of the file names).
- Constants are placed in COMMON declarations, located in the common "include" files **"file.h"**
- In general, variables are passed from subroutine to subroutine as arguments (and never as COMMON blocks).
- In some parts of the code, for "historical" reasons, the following rule is sometimes used: in the subroutine, the variables (ex: `name`) passed as an argument by the calling program are given the prefix `p` (ex: `pname`) while the local variables are given the prefix `z` (ex: `zname`). As a result, several variables change their prefix (and thus their name) when passing from a calling subroutine to a called subroutine.

5.3 Model organization

Figure 5.1 describes the main subroutines called by physiq.F.

5.4 Compiling the model

Technically, the model is compiled using the Unix utility `make`. The file `makefile`, which describes the code dependencies and requirements, is created automatically by the script

```
create_make_gcm
```

This utility script recreates the `makefile` file when necessary, for example, when a source file has been added or removed since the last compilation.

None of this is visible to the user. To compile the model just run the command

```
makegcm
```

with adequate options (e.g. `makegcm -d 62x48x32 -p mars gcm`), as discussed below and described in section 4.2.

The `makegcm` command compiles the model (**gcm**) and related utilities (**newstart**, **start2archive**, **testphys1d**). A detailed description of how to use it and of the various parameters that can be supplied is given in the help manual below (which will also be given by the `makegcm -h` command).

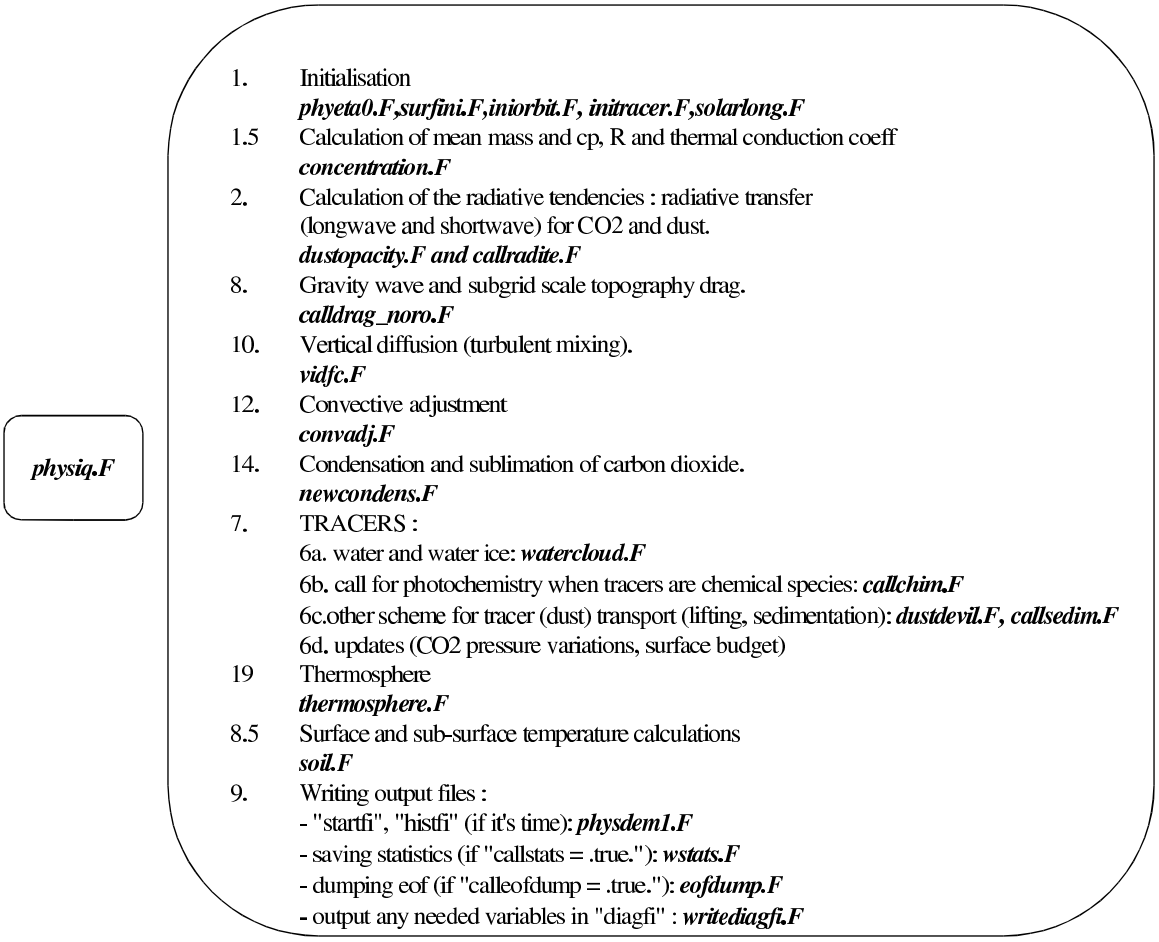

Figure 5.1: Organigram of subroutine function physiq.F

Note that before compiling the GCM with `makegcm` you should have set the environment variable **LIBOGCM** to a path where intermediate objects and libraries will be generated.
If using Csh:

```
setenv  LIBOGCM /where/you/want/objects/to/go/libo
```

If using Bash:

```
export  LIBOGCM=/where/you/want/objects/to/go/libo
```

Help manual for the makegcm script

```
makegcm [Options] prog


The makegcm script:
-------------------

1. compiles a series of subroutines located in the $LMDGCM/libf
 sub-directories.
 The objects are then stored in the libraries in $LIBOGCM.

2. then, makegcm compiles program prog.f located by default in
$LMDGCM/libf/dyn3d and makes the link with the libraries.

Environment Variables '$LMDGCM' and '$LIBOGCM'
 must be set as environment variables or directly
 in the makegcm file.

The makegcm command is used to control the different versions of the model
 in parallel, compiled using the compilation options
 and the various dimensions, without having to recompile the whole model.

The FORTRAN libraries are stored in directory $LIBOGCM.


OPTIONS:
--------

The following options can either be defined by default by editing the
makegcm "script", or in interactive mode:

-d imxjmxlm  where im, jm, and lm are the number of longitudes,
             latitudes and vertical layers respectively.

-t ntrac   Selects the number of tracers present in the model

             Options -d and -t overwrite file
             $LMDGCM/libf/grid/dimensions.h
             which contains the 3 dimensions of the
             horizontal grid
             im, jm, lm plus the number of tracers passively advected
             by the dynamics ntrac,
             in 4 PARAMETER FORTRAN format
             with a new file:
             $LMDGCM/libf/grid/dimension/dimensions.im.jm.lm.tntrac
             If the file does not exist already
             it is created by the script
             $LMDGCM/libf/grid/dimension/makdim

-p PHYS    Selects the set of physical parameterizations
           you want to compile the model with.
           The model is then compiled using the physical
           parameterization sources in directory:
            $LMDGCM/libf/phyPHYS

-g grille  Selects the grid type.
```

```
          This option overwrites file
          $LMDGCM/libf/grid/fxyprim.h
          with file
          $LMDGCM/libf/grid/fxy_grille.h
          the grid can take the following values:
          1. reg - the regular grid
          2. sin - to obtain equidistant points in terms of sin(latitude)
          3. new - to zoom into a part of the globe

-O "compilation options" set of fortran compilation options to use

-include path
          Used if the subroutines contain #include files (ccp) that
          are located in directories that are not referenced by default.

-adjnt    Compiles the adjoint model to the dynamical code.

-filtre  filter
          To select the longitudinal filter in the polar regions.
          "filter" corresponds to the name of a directory located in
          $LMDGCM/libf. The standard filter for the model is "filtrez"
          which can be used for a regular grid and for a
          grid with longitudinal zoom.

-link "-Ldir1 -lfile1 -Ldir2 -lfile2 ..."
          Adds a link to FORTRAN libraries
          libfile1.a, libfile2.a ...
          located in directories dir1, dir2 ...respectively
          If dirn is a directory with an automatic path
          (/usr/lib ... for example)
          there is no need to specify  -Ldirn.
```

Chapter 6

Input/Output

6.1 NetCDF format

GCM input/output data are written in **NetCDF** format (Network Common Data Form). NetCDF is an interface used to store and access geophysical data, and a library that provides an implementation of this interface. The NetCDF library also defines a machine-independent format for representing scientific data. Together, the interface, library and format support the creation, access and sharing of scientific data. NetCDF was developed at the Unidata Program Center in Boulder, Colorado. The freely available source can be obtained from the Unidata website.

`http://www.unidata.ucar.edu/software/netcdf`

A data set in NetCDF format is a single file, as it is self-descriptive.

6.1.1 NetCDF file editor: ncdump

The editor is included in the NetCDF library. By default it generates an ASCII representation as standard output from the NetCDF file specified at the input.

Main commands for ncdump

ncdump diagfi.nc

dump contents of NetCDF file `diagfi.nc` to standard output (i.e. the screen).

ncdump -c diagfi.nc

Displays the **coordinate** variable values (variables which are also dimensions), as well as the declarations, variables and attribute values. The values of the non-coordinate variable data are not displayed at the output.

ncdump -h diagfi.nc

Shows only the informative header of the file, which is the declaration of the dimensions, variables and attributes, but not the values of these variables. The output is identical to that in option **-c** except for the fact that the coordinated variable values are not included.

ncdump -v var1,...,varn diagfi.nc

The output includes the specific variable values, as well as all the dimensions, variables and attributes. More that one variable can be specified in the list following this option. The list must be a simple argument for the command, and must not contain any spaces. If no variable is specified, the command displays all the values of the variables in the file by default.

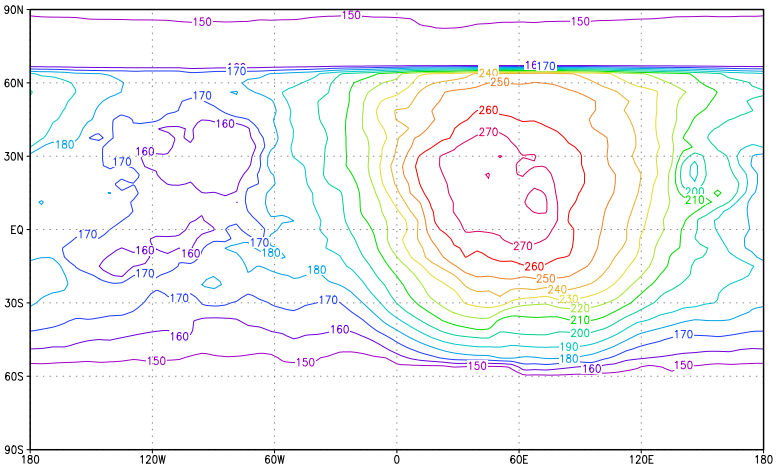

Figure 6.1: Example of temperature data at a given time using GrADS visualization

6.1.2 Graphic visualization of the NetCDF files using GrAds

GrAdS (The Grid Analysis and Display System) is a graphic software developed by Brian Doty at the "Center for Ocean-Land-Atmosphere (COLA)".

One of its functions is to enable data stored in NetCDF format to be visualized directly. In figure 6.1 for example, we can see the GrADS visualization of the temperature data at a given moment. However, unlike NetCDF, GrADS only recognizes files where all the variables are stored on the same horizontal grid. These variables can be in 1, 2, 3 or 4 dimensions (X,Y,Z and t).

GrADS can also be obtained on the WWW.

`http://grads.iges.org/grads/`

6.2 Input and parameter files

The (3D version of the) GCM requires the input of two initialization files (in NetCDF format):
-**start.nc** contains the initial states of the dynamical variables.
-**startfi.nc** contains the initial states of the physical variables.
Note that collections of initial states can be retreived at:
`http://www.lmd.jussieu.fr/~forget/datagcm/Starts`
Extracting `start.nc` and `startfi.nc` from these archived requires using program `newstart`, as described in section 4.8.

To run, the GCM also requires the four following parameter files (ascii text files):
-**run.def** the parameters of the dynamical part of the program, and the temporal integration of the model.
-**callphys.def** the parameters for calling the physical part.
-**traceur.def** the names of the tracer to use.
-**z2sig.def** the vertical distribution of the atmospheric layers.
Examples of these parameter files can be found in the `LMDZ.MARS/deftank` directory.

6.2.1 run.def

A typical `run.def` file is given as an example below. The choice of variables to be set is simple (e.g. `nday` number of modeled days to run), while the others do not need to be changed for normal use.
The format of the `run.def` file is quite straightforward (and flexible): values given to parameters must be given as:

```
parameter = value
```

Any blank line or line beginning with symbol # is a comment, and instruction lines may be written in any order. Moreover, not specifying a parameter/value set (e.g. deleting it or commenting it out) means you want the GCM to use a default built-in value. Additionally, one may use a specific keyword **INCLUDEDEF** to specify another (text) file in which to also read values of parameters; e.g.:

```
INCLUDEDEF=callphys.def
```

Here are some details about some of the parameters which may be set in `run.def`:

- **day_step**, the number of dynamical steps per day to use for the time integration. This needs to be large enough for the model to remain stable (this is related to the CFL stability criterion which essentially depends on the horizontal resolution of the model). On Mars, in theory, the GCM can run with `day_step`=480 using the 64×48 grid, but model stability improves when this number is higher: `day_step`=960 is recommended when using the 64×48 grid. According to the CFL criterion, `day_step` should vary in proportion with the resolution: for example `day_step`=480 using the 32×24 horizontal resolution. Note that `day_step` must also be divisible by `iperiod`.

- **tetagdiv, tetagrot, tetatemp** control the dissipation intensity. It is better to limit the dissipation intensity (tetagdiv, tetagrot, tetatemp should not be too low). However the model diverges if tetagdiv, tetagrot, tetatemp are too high, especially if there is a lot of dust in the atmosphere.
  Example used with nitergdiv=1 and nitergrot=niterh=2 :
  - using the 32×24 grid tetagdiv=6000 s ; tetagrot=tetatemp=30000 s
  - using the 64×48 grid: tetagdiv=3000 s ; tetagrot=tetatemp=9000 s

- **idissip** is the time step used for the dissipation: dissipation is computed and added every `idissip` dynamical time step. If `idissip` is too short, the model waste time in these calculations. But if idissip is too long, the dissipation will not be parametrized correctly and the model will be more likely to diverge. A check must be made, so that: `idissip` $<$ `tetagdiv`×`daystep`/88775 (same rule for `tetagrot` and `tetatemp`). This is tested automatically during the run.

- **iphysiq** is the time step used for the physics: physical tendencies are computed every `iphysiq` dynamical time step. In practice, we usually set the physical time step to be of the order of half an hour. We thus generally set `iphysiq`= `day_step`/48

Example of run.def file:

```
#
#-----------------------------------------------------------------------
#Parametres de controle du run:
#------------------------------

# Nombre de jours d'integration
     nday=9999

# nombre de pas par jour (multiple de iperiod) ( ici pour  dt = 1 min )
```

```
 day_step = 480

# periode pour le pas Matsuno (en pas)
  iperiod=5

# periode de sortie des variables de controle (en pas)
  iconser=120

# periode d'ecriture du fichier histoire (en jour)
    iecri=100

# periode de stockage fichier histmoy (en jour)
 periodav=60.

# periode de la dissipation (en pas)
  idissip=1

# choix de l'operateur de dissipation (star ou  non star )
 lstardis=.true.

# avec ou sans coordonnee hybrides
 hybrid=.true.

# nombre d'iterations de l'operateur de dissipation   gradiv
nitergdiv=1

# nombre d'iterations de l'operateur de dissipation  nxgradrot
nitergrot=2

# nombre d'iterations de l'operateur de dissipation  divgrad
   niterh=2

# temps de dissipation des plus petites long.d ondes pour u,v (gradiv)
 tetagdiv= 3000.

# temps de dissipation des plus petites long.d ondes pour u,v(nxgradrot)
 tetagrot=9000.

# temps de dissipation des plus petites long.d ondes pour  h ( divgrad)
 tetatemp=9000.

# coefficient pour gamdissip
  coefdis=0.

# choix du shema d'integration temporelle (Matsuno ou Matsuno-leapfrog)
  purmats=.false.

# avec ou sans physique
   physic=.true.

# periode de la physique (en pas)
  iphysiq=10

# choix d'une grille reguliere
  grireg=.true.

# frequence (en pas) de l'ecriture du fichier diagfi
 ecritphy=120

# longitude en degres du centre du zoom
   clon=63.

# latitude en degres du centre du zoom
   clat=0.

# facteur de grossissement du zoom,selon longitude
  grossismx=1.
```

```
# facteur de grossissement du zoom ,selon latitude
 grossismy=1.

#  Fonction  f(y)  hyperbolique  si = .true.  , sinon  sinusoidale
  fxyhypb=.false.

# extension en longitude  de la zone du zoom  ( fraction de la zone totale)
   dzoomx= 0.

# extension en latitude de la zone  du zoom  ( fraction de la zone totale)
   dzoomy=0.

#  raideur du zoom en  X
    taux=2.

#  raideur du zoom en  Y
    tauy=2.

#  Fonction  f(y) avec y = Sin(latit.) si = .TRUE. ,  Sinon  y = latit.
  ysinus= .false.

# Avec sponge layer
  callsponge  = .true.

# Sponge:  mode0(u=v=0), mode1(u=umoy,v=0), mode2(u=umoy,v=vmoy)
  mode_sponge= 2

# Sponge:  hauteur de sponge (km)
  hsponge= 90

# Sponge:  tetasponge (secondes)
  tetasponge = 50000

# some definitions for the physics, in file 'callphys.def'
INCLUDEDEF=callphys.def
```

6.2.2 callphys.def

The `callphys.def` file (along the same format as the `run.def` file) contains parameter/value sets for the physics.

Example of callphys.def file:

```
##General options
##~~~~~~~~~~~~~~~
#Run with or without tracer transport ?
tracer=.false.

#Diurnal cycle ?  if diurnal=False, diurnal averaged solar heating
diurnal=.true.

#Seasonal cycle ? if season=False, Ls stays constant, to value set in "start"
season = .true.

#write some more output on the screen ?
lwrite = .false.

#Save statistics in file "stats.nc" ?
stats =.true.

#Save EOF profiles in file "profiles" for Climate Database?
calleofdump = .false.
```

```
## Dust scenario. Used if the dust is prescribed (i.e. if tracer=F or active=F)
## ~~~~~~~~~~~~~
#  =1 Dust opt.deph read in startfi; =2 Viking scenario; =3 MGS scenario,
#  =4 Mars Year 24 from TES assimilation (same as =24 for now)
#  =24 Mars Year 24 from TES assimilation (ie: MCD reference case)
#  =25 Mars Year 25 from TES assimilation (ie: a year with a global dust storm)
#  =26 Mars Year 26 from TES assimilation
iaervar = 24
# Dust vertical distribution:
# (=0: old distrib. (Pollack90), =1: top set by "topdustref",
#  =2: Viking scenario; =3 MGS scenario)
iddist  = 3
# Dust top altitude (km). (Matters only if iddist=1)
topdustref = 55.

## Physical Parameterizations :
## ~~~~~~~~~~~~~~~~~~~~~~~~~~
# call radiative transfer ?
callrad   = .true.
# call NLTE radiative schemes ?   matters only if callrad=T
callnlte = .true.
# call CO2 NIR absorption ?   matters only if callrad=T
callnirco2 = .true.
# call turbulent vertical diffusion ?
calldifv  = .true.
# call convective adjustment ?
calladj  = .true.
# call CO2 condensation ?
callcond =.true.
# call thermal conduction in the soil ?
callsoil = .true.
# call Lott's gravity wave/subgrid topography scheme ?
calllott  = .true.

## Radiative transfer options :
## ~~~~~~~~~~~~~~~~~~~~~~~~~~
# the rad.transfer is computed every "iradia" physical timestep
iradia    = 1
# Output of the exchange coefficient mattrix ? for diagnostic only
callg2d  = .false.
# Rayleigh scattering : (should be .false. for now)
rayleigh  = .false.

## Tracer (dust water, ice and/or chemical species) options (used if tracer=T):
## ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# DUST: Transported dust ? (if >0, use 'dustbin' dust bins)
dustbin    = 0
# DUST: Radiatively active dust ? (matters if dustbin>0)
active  = .false.
# DUST: use mass and number mixing ratios to predict dust size ?
# (must also have dustbin=1)
doubleq   = .false.
# DUST: lifted by GCM surface winds ?
lifting = .false.
# DUST: lifted by dust devils ?
callddevil = .false.
# DUST: Scavenging by CO2 snowfall ?
scavenging = .false.
# DUST/WATERICE: Gravitationnal sedimentation ?
sedimentation = .false.
# WATERICE: Radiatively active transported atmospheric water ice ?
activice   = .false.
# WATER: Compute water cycle
water = .false.
# WATER: current permanent caps at both poles. True IS RECOMMENDED
#        (with .true., North cap is a source of water and South pole
#         is a cold trap)
```

```
caps  = .true.
# PHOTOCHEMISTRY: include chemical species
photochem  = .false.

## Thermospheric options (relevant if tracer=T) :
##~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# call thermosphere ?
callthermos = .false.
# WATER: included without cycle (only if water=.false.)
thermoswater = .false.
# call thermal conduction ? (only if callthermos=.true.)
callconduct = .false.
# call EUV heating ? (only if callthermos=.true.)
calleuv=.false.
# call molecular viscosity ? (only if callthermos=.true.)
callmolvis = .false.
# call molecular diffusion ? (only if callthermos=.true.)
callmoldiff = .false.
# call thermospheric photochemistry ? (only if callthermos=.true.)
thermochem = .false.
# date for solar flux calculation: (1985 < date < 2002)
## (Solar min=1996.4 ave=1993.4 max=1990.6)
solarcondate = 1993.4
```

6.2.3 traceur.def

Tracers in input (`start.nc` and `startfi.nc`) and output files (`restart.nc` and `restartfi.nc`) are stored using individual tracer names (e.g. `co2` for CO2 gas, `h2o_vap` for water vapour, `h2o_ice` for water ice, ...).
The first line of the `traceur.def` file (an ASCII file) must contain the number of tracers to load and use (this number should be the same as given to the `-t` option of the `makegcm` script when the GCM was compiled), followed by the tracer names (one per line). Note that if the corresponding tracers are not found in input files `start.nc` and `startfi.nc`, then the tracer is initialized to zero.

Example of a traceur.def file: (with water vapour and ice tracers)

```
2
h2o_ice
h2o_vap
```

6.2.4 z2sig.def

The `Z2sig.def` file contains the pseudo-altitudes (in km) at which the user wants to set the vertical levels.
Note that levels should be unevenly spread, with a higher resolution near the surface in order to capture the rapid variations of variables there. It is recommended to use the altitude levels as set in the `z2sig.def` file provided in the `deftank` directory.

Example of z2sig.def file (this version for 50 layers between 0 and 400 km):

```
10.00000     H: atmospheric scale height (km) (used as a reference only)
0.0040       Typical pseudo-altitude (m) for 1st layer (z=H*log(sigma))
0.018       ,, ,,  ,, ,, ,, ,,  ,, ,, ,,  2nd layer, etc...
0.0400
0.1000
0.228200
0.460400
0.907000
1.73630
```

3.19040
5.54010
8.97780
13.5138
18.9666
25.0626
31.5527
38.4369
45.4369
52.4369
59.4369
66.4369
73.4369
80.4369
87.4369
94.4369
101.4369
108.437
115.437
122.437
129.437
136.437
143.437
150.437
157.437
164.437
171.437
178.437
185.437
192.437
199.437
206.437
213.437
220.437
227.437
234.437
241.437
248.437
255.437
262.437
269.437
276.437
283.437
290.437
297.437
304.437
311.437
318.437
325.437
332.437
339.437
346.437
353.437
360.437

```
367.437
374.437
381.437
388.437
395.437
```

6.2.5 Initialization files: start and startfi

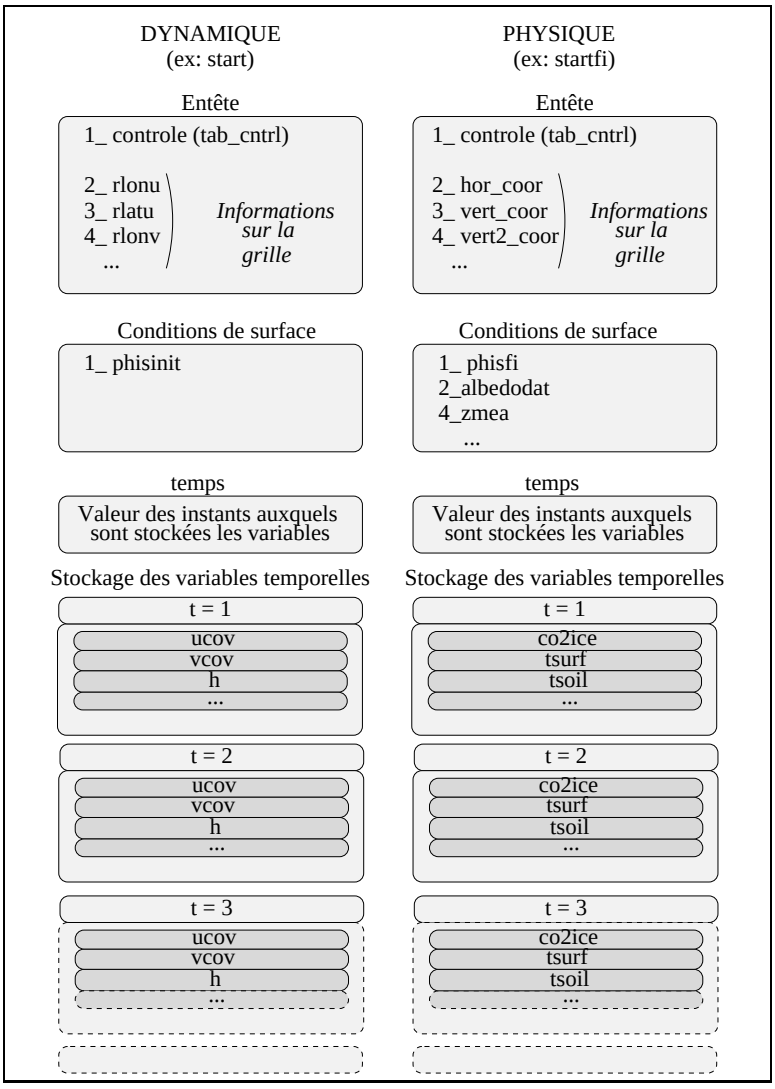

Figure 6.2: Organization of NetCDF files

Files `start.nc` and `startfi.nc`, like all the NetCDF files of the GCM, are constructed on the same model (see NetCDF file composition, figure 6.2). They contain:
- a header with a “control” variable followed by a series of variables defining the (physical and dynamical) grids
- a series of non temporal variables that give information about surface conditions on the planet.
- a “time” variable giving the values of the different instants at which the temporal variables are stored (a single time value (t=0) for start, as it describes the dynamical initial

states, and no time values for startfi, as it describes only a physical state).

To visualize the contents of a `start.nc` file using the `ncdump` command:

ncdump -h start.nc

```
netcdf start {
dimensions:
        index = 100 ;
        rlonu = 33 ;
        latitude = 25 ;
        longitude = 33 ;
        rlatv = 24 ;
        altitude = 18 ;
        interlayer = 19 ;
        Time = UNLIMITED ; // (1 currently)
variables:
        float controle(index) ;
                controle:title = "Parametres de controle" ;
        float rlonu(rlonu) ;
                rlonu:title = "Longitudes des points U" ;
        float rlatu(latitude) ;
                rlatu:title = "Latitudes des points U" ;
        float rlonv(longitude) ;
                rlonv:title = "Longitudes des points V" ;
        float rlatv(rlatv) ;
                rlatv:title = "Latitudes des points V" ;
        float ap(interlayer) ;
                ap:title = "Coef A: hybrid pressure levels" ;
        float bp(interlayer) ;
                bp:title = "Coef B: hybrid sigma levels" ;
        float aps(altitude) ;
                aps:title = "Coef AS: hybrid pressure at midlayers" ;
        float bps(altitude) ;
                bps:title = "Coef BS: hybrid sigma at midlayers" ;
        float presnivs(altitude) ;
        float latitude(latitude) ;
                latitude:units = "degrees_north" ;
                latitude:long_name = "North latitude" ;
        float longitude(longitude) ;
                longitude:long_name = "East longitude" ;
                longitude:units = "degrees_east" ;
        float altitude(altitude) ;
                altitude:long_name = "pseudo-alt" ;
                altitude:units = "km" ;
                altitude:positive = "up" ;
        float cu(latitude, rlonu) ;
                cu:title = "Coefficient de passage pour U" ;
        float cv(rlatv, longitude) ;
                cv:title = "Coefficient de passage pour V" ;
        float aire(latitude, longitude) ;
                aire:title = "Aires de chaque maille" ;
        float phisinit(latitude, longitude) ;
                phisinit:title = "Geopotentiel au sol" ;
        float Time(Time) ;
                Time:title = "Temps de simulation" ;
                Time:units = "days since    1-01-01 00:00:00" ;
        float ucov(Time, altitude, latitude, rlonu) ;
                ucov:title = "Vitesse U" ;
        float vcov(Time, altitude, rlatv, longitude) ;
                vcov:title = "Vitesse V" ;
        float teta(Time, altitude, latitude, longitude) ;
                teta:title = "Temperature" ;
        float h2o_ice(Time, altitude, latitude, longitude) ;
                h2o_ice:title = "Traceur h2o_ice" ;
        float h2o_vap(Time, altitude, latitude, longitude) ;
```

```
                h2o_vap:title = "Traceur h2o_vap" ;
        float masse(Time, altitude, latitude, longitude) ;
                masse:title = "C est quoi ?" ;
        float ps(Time, latitude, longitude) ;
                ps:title = "Pression au sol" ;

// global attributes:
                :title = "Dynamic start file" ;
}
```

List of contents of a `startfi.nc` file:

ncdump -h startfi.nc

```
netcdf startfi {
dimensions:
        index = 100 ;
        physical_points = 738 ;
        subsurface_layers = 18 ;
        nlayer_plus_1 = 19 ;
        number_of_advected_fields = 3 ;
variables:
        float controle(index) ;
                controle:title = "Control parameters" ;
        float soildepth(subsurface_layers) ;
                soildepth:title = "Soil mid-layer depth" ;
        float longitude(physical_points) ;
                longitude:title = "Longitudes of physics grid" ;
        float latitude(physical_points) ;
                latitude:title = "Latitudes of physics grid" ;
        float area(physical_points) ;
                area:title = "Mesh area" ;
        float phisfi(physical_points) ;
                phisfi:title = "Geopotential at the surface" ;
        float albedodat(physical_points) ;
                albedodat:title = "Albedo of bare ground" ;
        float ZMEA(physical_points) ;
                ZMEA:title = "Relief: mean relief" ;
        float ZSTD(physical_points) ;
                ZSTD:title = "Relief: standard deviation" ;
        float ZSIG(physical_points) ;
                ZSIG:title = "Relief: sigma parameter" ;
        float ZGAM(physical_points) ;
                ZGAM:title = "Relief: gamma parameter" ;
        float ZTHE(physical_points) ;
                ZTHE:title = "Relief: theta parameter" ;
        float co2ice(physical_points) ;
                co2_ice:title = "CO2 ice cover" ;
        float inertiedat(subsurface_layers, physical_points) ;
                inertiedat:title = "Soil thermal inertia" ;
        float tsurf(physical_points) ;
                tsurf:title = "Surface temperature" ;
        float tsoil(subsurface_layers, physical_points) ;
                tsoil:title = "Soil temperature" ;
        float emis(physical_points) ;
                emis:title = "Surface emissivity" ;
        float q2(nlayer_plus_1, physical_points) ;
                q2:title = "pbl wind variance" ;
        float h2o_ice(physical_points) ;
                h2o_ice:title = "tracer on surface" ;

// global attributes:
                :title = "Physics start file" ;
}
```

Physical and dynamical headers

There are two types of headers: one for the physical headers, and one for the dynamical headers. The headers always begin with a "control' variable (described below), that is allocated differently in the physical and dynamical parts. The other variables in the header concern the (physical and dynamical) grids. They are the following:

the horizontal coordinates
- **rlonu**, **rlatu**, **rlonv**, **rlatv** for the dynamical part,
- **lati**, **long** for the physical part,

the coefficients for passing from the physical grid to the dynamical grid
- **cu**,**cv** only in the dynamical header

and finally, the grid box areas
- **aire** for the dynamical part,
- **area** for the physical part.

Surface conditions

The surface conditions are mostly given in the physical NetCDF files by variables:
- **phisfi** for the initial state of surface geopotential,
- **albedodat** for the bare ground albedo,
- **inertiedat** for the surface thermal inertia,
- **zmea**, **zstd**, **zsig**, **zgam** and **zthe** for the subgrid scale topography.

For the dynamics:
- **physinit** for the initial state of surface geopotential

Remark: variables **phisfi** and **physinit** contain the same information (surface geopotential), but **phisfi** gives the geopotential values on the physical grid, while **physinit** give the values on the dynamical grid.

Physical and dynamical state variables

To save disk space, the initialization files store the variables used by the model, rather than the "natural" variables.

For the dynamics:

- **ucov** and **vcov** the covariant winds
  These variables are linked to the "natural" winds by
  `ucov = cu * u` and `vcov = cv * v`

- **teta** the potential temperature,
  or more precisely, the potential enthalpy linked to temperature **T** by $\theta = T\left(\frac{P}{Pref}\right)^{-K}$

- the tracers,

- **ps** surface pressure.

- **masse** the atmosphere mass in each grid box.

"Vectorial" variables **ucov** and **vcov** are stored on "staggered" grids u and v respectively (in the dynamics) (see section 2.2).
Scalar variables **h**, **q** (tracers), **ps**, **masse** are stored on the "scalar" grid of the dynamical part.

For the physics:

- **co2ice** surface dry ice,

- **tsurf** surface temperature,

- **tsoil** temperatures at different layers under the surface,

- **emis** surface emissivity,

- **q2** wind variance,
  or more precisely, the square root of the turbulent kinetic energy.

- the surface "tracer" budget (kg.m^{-2}),

All these variables are stored on the "physical" grid (see section 2.2).

The "control" array

Both physical and dynamical headers of the GCM NetCDF files start with a **controle** variable. This variable is an array of 100 reals (the vector called `tab_cntrl` in the program), which contains the program control parameters. Parameters differ between the physical and dynamical sections, and examples of both are listed below. The contents of table `tab_cntrl` can also be checked with the command `ncdump -ff -v controle`.

The "control" array in the header of a dynamical NetCDF file: start

```
tab_cntrl(1)  = FLOAT(iim) ! number of nodes along longitude
tab_cntrl(2)  = FLOAT(jjm) ! number of nodes along latitude
tab_cntrl(3)  = FLOAT(llm) ! number of atmospheric layers
tab_cntrl(4)  = FLOAT(idayref) ! initial day
tab_cntrl(5)  = rad   ! radius of the planet
tab_cntrl(6)  = omeg  ! rotation of the planet (rad/s)
tab_cntrl(7)  = g     ! gravity (m/s2) ~3.72 for Mars
tab_cntrl(8)  = cpp
tab_cntrl(9) = kappa   ! = r/cp
tab_cntrl(10) = daysec ! lenght of a sol (s) ~88775
tab_cntrl(11) = dtvr   ! dynamical time step (s)
tab_cntrl(12) = etot0  ! total energy
tab_cntrl(13) = ptot0  ! total pressure
tab_cntrl(14) = ztot0  ! total enstrophy
tab_cntrl(15) = stot0  ! total enthalpy
tab_cntrl(16) = ang0   ! total angular momentum
tab_cntrl(17) = pa
tab_cntrl(18) = preff  ! reference pressure (Pa)
tab_cntrl(19)  = clon  ! longitude of center of zoom
tab_cntrl(20)  = clat  ! latitude of center of zoom
tab_cntrl(21)  = grossismx ! zooming factor, along longitude
tab_cntrl(22)  = grossismy ! zooming factor, along latitude

tab_cntrl(24) = dzoomx ! extention (in longitude) of zoom
tab_cntrl(25) = dzoomy ! extention (in latitude) of zoom

tab_cntrl(27) = taux ! stiffness factor of zoom in longitude
tab_cntrl(28) = tauy ! stiffness factor of zoom in latitude
```

The "controle" array in the header of a physical NetCDF file: startfi.nc

```
c Informations on the physics grid
      tab_cntrl(1) = float(ngridmx)  ! number of nodes on physics grid
      tab_cntrl(2) = float(nlayermx) ! number of atmospheric layers
      tab_cntrl(3) = day_ini + int(time)         ! initial day
      tab_cntrl(4) = time -int(time)           ! initiale time of day

c Informations about Mars, used by dynamics and physics
      tab_cntrl(5) = rad      ! radius of Mars (m) ~3397200
      tab_cntrl(6) = omeg     ! rotation rate (rad.s-1)
      tab_cntrl(7) = g        ! gravity (m.s-2) ~3.72
      tab_cntrl(8) = mugaz    ! Molar mass of the atmosphere (g.mol-1) ~43.49
      tab_cntrl(9) = rcp      !  = r/cp  ~0.256793 (=kappa dans dynamique)
      tab_cntrl(10) = daysec  ! length of a sol (s)  ~88775

      tab_cntrl(11) = phystep  ! time step in the physics
      tab_cntrl(12) = 0.
      tab_cntrl(13) = 0.

c Informations about Mars, only for physics
      tab_cntrl(14) = year_day  ! length of year (sols) ~668.6
      tab_cntrl(15) = periheli  ! min. Sun-Mars distance (Mkm) ~206.66
      tab_cntrl(16) = aphelie   ! max. SUn-Mars distance (Mkm) ~249.22
      tab_cntrl(17) = peri_day  ! date of perihelion (sols since N. spring)
      tab_cntrl(18) = obliquit  ! Obliquity of the planet (deg) ~23.98

c Boundary layer and turbulence
      tab_cntrl(19) = z0        ! surface roughness (m) ~0.01
      tab_cntrl(20) = lmixmin   ! mixing length ~100
      tab_cntrl(21) = emin_turb ! minimal energy ~1.e-8

c Optical properties of polar caps and ground emissivity
      tab_cntrl(22) = albedice(1)  ! Albedo of northern cap ~0.5
      tab_cntrl(23) = albedice(2)  ! Albedo of southern cap ~0.5
      tab_cntrl(24) = emisice(1)   ! Emissivity of northern cap ~0.95
      tab_cntrl(25) = emisice(2)   ! Emissivity of southern cap ~0.95
      tab_cntrl(26) = emissiv      ! Emissivity of martian soil ~.95
      tab_cntrl(31) = iceradius(1) ! mean scat radius of CO2 snow (north)
      tab_cntrl(32) = iceradius(2) ! mean scat radius of CO2 snow (south)
      tab_cntrl(33) = dtemisice(1) ! time scale for snow metamorphism (north)
      tab_cntrl(34) = dtemisice(2) ! time scale for snow metamorphism (south)

c dust aerosol properties
      tab_cntrl(27) = tauvis      ! mean visible optical depth

      tab_cntrl(28) = 0.
      tab_cntrl(29) = 0.
      tab_cntrl(30) = 0.

! Soil properties:
      tab_cntrl(35) = volcapa ! soil volumetric heat capacity
```

6.3 Output files

6.3.1 NetCDF restart files - restart.nc and restartfi.nc

These files are of the exact same format as `start.nc` and `startfi.nc`

6.3.2 NetCDF file - diagfi.nc

NetCDF file `diagfi.nc` stores the instantaneous physical variables throughout the simulation at regular intervals (set by the value of parameter `ecritphy` in parameter file `run.def`; note that `ecritphy` should be a multiple of `iphysiq` as well as a divisor of `day_step`).

Any variable from any sub-routine of the physics can be stored by calling subroutine `writediagfi`

Illustrative example of the contents of a `diagfi.nc` file (using ncdump):
ncdump -h diagfi.nc

```
netcdf diagfi {
dimensions:
        Time = UNLIMITED ; // (12 currently)
        index = 100 ;
        rlonu = 65 ;
        latitude = 49 ;
        longitude = 65 ;
        rlatv = 48 ;
        interlayer = 26 ;
        altitude = 25 ;
        subsurface_layers = 18 ;
variables:
        float Time(Time) ;
                Time:long_name = "Time" ;
                Time:units = "days since 0000-00-0 00:00:00" ;
        float controle(index) ;
                controle:title = "Control parameters" ;
        float rlonu(rlonu) ;
                rlonu:title = "Longitudes at u nodes" ;
        float latitude(latitude) ;
                latitude:units = "degrees_north" ;
                latitude:long_name = "North latitude" ;
        float longitude(longitude) ;
                longitude:long_name = "East longitude" ;
                longitude:units = "degrees_east" ;
        float altitude(altitude) ;
                altitude:long_name = "pseudo-alt" ;
                altitude:units = "km" ;
                altitude:positive = "up" ;
        float rlatv(rlatv) ;
                rlatv:title = "Latitudes at v nodes" ;
        float aps(altitude) ;
                aps:title = "hybrid pressure at midlayers" ;
                aps:units = "Pa" ;
        float bps(altitude) ;
                bps:title = "hybrid sigma at midlayers" ;
                bps:units = "" ;
        float ap(interlayer) ;
                ap:title = "hybrid pressure at interlayers" ;
                ap:units = "Pa" ;
        float bp(interlayer) ;
                bp:title = "hybrid sigma at interlayers" ;
                bp:units = "" ;
        float soildepth(subsurface_layers) ;
                soildepth:long_name = "Soil mid-layer depth" ;
                soildepth:units = "m" ;
                soildepth:positive = "down" ;
```

```
        float cu(latitude, rlonu) ;
                cu:title = "Conversion coefficients cov <--> natural" ;
        float cv(rlatv, longitude) ;
                cv:title = "Conversion coefficients cov <--> natural" ;
        float aire(latitude, longitude) ;
                aire:title = "Mesh area" ;
        float phisinit(latitude, longitude) ;
                phisinit:title = "Geopotential at the surface" ;
        float emis(Time, latitude, longitude) ;
                emis:title = "Surface emissivity" ;
                emis:units = "w.m-1" ;
        float tsurf(Time, latitude, longitude) ;
                tsurf:title = "Surface temperature" ;
                tsurf:units = "K" ;
        float ps(Time, latitude, longitude) ;
                ps:title = "surface pressure" ;
                ps:units = "Pa" ;
        float co2ice(Time, latitude, longitude) ;
                co2ice:title = "co2 ice thickness" ;
                co2ice:units = "kg.m-2" ;
        float mtot(Time, latitude, longitude) ;
                mtot:title = "total mass of water vapor" ;
                mtot:units = "kg/m2" ;
        float icetot(Time, latitude, longitude) ;
                icetot:title = "total mass of water ice" ;
                icetot:units = "kg/m2" ;
        float tauTES(Time, latitude, longitude) ;
                tauTES:title = "tau abs 825 cm-1" ;
                tauTES:units = "" ;
        float h2o_ice_s(Time, latitude, longitude) ;
                h2o_ice_s:title = "surface h2o_ice" ;
                h2o_ice_s:units = "kg.m-2" ;
}
```

The structure of the file is thus as follows:

- the dimensions
- variable “time” containing the time of the timestep stored in the file (in Martian days since the beginning of the run)
- variable “control” containing many parameters, as described above.
- from “ rhonu” to ’phisinit”: a list of data describing the geometrical coordinates of the data file, plus the surface topography
- finally, all the 2D or 3D data stored in the run.

6.3.3 Stats files

As an option (`stats` must be set to `.true.` in `callphys.def`), the model can accumulate any variable from any subroutine of the physics by calling subroutine `wstat`

This save is performed at regular intervals 12 times a day. An average of the daily evolutions over the whole run is calculated (for example, for a 10 day run, the averages of the variable values at 0hTU, 2hTU, 4hTU,...24hTU are calculated), along with RMS standard deviations of the variables. This ouput is given in file `stats.nc`.

Illustrative example of the contents of a `stats.nc` file (using ncdump):
ncdump -h stats.nc

```
netcdf stats {
dimensions:
```

```
        latitude = 49 ;
        longitude = 65 ;
        altitude = 25 ;
        llmp1 = 26 ;
        Time = UNLIMITED ; // (12 currently)
variables:
        float Time(Time) ;
                Time:title = "Time" ;
                Time:units = "days since 0000-00-0 00:00:00" ;
        float latitude(latitude) ;
                latitude:title = "latitude" ;
                latitude:units = "degrees_north" ;
        float longitude(longitude) ;
                longitude:title = "East longitude" ;
                longitude:units = "degrees_east" ;
        float altitude(altitude) ;
                altitude:long_name = "altitude" ;
                altitude:units = "km" ;
                altitude:positive = "up" ;
        float aps(altitude) ;
                aps:title = "hybrid pressure at midlayers" ;
                aps:units = "" ;
        float bps(altitude) ;
                bps:title = "hybrid sigma at midlayers" ;
                bps:units = "" ;
        float ps(Time, latitude, longitude) ;
                ps:title = "Surface pressure" ;
                ps:units = "Pa" ;
        float ps_sd(Time, latitude, longitude) ;
                ps_sd:title = "Surface pressure total standard deviation over th
e season" ;
                ps_sd:units = "Pa" ;
        float tsurf(Time, latitude, longitude) ;
                tsurf:title = "Surface temperature" ;
                tsurf:units = "K" ;
        float tsurf_sd(Time, latitude, longitude) ;
                tsurf_sd:title = "Surface temperature total standard deviation o
ver the season" ;
                tsurf_sd:units = "K" ;
        float co2ice(Time, latitude, longitude) ;
                co2ice:title = "CO2 ice cover" ;
                co2ice:units = "kg.m-2" ;
        float co2ice_sd(Time, latitude, longitude) ;
                co2ice_sd:title = "CO2 ice cover total standard deviation over t
he season" ;
                co2ice_sd:units = "kg.m-2" ;
        float fluxsurf_lw(Time, latitude, longitude) ;
                fluxsurf_lw:title = "Thermal IR radiative flux to surface" ;
                fluxsurf_lw:units = "W.m-2" ;
        float fluxsurf_lw_sd(Time, latitude, longitude) ;
                fluxsurf_lw_sd:title = "Thermal IR radiative flux to surface tot
al standard deviation over the season" ;
                fluxsurf_lw_sd:units = "W.m-2" ;
        float fluxsurf_sw(Time, latitude, longitude) ;
                fluxsurf_sw:title = "Solar radiative flux to surface" ;
                fluxsurf_sw:units = "W.m-2" ;
        float fluxsurf_sw_sd(Time, latitude, longitude) ;
                fluxsurf_sw_sd:title = "Solar radiative flux to surface total st
andard deviation over the season" ;
                fluxsurf_sw_sd:units = "W.m-2" ;
        float fluxtop_lw(Time, latitude, longitude) ;
                fluxtop_lw:title = "Thermal IR radiative flux to space" ;
                fluxtop_lw:units = "W.m-2" ;
        float fluxtop_lw_sd(Time, latitude, longitude) ;
                fluxtop_lw_sd:title = "Thermal IR radiative flux to space total
standard deviation over the season" ;
                fluxtop_lw_sd:units = "W.m-2" ;
```

```
        float fluxtop_sw(Time, latitude, longitude) ;
                fluxtop_sw:title = "Solar radiative flux to space" ;
                fluxtop_sw:units = "W.m-2" ;
        float fluxtop_sw_sd(Time, latitude, longitude) ;
                fluxtop_sw_sd:title = "Solar radiative flux to space total stand
ard deviation over the season" ;
                fluxtop_sw_sd:units = "W.m-2" ;
        float dod(Time, latitude, longitude) ;
                dod:title = "Dust optical depth" ;
                dod:units = "" ;
        float dod_sd(Time, latitude, longitude) ;
                dod_sd:title = "Dust optical depth total standard deviation over
 the season" ;
                dod_sd:units = "" ;
        float temp(Time, altitude, latitude, longitude) ;
                temp:title = "Atmospheric temperature" ;
                temp:units = "K" ;
        float temp_sd(Time, altitude, latitude, longitude) ;
                temp_sd:title = "Atmospheric temperature total standard deviatio
n over the season" ;
                temp_sd:units = "K" ;
        float u(Time, altitude, latitude, longitude) ;
                u:title = "Zonal (East-West) wind" ;
                u:units = "m.s-1" ;
        float u_sd(Time, altitude, latitude, longitude) ;
                u_sd:title = "Zonal (East-West) wind total standard deviation ov
er the season" ;
                u_sd:units = "m.s-1" ;
        float v(Time, altitude, latitude, longitude) ;
                v:title = "Meridional (North-South) wind" ;
                v:units = "m.s-1" ;
        float v_sd(Time, altitude, latitude, longitude) ;
                v_sd:title = "Meridional (North-South) wind total standard devia
tion over the season" ;
                v_sd:units = "m.s-1" ;
        float w(Time, altitude, latitude, longitude) ;
                w:title = "Vertical (down-up) wind" ;
                w:units = "m.s-1" ;
        float w_sd(Time, altitude, latitude, longitude) ;
                w_sd:title = "Vertical (down-up) wind total standard deviation o
ver the season" ;
                w_sd:units = "m.s-1" ;
        float rho(Time, altitude, latitude, longitude) ;
                rho:title = "Atmospheric density" ;
                rho:units = "none" ;
        float rho_sd(Time, altitude, latitude, longitude) ;
                rho_sd:title = "Atmospheric density total standard deviation ove
r the season" ;
                rho_sd:units = "none" ;
        float q2(Time, altitude, latitude, longitude) ;
                q2:title = "Boundary layer eddy kinetic energy" ;
                q2:units = "m2.s-2" ;
        float q2_sd(Time, altitude, latitude, longitude) ;
                q2_sd:title = "Boundary layer eddy kinetic energy total standard
 deviation over the season" ;
                q2_sd:units = "m2.s-2" ;
        float vmr_h2ovapor(Time, altitude, latitude, longitude) ;
                vmr_h2ovapor:title = "H2O vapor volume mixing ratio" ;
                vmr_h2ovapor:units = "mol/mol" ;
        float vmr_h2ovapor_sd(Time, altitude, latitude, longitude) ;
                vmr_h2ovapor_sd:title = "H2O vapor volume mixing ratio total sta
ndard deviation over the season" ;
                vmr_h2ovapor_sd:units = "mol/mol" ;
        float vmr_h2oice(Time, altitude, latitude, longitude) ;
                vmr_h2oice:title = "H2O ice volume mixing ratio" ;
                vmr_h2oice:units = "mol/mol" ;
        float vmr_h2oice_sd(Time, altitude, latitude, longitude) ;
```

```
                vmr_h2oice_sd:title = "H2O ice volume mixing ratio total standar
d deviation over the season" ;
                vmr_h2oice_sd:units = "mol/mol" ;
        float mtot(Time, latitude, longitude) ;
                mtot:title = "total mass of water vapor" ;
                mtot:units = "kg/m2" ;
        float mtot_sd(Time, latitude, longitude) ;
                mtot_sd:title = "total mass of water vapor total standard deviat
ion over the season" ;
                mtot_sd:units = "kg/m2" ;
        float icetot(Time, latitude, longitude) ;
                icetot:title = "total mass of water ice" ;
                icetot:units = "kg/m2" ;
        float icetot_sd(Time, latitude, longitude) ;
                icetot_sd:title = "total mass of water ice total standard deviat
ion over the season" ;
                icetot_sd:units = "kg/m2" ;
}
```

The structure of the file is simillar to the `diagfi.nc` file, except that, as stated before, the average of variables are given for 12 times of the day and that RMS standard deviation are also provided.

Chapter 7

Zoomed simulations

The LMD GCM can use a zoom to enhance the resolution locally. In practice, one can increase the latitudinal resolution on the one hand, and the longitudinal resolution on the other hand.

7.1 To define the zoomed area

The zoom is defined in `run.def`. Here are the variables that you want to set:

- East longitude (in degrees) of zoom center `clon`
- latitude (in degrees) of zoom center `clat`
- zooming factors, along longitude `grossismx`. *Typically 1.5, 2 or even 3 (see below)*
- zooming factors, along latitude `grossismy`. *Typically 1.5, 2 or even 3 (see below)*
- `fxyhypb`: **must be set to "T" for a zoom**, *whereas it must be F otherwise*
- extention in longitude of zoomed area `dzoomx`. This is the total longitudinal extension of the zoomed region (degree).
  It is recommended that `grossismx` $\times$ `dzoomx` $< 200^o$
- extention in latitude of the zoomed region `dzoomy`. This is the total latitudinal extension of the zoomed region (degree).
  It is recommended that `grossismy` $\times$ `dzoomy` $< 100^o$
- stiffness of the zoom along longitudes `taux`. 2 is for a smooth transition in longitude, more means sharper transition.
- stiffness of the zoom along latitudes `taux`. 2 is for a smooth transition in latitude, more means sharper transition.

7.2 Making a zoomed initial state

One must start from an initial state archive `start_archive.nc` obtained from a previous simulation (see section 4.8) Then compile and run `newstart.e` **using the** `run.def` **file designed for the zoom**.

After running `newstart.e`. The zoomed grid may be visualized using grads, for instance. Here is a grads script that can be used to map the grid above a topography map:

```
set mpdraw off
set grid off
sdfopen restart.nc
set gxout grid
set digsiz 0
set lon -180 180
d ps
close 1
*** replace the path to surface.nc in the following line:
sdfopen  /u/forget/WWW/datagcm/datafile/surface.nc
set lon -180 180
set gxout contour
set clab off
set cint 3
d zMOL
```

7.3 Running a zoomed simulation and stability issue

- **dynamical timestep** Because of their higher resolution, zoomed simulation requires a higher timestep. Therefore in `run.def`, the number of dynamical timestep per day `day_step` must be increased by more than `grossismx` or `grossismy` (twice that if necessary). However, you can keep the same physical timestep (48/sol) and thus increase `iphysiq` accordingly (`iphysiq = day_step/48`).

- It has been found that when zooming in longitude, on must set `ngroup=1` in `dyn3d/groupeun.F`. Otherwise the run is less stable.

- The very first initial state made with `newstart.e` can be noisy and dynamically unstable. It may be necessary to strongly increase the intensity of the dissipation and increase `day_step` in `run.def` for 1 to 3 sols, and then use less strict values.

- If the run remains very unstable and requires too much dissipation or a too small timestep, a good tip to help stabilize the model is to decrease the vertical extension of your run and the number of layer (one generally zoom to study near-surface process, so 20 to 22 layers and a vertical extension up to 60 or 80 km is usually enough).

Chapter 8

Water Cycle Simulation

In order to simulate the water cycle with the LMD GCM:

- In `callphys.def`, set tracer to true: `tracer=.true.`. Use the same options as below for the Tracer part, the rest does not change compared to the basic `callphys.def`. The important parameters are `water=.true.`, to use water vapor and ice tracers, and `sedimentation=.true.` to allow sedimentation of water ice clouds.

```
## Tracer (dust water, ice and/or chemical species) options (used if tracer=T):
## ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# DUST: Transported dust ? (if >0, use 'dustbin' dust bins)
dustbin    = 0
# DUST: Radiatively active dust ? (matters if dustbin>0)
active  = .false.
# DUST: use mass and number mixing ratios to predict dust size ?
# (must also have dustbin=1)
doubleq   = .false.
# DUST: lifted by GCM surface winds ?
lifting = .false.
# DUST: lifted by dust devils ?
callddevil = .false.
# DUST: Scavenging by CO2 snowfall ?
scavenging = .false.
# DUST/WATERICE: Gravitationnal sedimentation ?
sedimentation = .true.
# WATERICE: Radiatively active transported atmospheric water ice ?
activice   = .false.
# WATER: Compute water cycle
water = .true.
# WATER: current permanent caps at both poles. True IS RECOMMENDED
#        (with .true., North cap is a source of water and South pole
#         is a cold trap)
caps  = .true.
```

- **Compilation** You need to compile with at least 2 tracers. If you don't have dust (`dustbin=0`) or other chemical species (`photochem=F`), compilation is done with the command lines:

  ```
  makegcm -d 64x48x25 -t 2 -p mars newstart

  makegcm -d 64x48x25 -t 2 -p mars gcm
  ```

  Of course, you will also need an appropriate `traceur.def` file indicating you will use tracers `h2o_vap` and `h2o_ice`; if you only run with 2 tracers, then the contents of the `traceeur.def` file should be:

```
2
h2o_ice
h2o_vap
```

Note that the order in which tracers are set in the `tracer.def` file is not important.

- **Run**

  Same as usual. Just make sure that your start files contains the initial states for water, with an initial state for water vapor and water ice particles.

Chapter 9

Photochemical Module

The LMD GCM now includes a photochemical module, which allows to compute the atmospheric composition.

- 14 chemical species are included: CO_2 (background gas), CO, O, O(^1D), O_2, O_3, H, H_2, OH, HO_2, H_2O_2, N_2, Ar (inert) and H_2O.
- In `callphys.def`, set tracer to true `tracer=.true.`. Use the same options as shown below for the tracer part of `callphys.def`. You need to set `photochem=.true.`, and to include the water cycle (`water=.true.`, `sedimentation=.true.`; see Chapter 8), because composition is extremely dependent on the water vapor abundance.

```
## Tracer (dust water, ice and/or chemical species) options (used if tracer=T):
## ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# DUST: Transported dust ? (if >0, use 'dustbin' dust bins)
dustbin    = 0
# DUST: Radiatively active dust ? (matters if dustbin>0)
active  = .false.
# DUST: use mass and number mixing ratios to predict dust size ?
# (must also have dustbin=1)
doubleq   = .false.
# DUST: lifted by GCM surface winds ?
lifting = .false.
# DUST: lifted by dust devils ?
callddevil = .false.
# DUST: Scavenging by CO2 snowfall ?
scavenging = .false.
# DUST/WATERICE: Gravitationnal sedimentation ?
sedimentation = .true.
# WATERICE: Radiatively active transported atmospheric water ice ?
activice   = .false.
# WATER: Compute water cycle
water = .true.
# WATER: current permanent caps at both poles. True IS RECOMMENDED
#        (with .true., North cap is a source of water and South pole
#         is a cold trap)
caps  = .true.
# PHOTOCHEMISTRY: include chemical species
photochem  = .true.
```

- You will need the up-to-date file `jmars.yyyymmdd` (e.g. `jmars.20030707`), which contains the photodissociation rates. It should be in the *datafile* directory in which are stored datafiles used by the GCM (the path to these files is set in file `datafile.h`, in the `phymars` directory).
- **Compilation**

You need to compile with 15 tracers (if you don't have dust, `dustbin=0`): 13 chemical species (co2, co, o, o(1d), o2, o3, h, h2, oh, ho2, h2o2, n2, ar) along with water ice (h2o_ice) and water vapor (h2o_vap).
Compilation is done with the command lines:

```
makegcm -d 64x48x25 -t 15 -p mars newstart

makegcm -d 64x48x25 -t 15 -p mars gcm
```

Of course, the `traceur.def` file should contain the number and name of all the tracers, e.g.:

```
15
co2
co
o
o1d
o2
o3
h
h2
oh
ho2
h2o2
n2
ar
h2o_ice
h2o_vap
```

- **Run**

  Same as usual. Just make sure that your start files contains the correct number of tracers. If you need to initialize the composition, you can run **newstart** and use the options

  - ini_q: the 15 tracers are initialized, including water ice and vapor.
  - ini_q-h2o: the 13 chemical species are initialized, water ice is put to zero, and water vapor is kept untouched.
  - ini_q-iceh2o: the 13 chemical species are initialized, water ice and vapor are kept untouched.

  The initialization is done with the files `atmosfera_LMD_may.dat` and `atmosfera_LMD_min.dat`, which should also be found in the *datafile* directory.

- **Outputs**

  The outputs can be done from the `aeronomars/calchim.F` routine for the 14 chemical species. The variables put in the `diagfi.nc` and `stats.nc` files are labeled (where *name* is the name of the chemical species, e.g. co2):

  - n_*name*: local density (in molecule cm^{-3}, 3-dimensional field)
  - c_*name*: integrated column density (in molecule cm^{-2}, 2-dimensional field)

Chapter 10

1D version of the Mars model

The physical part of the model can be used to generate realistic 1-D simulations (one atmosphere column). In practice, the simulation is controlled from a main program called `testphys1d.F` which, after initialization, then calls the master subroutine of the physics `physiq.F` described in the preceeding chapters.

10.1 Compilation

- For example, to compile the Martian model in 1-D with 25 layers, type (in compliance with the makegcm function manual described in section 5.4)

```
makegcm -d 25 -p mars testphys1d
```

You can find executable **testphys1d.e** (the compiled model) in the directory from which you ran the makegcm command.

10.2 1-D runs and input files

The 1-D model does not use an initial state file (the simulation must be long enough to obtain a balanced state). Thus, to generate a simulation simply type:

```
> testphys1d.e
```

The following example files are available in the `deftank` directory (copy them into your working directory first):

- **callphys.def** : controls the options in the physics, just like for the 3D GCM.
- **z2sig.def** : controls the vertical discretization (no change needed, in general), functions as with the 3D GCM.
- **traceur.def** : controls the tracer names (this file may not be present, as long as you run without tracers (option `tracer=.false.` in callphys.def)
- **run.def** : controls the 1-D run parameters and initializations (this is actally file `run.def.1d` the `deftank` directory, which must be renamed `run.def` to be read by the program).

The last file is different from the 3D GCM's `run.def` input file, as it contains options specific to the 1-D model, as shown in the example below:

```
#
#-------------------------------------------------------------------------
# Run parameters for the 1D 'testphys1d.e' model
#-------------------------------------------------------------------------
```

```
#### Time integration parameters
#
# Initial date (in martian sols ; =0 at Ls=0)
day0=0
# Initial local time (in hours, between 0 and 24)
time=0
# Number of time steps per sol
day_step=48
# Number of sols to run
ndt = 100

#### Physical parameters
#
# Surface pressure (Pa)
psurf= 610
# Reference dust opacity at 700 Pa, in the visible (true tau~tauref*psurf/700)
tauvis=0.2
# latitude (in degrees)
latitude= 0.
# Albedo of bare ground
albedo=0.2
# Soil thermal inertia (SI)
inertia=400
# zonal eastward component of the geostrophic wind (m/s)
u=10.
# meridional northward component of the geostrophic wind (m/s)
v=0.
# Initial CO2 ice on the surface (kg.m-2)
co2ice=0
# hybrid vertical coordinate ? (.true. for hybrid and .false. for sigma levels)
hybrid=.true.


###### Initial atmospheric temperature profile
#
# Type of initial temperature profile
#         ichoice=1   Constant Temperature:  T=tref
#         ichoice=2   Savidjari profile (as Seiff but with dT/dz=cte)
#         ichoice=3   Lindner (polar profile)
#         ichoice=4   inversion
#         ichoice=5   Seiff  (standard profile, based on Viking entry)
#         ichoice=6   constant T  +  gaussian perturbation (levels)
#         ichoice=7   constant T  + gaussian perturbation (km)
#         ichoice=8   Read in an ascii file "profile"
ichoice=5
# Reference temperature tref (K)
tref=200
# Add a perturbation to profile if isin=1
isin=0
# peak of gaussian perturbation (for ichoice=6 or 7)
pic=26.522
# width of the gaussian perturbation (for ichoice=6 or 7)
largeur=10
# height of the gaussian perturbation (for ichoice=6 or 7)
hauteur=30.


# some definitions for the physics, in file 'callphys.def'
INCLUDEDEF=callphys.def
```

Note that, just as for the 3-D GCM `run.def` file, input parameters may be given in any order, or even not given at all (in which case default values are used by the program).

10.3 Output data

During the entire 1D simulation, you can obtain output data for any variable from any physical subroutine by using subroutine `writeg1d`. This subroutine creates file `g1d.nc` that can be read by GRADS. This subroutine is typically called at the end of subroutine `physiq` .

Example of a call to subroutine `writeg1d` requesting temperature output: (`ngrid` horizontal point, `nlayer` layers, variable `pt` called “T” in K units):

```
CALL writeg1d(ngrid,nlayer,pt,'T','K')
```

Appendix A

GCM Martian Calendar

For Mars, dates and seasons are expressed in Solar Longitude (L_s, in degrees or in radians) counting from the northern hemisphere spring equinox. In the GCM, time is counted in Martian solar days, or "sols" (1 sols = 88775 s) from the northern spring equinox. The following table gives the correspondence between sols and L_s, calculated for the GCM using one Martian year = 669 sols exactly.

sol	L_s		sol	L_s		sol	L_s	
0.	360.000	Spring equinox N	240.	111.455		480.	247.408	
5.	2.550		245.	113.816		485.	250.666	
10.	5.080		250.	116.190		490.	253.925	
15.	7.590		255.	118.578		495.	257.182	
20.	10.081		260.	120.981		500.	260.435	
25.	12.554		265.	123.400		505.	263.683	
30.	15.009		270.	125.835		510.	266.924	
						514.76	270.	Winter solstice N
35.	17.447		275.	128.287		515.	270.156	
40.	19.869		280.	130.756		520.	273.377	
45.	22.275		285.	133.243		525.	276.587	
50.	24.666		290.	135.750		530.	279.783	
55.	27.043		295.	138.275		535.	282.965	
60.	29.407		300.	140.821		540.	286.130	
65.	31.758		305.	143.388		545.	289.277	
70.	34.096		310.	145.975		550.	292.406	
75.	36.423		315.	148.585		555.	295.515	
80.	38.739		320.	151.217		560.	298.604	
85.	41.046		325.	153.872		565.	301.671	
90.	43.343		330.	156.550		570.	304.715	
95.	45.631		335.	159.251		575.	307.737	
100.	47.912		340.	161.977		580.	310.735	
105.	50.186		345.	164.727		585.	313.709	
110.	52.453		350.	167.502		590.	316.658	
115.	54.714		355.	170.301		595.	319.583	
120.	56.970		360.	173.126		600.	322.483	
125.	59.222		365.	175.975		605.	325.358	
130.	61.471		370.	178.850		610.	328.207	
.			371.99	180.	Autumn equinox N	.		
135.	63.716		375.	181.750		615.	331.032	
140.	65.959		380.	184.675		620.	333.831	
145.	68.201		385.	187.624		625.	336.606	
150.	70.442		390.	190.598		630.	339.356	
155.	72.683		395.	193.596		635.	342.082	
160.	74.925		400.	196.618		640.	344.783	
165.	77.168		405.	199.662		645.	347.461	
170.	79.413		410.	202.729		650.	350.116	
175.	81.661		415.	205.818		655.	352.748	
180.	83.912		420.	208.927		660.	355.357	
185.	86.167		425.	212.056		665.	357.945	
						669.	0.	Spring equinox N
190.	88.427		430.	215.203		670.	0.512	
193.47	90.	Summer solstice N						
195.	90.693		435.	218.368		675.	3.057	
200.	92.965		440.	221.549		680.	5.583	
205.	95.245		445.	224.746		685.	8.089	
210.	97.532		450.	227.955		690.	10.577	
215.	99.827		455.	231.177		695.	13.046	
220.	102.131		460.	234.409		700.	15.498	
225.	104.446		465.	237.650		705.	17.933	
230.	106.770		470.	240.898		710.	20.351	
235.	109.107		475.	244.151		715.	22.755	
240.	111.455		480.	247.408		720.	25.143	

Appendix B

Utilities

A few post-processing tools, which handle GCM outputs (files `diagfi.nc` and `stats.nc`) are available on the web at:
`http://www.lmd.jussieu.fr/~forget/datagcm/Utilities/`
The directory contains compiled executables (`*.e` files) of the tools decribed below, along with some examples of input instruction (`*.def` files) and a `README`.
There is also a `SOURCES` directory which contains the (Fortran) sources of the codes, if you should need to recompile them on your platform.

B.1 concatnc

This program concatenates consecutive output files (`diagfi.nc` or even `stats.nc` files) for a selection of variable, in order to obtain one single big file. The time dimension of the output can be "sols" or "Ls" (note that in that latter case, Ls values won't be evenly distributed, and software like Grads may not be able to use and plot the data).
To obtain an evenly sampled "Ls" timescale, you can use the `lslin.e` program (described below).
The output file created by `conctanc.e` is `concat.nc`

B.2 lslin

This program is designed to interpolate data given in irregular Solar Longitude (Ls) into an evenly sampled linear time coordinate (usable with Grads). Input Netcdf files may be `diagfi.nc` or `concat.nc` files and the resulting output file is `lslin.nc` lslin also create a `lslin.ctl` file that can be read directly by grads (`>xdfopen lslin.ctl`) to plot in Ls coordinate to avoid some problem with grads when Grads think that "the time interval is too small"...

B.3 localtime

The `localtime.e` program is designed to re-interpolate data in order to yield values at the same given local time (useful to mimic satellite observations, or analyse day to day variations at given local time).
Input files may be of `diagfi.nc`, `stats.nc` or `concat.nc` type and the output file name is build from the input one, to which `_LT.nc` is appened (e.g. if the input file is `myfile.nc` then output file will be `myfile_LT.nc`).

B.4 zrecast

With this program you can recast atmospheric (i.e.: 4D-dimentional longitude-latitude-altitude-time) data from GCM outputs (e.g. as given in `diagfi.nc`, `concat.nc` and `stats.nc` files) onto either *pressure* or *altitude above areoid* vertical coordinates.
Since integrating the hydrostatic equation is required to recast the data, the input file must contain surface pressure and atmospheric temperature, as well as the ground geopotential.
If recasting data onto *pressure* coordinates, then the output file name is given by the input file name to which `_P.nc` will be appened. If recasting data onto *altitude above areoid* coordinates, then a `_A.nc` will be appened.

Bibliography

[1] R. T. Clancy and S. W. Lee. A new look at dust and clouds in the Mars atmosphere: Analysis of emission-phase function sequences from global Viking IRTM observations. *Icarus*, 93:135–158, 1991.

[2] J.-L. Dufresne, R. Fournier, C. Hourdin, and F. Hourdin. Net Exchange Reformulation of Radiative Transfer in the CO_2 15-μm Band on Mars. *Journal of Atmospheric Sciences*, 62:3303–3319, 2005.

[3] F. Forget. Improved optical properties of the Martian atmospheric dust for radiative transfer calculations in the infrared. *Geophys. Res. Lett.*, 25:1105–1109, 1998.

[4] F. Forget, F. Hourdin, R. Fournier, C. Hourdin, O. Talagrand, M. Collins, S. R. Lewis, P. L. Read, and J.-P. Huot. Improved general circulation models of the Martian atmosphere from the surface to above 80 km. *J. Geophys. Res.*, 104:24,155–24,176, 1999.

[5] F. Forget, F. Hourdin, and O. Talagrand. CO_2 snow fall on Mars: Simulation with a general circulation model. *Icarus*, 131:302–316, 1998.

[6] Y. Fouquart and B. Bonnel. Computations of solar heating of the Earth's atmosphere: A new parametrization. *Contrib. Atmos. Phys.*, 53:35–62, 1980.

[7] C. Hourdin, J.-L. Dufresnes, R. Fournier, and F. Hourdin. Net exchange reformulation of radiative transfer in the CO_2 15μm band on mars. Article in preparation, 2000.

[8] F. Hourdin. A new representation of the CO_2 15 μm band for a Martian general circulation model. *J. Geophys. Res.*, 97(E11):18,319–18,335, 1992.

[9] F. Hourdin and A. Armengaud. Test of a hierarchy of finite-volume schemes for transport of trace species in an atmospheric general circulation model. *Mon. Wea. Rev.*, 127:822–837, 1999.

[10] F. Hourdin, P. Le Van, F. Forget, and O. Talagrand. Meteorological variability and the annual surface pressure cycle on Mars. *J. Atmos. Sci.*, 50:3625–3640, 1993.

[11] S. Lefèvre, S. Lebonnois, F. Montmessin, and F. Forget. Three-dimensional modeling of ozone on Mars . *Journal of Geophysical Research (Planets)*, 109:E07004, 2004.

[12] S. R. Lewis, M. Collins, P. L. Read, F. Forget, F. Hourdin, R. Fournier, C. Hourdin, O. Talagrand, and J.-P. Huot. A climate database for Mars. *J. Geophys. Res.*, 104:24,177–24,194, 1999.

[13] F. Lott and M. Miller. A new sub-grid scale orographic drag parametrization: its formulation and testing. *Q. J. R. Meteorol. Soc.*, 123:101–128, 1997.

[14] F. Montmessin, F. Forget, P. Rannou, M. Cabane, and R. M. Haberle. Origin and role of water ice clouds in the Martian water cycle as inferred from a general circulation model. *Journal of Geophysical Research (Planets)*, 109(E18):10004–+, 2004.

[15] M. E. Ockert-Bell, J. F. Bell III, C. McKay, J. Pollack, and F. Forget. Absorption and scattering properties of the Martian dust in the solar wavelengths. *J. Geophys. Res.*, 102:9039–9050, 1997.

[16] N. O. Rennò, M. L. Burkett, and M. P. Larkin. A simple thermodynamical theory for dust devils. *J. Atmos. Sci.*, 55:3244–3252, 1998.

[17] O. B. Toon, C. P. McKay, T. P. Ackerman, and K. Santhanam. Rapid calculation of radiative heating rates and photodissociation rates in inhomogeneous multiple scattering atmospheres. *J. Geophys. Res.*, 94:16,287–16,301, 1989.