VENUS CLIMATE DATABASE v2.3
USER MANUAL

("Maintenance and Update of the Venus Climate Database")

(ESA Contract N° 4000138542/22/NL/CRS "Venus Climate Database")

S. Lebonnois, E. Millour, F. Forget, A. Martinez, T. Pierron, A. Boissinot (LMD, Paris)

May 2023

Abstract

This document is the User Manual for version 2.3 of the Venus Climate Database (VCD) developed by LMD (Paris) with the support of the European Space Agency. This is a database of atmospheric statistics compiled from Global Climate Model (GCM) numerical simulations of the Venusian atmosphere. This document includes a thorough description of the access software provided to extract and postprocess data from the database.

The database extends up to exobase (the top of the thermosphere, roughly at 250 km in altitude); in addition to statistics on temperature, wind, pressure, radiative fluxes, it provides data such as atmospheric composition for different solar EUV conditions. Linear interpolation in time of datafiles data is used to reconstruct variables at user-specified time of day and various kind of vertical coordinates may be specified as input. The VCD access software, the Fortran routine `call_vcd` includes a "high resolution mode"using high resolution (23 pixels/degree) topography. Examples of interfaces for users interested in calling subroutine `call_vcd` from C, C++ or Python programs or IDL, Matlab and Scilab softwares are also given.

For descriptions of the contents and structure of the datafiles, details on the scenarios, a description of the variability models, and of the high resolution mode, see the **Detailed Design Document**.

Contents

1 Introduction

The Venus Climate Database (VCD) is a database of atmospheric statistics compiled from state-of-the art Global Climate Model (GCM) simulations of the Venusian atmosphere.

The GCM computes in 3D the atmospheric circulation and climate taking into account radiative transfer through the gaseous atmospheres, includes a representation of non-LTE processes, EUV heating, conduction and molecular diffusion in the thermosphere. Sub-grid processes such as convection in the boundary layer, non orographic and orographic gravity waves are also simulated. Photochemistry that controls the atmospheric composition is also implemented in the model.

The models used to compile the statistics have been extensively validated using available observational data and represent the current best knowledge of the state of the Venusian atmosphere given the observations and the physical laws which govern the atmospheric circulation and surface conditions on the planet.

The Venus Climate Database access software moreover includes several post-processing schemes to better represent and account for the Venusian environment variability and accurately compute surface pressure and atmospheric temperature at high spatial resolution.

The VCD is under the ESA-PL (European Space Agency Public License) and is therefore freely available to all on request. A web interface is available to make plots and figures using the interactive server on our online server at:
`http://www-venus.lmd.jussieu.fr`

1.1 Available Data

The VCD contains several statistics on simulated data stored on a $3.75° \times 1.875°$ longitude–latitude grid from the surface up to an approximate altitude of 250 km: temperature, wind, density, pressure, radiative fluxes, atmospheric composition and gas concentrations etc...

Fields are averaged and stored 24 times a day to give a **comprehensive representation of the diurnal cycles**. In other words, at every grid-point, the database contains 24 time steps. In addition, information on the variability of the data within one hour and the Venusian day to day oscillations are also stored in the database. Software tools are provided to reconstruct and synthetize this variability (section **??**).

1.2 Database scenarios

Various combinations of cloud albedo and solar scenarios are provided.

- On the one hand, the solar conditions describe variations in the Extreme UV input which control the heating of the atmosphere above around 100 km, which typically varies on an 11-year cycle. Depending on the scenarios, either fixed (i.e. constant over time) solar maximum, average and/or minimum conditions are provided, or deduced from Earth date for a realistic solar EUV, i.e. as observed at the desired date. The minimum, average and maximum EUV scenarios correspond, respectively, to a value of E10.7 factor of 70, 140 and 300 sfu (solar flux unit : 1 sfu $= 10^{-22}$ W.m^{-2}.Hz^{-1}). The VCD user has also the possibility to set the solar EUV at a desired value as input of the `call_vcd` routine. A linear interpolation is used between minimum and average or average and maximum solar EUV scenarios.

- On the other hand, there is still uncertainties about the Venus solar heating rate. As shown in the work of Lee et al., 2019, this Venus solar heating rate is directly related to the cloud albedo through a radiative transfer model. Therefore the heating rate is calculated for each scenario in order to simulate the albedo measured by Hubble and MESSENGER mission during its flight over Venus. Three kinds of cloud albedo scenarios are proposed :

  1. The **"Standard cloud albedo" (standard)** scenario, which is a simulations ran using the latest version of the LMD Global Climate Model (GCM)

2. The **low** scenario corresponds to a low cloud albedo conditions.
3. The **high** scenario corresponds to high cloud albedo conditions.

- All the combinations of solar conditions and cloud albedo are available as long as all the data have been downloaded. Overall, this leads to a total of up to 15 different available scenarios since there are 5 EUV cases (including the possibility to set manually the solar EUV or to deduce it from an Earth date) and 3 albedo cases scenarios, independent from each other.

The user can refer to the Detailed Design Document for further information on specific aspects of these scenarios.

1.3 High resolution mode

The Venus Climate Database has been compiled from the output of a general circulation model in which the topography is very smoothed because of its low resolution.

The access software includes a "high resolution" mode with a high resolution (23 pixels/degree) topography from Magellan dataset. The latter is then used to reconstruct vertical pressure levels and hence, within the restrictions of the procedure, yield high resolution values of atmospheric variables.

2 Contents of the Venus Climate database

The contents of each subdirectory of the VCD distribution are summarized here.

`docs` This directory contains files in pdf formats which can be used to print further copies of the documentation:

- The User Manual (`user_manual.pdf`) of the database; this document.
- Detailed Design Document (`detailed_design.pdf`)
- pdf versions of the scientific reference articles describing various aspects of the Global Climate Model used to compile it are also provided.

`vcd` This directory contains Fortran source code for the climate database access softwares (see section **??**, and the `README` file in the directory): the main module `VCD.F90` containing the main routine `call_vcd` and subsidiary routines, the module `VCD_var.F90` including common parameters to `call_vcd` and a test program.
Subdirectory `testcase` contains a simple tool to test the results from the software after installation.
Subdirectories `idl`, `matlab`, `scilab`, `c_interface` and `python` contain examples of interfaces of the Fortran subroutine with other languages and softwares.

`data` The full VCD datasets derived from model runs. Essential and common datafiles (`EUV_julian.txt`, `relief.nc`, `surface.nc` and subdirectory `VIRA`) are in this directory. Scenarios datafiles are in corresponding subdirectories (`mean_aveEUV` for the baseline standard cloud albedo scenario with average Extreme UV (EUV) input, `mean_minEUV` for the baseline standard cloud albedo scenario with minimum EUV input, `mean_maxEUV` for the baseline standard cloud albedo scenario with maximum EUV input, `low_aveEUV` for the low cloud albedo scenario with average EUV input, `low_minEUV` for the low cloud albedo scenario with minimum EUV input, `low_maxEUV` for the low cloud albedo scenario with maximum EUV input, `high_aveEUV` for the high cloud albedo scenario with average EUV input, `high_minEUV` for the high cloud albedo scenario with minimum EUV input and `high_maxEUV` for the high cloud albedo scenario with maximum EUV input). By default, only the three standard cloud albedo scenarios are included. The others have to be downloaded separately, in two add-on of three scenarios sharing the same albedo each. The scenarios datafiles are necessary to use the VCD with the corresponding input of `EUV_scena` and `albedo_scena` (see section **??**).

The ESA-PL licence which defines the using and diffusion principles of the software is also given in the VCD with a README file describing the contents of each subdirectory.

3 Installation

3.1 System and Software Requirements

- The VCD is primarily designed to operate in the **Unix/Linux** environment on a PC or a workstation. Access software is written in **Fortran** (and uses some Fortran90 extentions and intrinsics), for which a Fortran compiler is needed.

  Since the NetCDF libraries (see below) are also available under **Windows** or **McOS** systems, it should be possible to compile and use the access software as under **Unix/Linux**.

- The data in the VCD is written as Network Common Data Form (**NetCDF**) files. It is therefore necessary to install it to be able to use the VCD software. The NetCDF library is freely available from the Unidata web site :

  `http://www.unidata.ucar.edu/software/netcdf/`
  Note that you do not necessarily need recent versions (i.e. 4.x) of the library; older ones (e.g. 3.6.x) will work nicely. Note that for the more recent versions, the Fortran library is called libnetcdff (rather than libnetcdf).

3.2 Installing the VCD

1. Create a working directory, e.g. `venus`, on a disk where you wish to use the database.

2. Extract (untar) the contents of the VCD distribution there.

3. With the installation of the minimal baseline VCD distribution, only the “standard” cloud albedo scenario with an average Extreme UV solar input is provided (corresponding datafiles are in the `mean_aveEUV` subdirectory of the `data` directory). The other scenarios (`mean_minEUV`, `mean_maxEUV`, `low_ave`, `low_min`, `low_max`, `high_ave`, `high_min` and `high_max` are provided separately and should likewise be added as subdirectories in the `data` directory. To be able to use all the available scenarios, one have to download all the data files. A full installation (all 9 scenarios) of the VCD takes about 22 Gb of disk space.

4. In the working directory (i.e. where you will run the software) it is convenient to set up a `VCD_DATA` symbolic link[1] to point to the VCD `data` directory, wherever it has been stored:

   ```
   ln -s /full/path/to/vcd/data VCD_DATA
   ```

   N.B In the `call_vcd` subroutine, the path to the directory can be given as an input using the `dset` argument (e.g. `dset='/full/path/to/vcd/data/'`) although by default the subroutine will use `VCD_DATA/` if `dset` is not initialized or set to `' '`.

5. If NetCDF is not available on your system, you must install the NetCDF library following the instructions given on their web site (see above). For this purpose, you have the choice either to build and install the NetCDF package from source, or use prebuilt binary releases if they are available for your platform (check the ”Frequently Asked Question”).

[1]This ”symbolic link” strategy unfortunately *does not work* with Windows where the full path to the data directory must be used.

Ideally, you can install the full NetCDF package as recommanded on the web site. In practice, the minimum you need to run the access software are only two files : an include file named `netcdf.inc` and a library file named `libnetcdf.a` (for older, i.e. pre-version 4 NetCDF; the more recent versions separate Fortran and underlying C libraries as `libnetcdff.a` and `libnetcdf.a`). The version of the files depends of the machine and of the compiler. An easy way to get them is to go to the unidata netcdf web page, click on *precompiled binaries*, download a compressed file corresponding to your operating system, uncompress the file. You'll find the `netcdf.inc` in the `include` directory, and the `libnetcdf.a` in the `lib` directory. You'll need to provide the path to these two files when compiling applications (see the `compile` files in the database). Unfortunately, this does not always work : if your compiler doesn't work with the precompiled library, you will have to recompile Netcdf with it, and follow the web site instructions.

4 Ways to access the database

There are two main ways of accessing data from the VCD which have been implemented to date.

Firstly, if you know Fortran, the **best way** to retrieve environmental data from the Venus climate database at any given locations and times is to use the **subroutine mode** of the software supplied with the Venus Climate Database. In practice, one only has to call a main subroutine named `call_vcd` from within any program written in Fortran. A simple example of such a program (`test_vcd.F90`), which can be easily modified, is provided. This mode was developed with particular attention to trajectory simulation applications, but it can also be used for most other purposes. Further information on the use of `call_vcd` are available below.

Secondly, users who prefer using **IDL**, **Matlab** or **Scilab** software, or who program in **C**, **C++** and/or **Python** will find examples of how to interface their favorite tools with the Fortran subroutine `call_vcd` in corresponding subdirectories (see sections **??** to **??**).

5 Using the CALL_VCD subroutine

5.1 What is the CALL_VCD subroutine ?

The purpose of the `call_vcd` subroutine is to extract and compute meteorological variables useful for atmospheric trajectory computations as well as scientific studies. Data which may thus be obtained includes:

- Atmospheric and surface pressure
- Atmospheric and surface temperature
- Density
- Radiative fluxes (Solar and thermal IR) at a given altitude
- Wind speed defined by two components the meridional wind (positive when oriented from south to north) the zonal wind (positive when oriented from west to east)
- Vertical wind
- The main atmospheric composition : volume mixing ratios of 14 chemical species (including CO_2, N_2, CO, O, He, H)
- The volume mixing ratio of the electrons, accurate between 90 and 180 km
- Column values of all the chemical species
- Air viscosity, heat capacity and γ ratio
- Vertically integrated O_2 nightglow

The Fortran subroutine `call_vcd` retrieves database data at any date (Earth date or Venus local time) and at any point in space defined by latitude and east longitude and a vertical coordinate which can be a distance from the planet center, an altitude (above local surface or above reference sphere), or a pressure level. The returned values of meteorological variables are computed by interpolation in space and time of day from data stored in the VCD. The VCD includes an additional high resolution interpolation procedure (which uses 23 pixels/degree topography) to simulate local pressure and density as accurately as possible.

In order to better estimate atmospheric composition and fit the model with the species densitiy data measured by Pioneer Venus Orbiter above 150km, the photodissociation rate of CO_2 in CO and O has been adjusted to an empirical value. Above the top level of the database, density and pressure are estimated by integration of the hydrostatic equation (separately for each species) assuming a constant temperature.

The subroutine delivers mean values and, if requested, adds a different type of perturbation to density, pressure, temperature and winds. The available types of perturbation are :

- Small scale perturbations due to the propagation (both upwards and horizontally) of gravity waves (generated from cloud convective region) for any altitudes above clouds.
- Large scale perturbations due to the motion of baroclinic weather systems and other transient waves. These perturbations are correlated in longitude and altitude, and are reconstructed from the actual systems predicted by the model.

The perturbations have a random component. A comprehensive explanation of the perturbations is included in the Detailed Design Document[2]. See also section **??** for some discussion and recommendations on the best way to use perturbations to generate realisticly perturbed atmospheres.

5.2 Software Package

The module `VCD.F90` that contains the `call_vcd` subroutine is in the directory `vcd`. This directory includes:

- `README`, a short text file which summarizes the information contained here.
- File `VCD.F90`, which contains the CALL_VCD main subroutine and most of the subroutines and functions it calls.
- The module `VCD_var.F90` used by `call_vcd` including common parameters.
- File `test_vcd.F90` which contains a simple and straightforward illustration of a program using `call_vcd` and `julian`.

[2]In summary the perturbations are calculated as follow :

- The **gravity wave perturbation** of a meteorological variable is calculated by considering vertical displacements of the form
$$\delta z = \delta h \sin\left(\frac{2\pi z}{\lambda} + \phi_0\right) \qquad (1)$$
where λ is a characteristic vertical wavelength for the gravity wave and ϕ_0 is a randomly generated surface phase angle. surface angle. δz is the amplitude of the wave depending on the height z. δh is set in order to have the right temperature perturbations amplitude. If z is higher than 100 km, the amplitude of the wave is taken to be equal to the amplitude at 100 km. The amplitude of the wave is limited to $\lambda/2/\pi$ to saturate the amplitude of the perturbation when it becomes statically unstable. Moreover a horizontal wavelength of ten times λ is used to account for the horizontal propagation of the wave.
- The **large scale perturbation** in the Venus Climate Database is represented using a technique that is widely used in meteorological data analysis, namely, Empirical Orthogonal Function (EOF) analysis. A two-dimensional, multivariate EOF of the main atmospheric variables (surface pressure, atmospheric temperature and wind components) is used which describes correlations in the model variability as a function of both height and longitude. 200 EOFs have been retained in the series in order to reproduce the variability of the original fields. Above the top level of the database, the perturbation represents a constant percentage of the mean value; this percentage is equal to those at the top of the database.

- The `compile` file which contains an example of the (Unix) command line to compile the subroutine and program.
- The file `test_vcd.def` contains a list of input data for `test_vcd` (see section **??**).
- A subdirectory `testcase` containing test cases used to test the accuracy of your installation of the database.
- The `julian.F90` file, which contains a subroutine which computes the Julian date corresponding to a given calendar date.
- Subdirectories `idl`, `matlab`, `scilab`, `c_interfaces`, and `python` which contain examples of interfaces.

5.3 Compiling and running CALL_VCD

A simple program using the `call_vcd` subroutine (as well as the complementary subroutine `julian`) called `test_vcd` is provided in the `vcd` directory. You can easily modify it or use part of this code for your own purpose. To compile the program, edit the `compile` file and make the necessary changes (e.g.: compiler name, path to NetCDF library and include file, see comments in the script). Then, for instance, to compile and run *test_vcd*, type[3]:

```
> compile
> test_vcd
```

Then, just answer the questions...

Alternatively you can edit the file `test_vcd.def` and redirect it to `test_vcd`:

```
> test_vcd < test_vcd.def
```

If the program returns an error, see the list of return error codes in table **??** to identify the problem.

In the `vcd/testcase` sub-directory, a tool to test that `call_vcd` is running accurately on your computer (using `test_vcd`) is provided. Please read `vcd/testcase/README` for further information.

If your machine runs under **Windows** you have 2 solutions:

1. Install a Unix environment emulator for Windows. The most popular is **Cygwin**, which you can download from `http://www.cygwin.com`. This emulator includes most Unix features and softwares. NetCDF libraries can be built on Cygwin as easely as on other Unix systems. Requirements and steps necessary to install and use the provided access software under Cygwin should be almost the same as those for 'standard' **Unix** systems.
2. Port the Venus Climate Database to Windows. NetCDF librairies for Windows can be dowloaded from the NetCDF official website at: `http://www.unidata.ucar.edu/software/netcdf` Note that, with Windows, the "symbolic link strategy" to the Venus Climate Database `data` directory described in section **??** will not work; the 'true' path to that directory must be used in the Fortran routines and programs (see variable `dset` in the description of `call_vcd` arguments in section **??**).

[3]In the examples given here, the > at the begining of command lines is the Unix session command prompt.

5.4 CALL_VCD input and output arguments

A Fortran call to subroutine `call_vcd` should look something like:

```
call call_vcd(z_key,z,lon,lat,hires_key,date_key,juliandate,&
 localtime,dset,EUV_scena,albedo_scena,varE107,perturb_key,perturb_seed,&
 perturb_gw_length,extvar_keys,zon_wind,mer_wind,vert_wind,&
 temp,pres,dens,extvar,seed_out,ier)
```

All the input arguments (ie: values which must be set before calling the routine and which are not altered by it) are described in table **??**. Outputs are described in table **??**.

As `call_vcd` runs, it writes informational (and error) messages to standard output. Users who wish to run `call_vcd` silently (i.e. without any messages sent to standard output) should edit file `VCD_var.F90` and change the value of parameter `output_messages` to `.false.`
The "standard output" unit number which will be used by `call_vcd` is set to the value of the `out` parameter, also defined in file `VCD_var.F90`. The default value of `out` is 6, which is the standard Fortran value preconnected to the screen (on most systems). Setting `out` to any other positive integer value n (except 5 which is usually preconnected to standard input) will send messages to the corresponding file. It is thus advised to open the corresponding file (using Fortran command `open(unit=`n`, file="myfilename")`) prior to any call to `call_vcd` otherwise default file `fort.`n will be used (this behaviour is possibly system-dependent).

Warning

The version 2.3 introduces new physical fields as supplementary variables. In order to keep the `extvar_keys` logically ordered, some changes have been effected. The newly added supplementary variables are the H_2SO_4, liquid H_2O and liquid H_2SO_4 volume mixing ratios and the H_2SO_4 column density. The pre-existing supplementary variables whose the key is modified are the SO_2, SO, OCS, O_3, HCl, He and electron volume mixing ratios, the SO_2, SO, OCS, O_3, HCl and He column densities and the vertically integrated O_2 nightglow. All the keys affected by those modifications are written in bold characters in the table **??**.

Table 1: **CALL_VCD Input arguments**

Name	**Type**	**Description**
`z_key`	integer	Flag to set the type of vertical coordinate `z` is given as: 0: z is the pressure level (Pa). 1: z is the radial distance from the center of the planet (m). 2: z is the altitude above the Venus reference sphere, ie. a sphere of radius 6051848m, in meters. 3: z is the altitude above the local surface (m). Depending on the value of flag `hires_key`, topography is with respect to GCM grid or high resolution data.
`z`	real	Vertical coordinate of the requested point. Its exact definition depends on the value of input argument `z_key`.
`lon`	real	East Longitude (planetocentric), in degrees.
`lat`	real	Latitude (planetocentric), in degrees.
`hires_key`	integer	Flag to set the resolution at which data retrieval and postprocessing should be done: 0: Interpolate data from GCM $3.75° \times 1.875°$ grid. 1: Use high resolution (23 pix/deg.) topography, as well as some internal post-processing scheme to reconstruct data (see section **??**).
`date_key`	integer	Flag to set the dates (`juliandate` and `localtime`): 0: "Earth time"; the user specified a date in Julian days with the argument `juliandate`. With `date_key=0`, the `localtime` argument, although unused, **must be set to zero**. 1: "Venus time"; `localtime` is the value of the local true solar time at longitude `lon`, given in Venusian hours. In this case the argument `juliandate` **must be set to zero**.
`juliandate`	double precision (`REAL*8`)	Julian date. Should only be specified if `date_key=0` and must be set to zero if `date_key=1`. **N.B.** The subroutine `julian.F90` can be used to compute the Julian date corresponding to a given calendar date (day, month, year, hours, minutes, seconds) on Earth. Note that this date essentially matters in order to compute the corresponding time of day on Venus (and that a different Earth date leading to same local time will yield identical results).
`localtime`	real	Local true solar time at longitude `lon`, in Venusian hours. Should only be specified if `date_key=1` and must be set to zero if `date_key=0`. **N.B.** Local true solar time is such that the sun is highest in the sky at noon. A Venusian hour is defined as $1/24^{\text{th}}$ of a Venusian day (which is 10087200 s long).

Table 1: **CALL_VCD Input arguments (continued)**

`dset`	character string dim(*)	Path to the directory where the datafiles are to be found. `dset` may be of any size. If `dset` is an empty string, then the path to the datasets is set the default path `VCD_DATA/`. **N.B.** the given path must end with a "/" (e.g.: `/home/data/`) on Linux and "\" on Windows.
`EUV_scena`	integer	**Solar EUV scenarios** : 1 = Solar EUV average conditions 2 = Solar EUV minimum conditions 3 = Solar EUV maximum conditions 4 = EUV input as deduced from the input Julian date `juliandate` 5 = EUV input as specified from input argument `varE107` **N.B.** The solar conditions describe variations in the Extreme UV input which control the heating of the atmosphere above ∼95 km, which typically varies on a 11 years cycle. The **average** scenario is designed be representative of a baseline typical Venus day while the **minimum** and **maximum** scenarios are provided to bracket the possible solar extreme UV fluxes.
`albedo_scena`	integer	**Cloud Albedo scenarios** : 1 = Standard cloud albedo scenario 2 = Low cloud albedo scenario 3 = High cloud albedo scenario **N.B.** The **standard** scenario is designed be representative of a baseline typical Venus day while the **low** and **high** scenarios are provided to bracket the possible cloud albedos.
`varE107`	real	Value of the E10.7 factor (in SFU) specifying the solar EUV Should be between 70 sfu and 300 sfu. Should be 0 when selecting other scenario than 5.
`perturb_key`	integer	Flag to set the type of perturbation to add. 0: None. 1: Add small scale perturbations (gravity waves). 2: Add large scale perturbations (using EOFs). 3: Add small and large scale perturbations. **N.B.** For the small scale or large scale perturbations, a seed for the random number generator must be specified (see `perturb_seed` argument). When large scale perturbation is requested, as long as `perturb_seed` remains the same, no new random vector is generated and you work with the same correlated perturbed atmosphere. The wavelength of gravity waves is given as input with argument `perturb_gw_length`
`perturb_seed`	integer	if `perturb_key=1`, `2` or `3` : Random number generator seed and flag. For the first call to `call_vcd`, this value is used to seed the random number generator. If the value of `perturb_seed` is changed between subsequent calls to `call_vcd`, it triggers the reseeding of the random number generator and subsequently the regeneration of a new perturbed atmosphere (see section **??**).

Table 1: **CALL_VCD Input arguments (continued)**

`perturb_gw _length`	real	Vertical wavelength of the gravity wave (in meters); the horizontal wavelength of the gravity wave is set to ten times its vertical wavelength. Used for small scale perturbations (ie: if `perturb_key=1` or `3`). Should be between 2000 and 30000 m Should be set to zero if `perturb_key=0` or `2` **N.B.** *Feature for specialists*: Changing the value of `perturb_gw_length` between calls to `call_vcd` triggers the generation of a new random phase for the gravity wave (and without altering the large scale perturbation, if the later is also requested, i.e. in the `perturb_key=2` case).

`extvar_keys`	integer (dim 100)	**Flags to request extra variables on output**
		Each element i of this array signals whether the ith element of optional outputs (the `extvar` array) should be given on output.
		if `extvar_keys(i)=0`, then extra variable `extvar(i)` is not computed.
		if `extvar_keys(i)=1`, then extra variable `extvar(i)` is computed.

Table 2: **CALL_VCD output arguments**

Name	**Type**	**description**
`zon_wind`	real	**Zonal component of wind**, in m/s (>0 if westward)
`mer_wind`	real	**Meridional component of wind**, in m/s (>0 if northward)
`vert_wind`	real	**Vertical component of wind**, in m/s (>0 if upward)
`temp`	real	**Atmospheric temperature** (K)
`pres`	real	**Atmospheric pressure** (Pa)
`dens`	real	**Atmospheric density** (kg/m^3)

Table 2: **CALL_VCD output arguments (continued)**

extvar	real (dim 100)	**Supplementary variables** array Outputs only computed and set if the corresponding input argument `extvar_keys=(i)` is set to 1. These are otherwise set to zero. • extvar(1) = Radial distance to planet center (m) • extvar(2) = Altitude above the reference sphere (m) • extvar(3) = Altitude above local surface (m) • extvar(4) = Orographic height (m) (altitude of the surface with respect to the reference sphere). • extvar(5) = Orographic height in the GCM (m) • extvar(6) = Ls, solar longitude of Venus (deg), Only given if `date_key`=0 (Earth date), otherwise set to -999. • extvar(7) = Sun-Venus distance (in Astronomical Unit AU). Precisely calculated if `date_key`=0 (Earth date), otherwise set to mean value : 0.7245AU • extvar(8) = LTST: Local True Solar Time at longitude `lon` (in Venusian hours = 1/24 of a Venus day). • extvar(9) = Universal solar time (LTST at `lon`=0) (hrs) • extvar(10) = Solar zenith angle (deg) • extvar(11) = Unperturbed (climatological) zonal wind (m/s) • extvar(12) = Unperturbed (climatological) meridional wind (m/s) • extvar(13) = Unperturbed (climatological) vertical wind (m/s) • extvar(14) = Unperturbed (climatological) atmospheric temperature (K) • extvar(15) = Unperturbed (climatological) atmospheric pressure (Pa) • extvar(16) = Unperturbed (climatological) atmospheric density (kg/m^3) • extvar(17) = Surface temperature (K) • extvar(18) = Surface pressure (Pa) All RMS is either pressure-wise (if `z_key`=0) or altitude-wise (if `z_key`=1, 2 or 3). • extvar(20) = V-hourly variability (RMS) of the zonal wind (m/s) • extvar(21) = V-hourly variability (RMS) of the meridional wind (m/s) • extvar(22) = V-hourly variability (RMS) of the vertical wind (m/s) • extvar(23) = V-hourly variability (RMS) of the atmospheric temperature (K) • extvar(24) = V-hourly variability (RMS) of the atmospheric pressure (Pa), if `z_key`=1, 2 or 3. Otherwise set to zero. • extvar(25) = V-hourly variability (RMS) of the atmospheric density (kg/m^3) • extvar(26) = V-hourly variability (RMS) of the surface temperature (K) • extvar(27) = V-hourly variability (RMS) of the surface pressure (Pa)

Table 2: **CALL_VCD output arguments (continued)**

		• extvar(30) = Vday to Vday variability (RMS) of the zonal wind (m/s) • extvar(31) = Vday to Vday variability (RMS) of the meridional wind (m/s) • extvar(32) = Vday to Vday variability (RMS) of the vertical wind (m/s) • extvar(33) = Vday to Vday variability (RMS) of the atmospheric temperature (K) • extvar(34) = Vday to Vday variability (RMS) of the atmospheric pressure (Pa) if `z_key`=1, 2 or 3. Otherwise set to zero. • extvar(35) = Vday to Vday variability (RMS) of the atmospheric density (kg/m^3) • extvar(36) = Vday to Vday variability (RMS) of the surface temperature (K) • extvar(37) = Vday to Vday variability (RMS) of the surface pressure (Pa) • extvar(40) = Net Solar Flux (SW) received at the top of the atmosphere (W/m^2), positive downward • extvar(41) = SW[4] net flux at given altitude (W/m^2), positive downward • extvar(42) = LW[5] net flux at given altitude (W/m^2), positive upward • extvar(43) = Atmospheric scale height at given altitude (m) • extvar(44) = Atmospheric mean molar mass at given altitude (g/mol) • extvar(45) = Atmospheric speed of sound cs (m/s) • extvar(46) = Atmospheric reduced molecular gas constant r (J/K/kg) • extvar(47) = Atmospheric heat capacity Cp (J/kg/K) • extvar(48) = Atmospheric specific heat ratio γ (N/A) • extvar(49) = Atmospheric viscosity estimation (Pa.s) • extvar(50) = CO_2 volume mixing ratio (mol/mol) • extvar(51) = CO volume mixing ratio (mol/mol$_{air}$) • extvar(52) = O_2 volume mixing ratio (mol/mol$_{air}$) • extvar(53) = O volume mixing ratio (mol/mol$_{air}$) • extvar(54) = H volume mixing ratio (mol/mol$_{air}$) • extvar(55) = H_2 volume mixing ratio (mol/mol$_{air}$) • extvar(56) = H_2O volume mixing ratio (mol/mol$_{air}$) • **extvar(57) = H_2SO_4 volume mixing ratio (mol/mol$_{air}$)** • **extvar(58) = SO_2 volume mixing ratio (mol/mol$_{air}$)** • **extvar(59) = SO volume mixing ratio (mol/mol$_{air}$)** • **extvar(60) = OCS volume mixing ratio (mol/mol$_{air}$)**

[4]Short wavelength corresponding mainly to the incident solar flux

Table 2: **CALL_VCD output arguments (continued)**

		• **extvar(61) = O_3 volume mixing ratio (mol/mol$_{air}$)** • **extvar(62) = HCl volume mixing ratio (mol/mol$_{air}$)** • **extvar(63) = N_2 volume mixing ratio (mol/mol$_{air}$)** • **extvar(64) = He volume mixing ratio (mol/mol$_{air}$)** • **extvar(65) = Electron volume mixing ratio[6] (mol/mol$_{air}$)** • **extvar(66) = Liquid H_2O volume mixing ratio (mol/mol$_{air}$)** • **extvar(67) = Liquid H_2SO_4 volume mixing ratio (mol/mol$_{air}$)** • extvar(70) = CO_2 column (kg/m^2) • extvar(71) = CO column (kg/m^2) • extvar(72) = O_2 column (kg/m^2) • extvar(73) = O column (kg/m^2) • extvar(74) = H column (kg/m^2) • extvar(75) = H_2 column (kg/m^2) • extvar(76) = H_2O column (kg/m^2) • **extvar(77) = H_2SO_4 column (kg/m^2)** • **extvar(78) = SO_2 column (kg/m^2)** • **extvar(79) = SO column (kg/m^2)** • **extvar(80) = OCS column (kg/m^2)** • **extvar(81) = O_3 column (kg/m^2)** • **extvar(82) = HCl column (kg/m^2)** • **extvar(83) = N_2 column (kg/m^2)** • **extvar(84) = He column (kg/m^2)** • **extvar(89) = Vertically integrated O2 nightglow (Δ emission) (MR[7])** • extvar(90) = VIRA temperature (K) at the same location • extvar(91) = VIRA pressure (Pa) at the same location • extvar(92) = VIRA density (kg/m^3) at the same location VIRA outputs are set to -999 if outside the VIRA tabulated range.
`seed_out`	integer	The current index of the random number generator. May be used to trigger (by setting `perturb_seed` to this value) the generation of a new set of perturbations for the next call to `call_vcd`.

[5] Long wavelength (1.7 to $250\mu m$) corresponding mainly to the energy from the surface and the atmopshere towards space

[6] The electronic density is accurate only between 90 and 180 km due to the physical process currently implemented in the GCM.

[7] 1R= $\frac{10^{10}}{4\pi}$ photons.m^{-2}.s^{-1}.sr^{-1}

Table 2: **CALL_VCD output arguments (continued)**

`ier`	integer	**Status code**: When an error occurs in `call_vcd`, all the outputs arguments are set to -999 and a message is written to the standard output. The value of `ier` summarises the status of `call_vcd` : 0: OK (no error) 1: Invalid vertical coordinate flag `z_key` 2: Invalid value for high resolution flag `hires_key` 3: Invalid value for date flag `date_key` 4: Invalid value for latitude `lat` 5: Invalid value for Julian date `juliandate` 6: Invalid value for Local time `localtime` 7: Invalid value for scenario `scena` 8: Invalid value for Solar EUV flux `varE107` 9: Invalid value for perturbation flag `perturb_key` 10: Invalid value for wavelength `perturb_gw_length` 11: Invalid value for extra variables output flag `extvar_keys` 12: Given (or computed) altitude is below the surface 13: Input datafile not found 14: Failed loading data from a VCD database file

5.5 The right use of the CALL_VCD subroutine

5.5.1 Optimizing run time with repeated calls to the VCD

In order to minimize computational time, the datasets corresponding to encompassing Venusian hours of a given VCD cloud albedo and EUV scenario are loaded from the database at the first call of the `call_vcd` subroutine. This initial loading is time consuming, but once loaded these values can then be used for further calls, as long as the sought dates do not lead to a change in bracketing Venusian hours. This should be taken into account when simulating trajectories (or maps) over more than an hour.

Similarly, only required data sets are loaded: if for instance no perturbations are requested, then only mean values are loaded. Note that requesting perturbations (`perturb_key` set to 1,2 or 3), as well as supplementary outputs (elements of `extvar_keys` set to 1), implies extra computations which will slow down the program.

Extra optimizations in the code avoid some recomputation as long as the universal time (local time at longitude 0°) remains unchanged. It is therefore recommended, when possible, to have the vertical position variations and the latitude variations in the innermost loops of repeated calls; just as (as explained above) longitude and time variations should be in the outermost loops.

5.5.2 Perturbed atmospheres

In addition to the mean atmospheric state, the user may obtain a realistically perturbed (using input flag `perturb_key`) atmosphere.

To generate randomly perturbed atmospheres you must use small or large scale perturbations which take into account some correlation of perturbations in the space and between variables.

When generating a randomly perturbed atmosphere **using the large scale perturbations** (`perturb_key = 2` or `3`) to simulate a trajectory, the value of `perturb_seed` should be kept constant in order to work with the same correlated perturbed atmosphere.

Reseting the large scale perturbations (by modifying the value of `perturb_seed` between calls to call_vcd) to build profiles at a given location should only be done in the context of generating a range of different possibilities (an example is given in table **??**).

When **using the perturbation due to gravity wave propagation (small scale perturbation)** (perturb_key = 1 or 3), the phase (ie: the value of perturb_seed), as well as the associated wavelength perturb_gw_length should remain fixed as long as one is not considering a different possible atmosphere (as illustrated in the example in table **??**).

6 Calling the CALL_VCD subroutine from IDL

The Interactive Data Language (IDL) is a commercial software for data analysis and visualization tool that is widely used in earth, planetary science and astronomy.

The vcd/idl subdirectory contains IDL scripts and Fortran wrapper routines to use to be able to interactively call the VCD call_vcd routine from IDL. Refer to the README file in the directory for instructions on the required steps to compile the wrapper routines, and check the illustrative example script test_vcd4idl.pro to learn how to call the VCD from your own IDL scripts.

7 Calling the CALL_VCD subroutine from Matlab

An interface to interactively retrieve data from the VCD from within Matlab is provided in the matlab directory. Gateway Fortran routines mlb_call_vcd.F90 to access call_vcd.F90 are given, along with Matlab scripts to generate the needed interface files. See the README file in the same directory for further comments on adapting these interfaces to your settings, and the test_mlb2vcd.m Matlab demo script to see how one then interactively calls the VCD from Matlab.

8 Calling the CALL_VCD subroutine from Scilab

Scilab is a free open source scientific software (similar to Matlab) providing a powerful computing environment for engineering and scientific applications.

An interactive interface to retrieve VCD data from within Scilab is provided in the vcd/scilab subdirectory. This requires using wrapper (or gateway) C functions which must first be compiled, as explained in the README file. See also the illustrative example Scilab script call_vcd.sce to learn how to call the VCD from your own Scilab scripts.

9 Calling the CALL_VCD subroutine from C or C++ programs

Examples of C and C++ programs interfaced with the Fortran subroutine call_vcd are given in the vcd/c_interfaces subdirectory. These files, along with the header file vcd.h, illustrate how to call the Fortran subroutine call_vcd from C (test_vcd.c) and C++ (test_vcd.cpp) main programs.

Unfortunately, inter-language calling conventions vary with compilers and operating systems; although the C and C++ interfaces have been tested on our Linux systems, using Gnu compilers (gfortran, gcc, g++) as well as Portland Group compilers (pgfortran, pgcc, pgCC), they will certainly need to be adapted to your settings. Some examples of compiling and linking commands are given in the provided Makefile and README files.

To summarize, compiling and linking in order to build the main (C or C++) program requires the following steps:

1. Create the object files corresponding to the Fortran subroutines which will be called by the main program, e.g.:
   ```
   > pgf90 -c julian.F90
   > pgf90 -c VCD_var.F90 VCD.F90 -I/path/to/netcdf/include
   ```
   which will create objects julian.o, VCD_var.o and VCD.o.

Table 3: Example of Fortran code to illustrate the use of (re-)setting perturbations

```
! build a density profile at a given time and location
! with EOF and GW perturbations
      perturb_key=3
      perturb_seed=100   ! seed perturbations
      z_key=3  ! work in "altitude above local surface" coordinate
      do i=1,100

        z=(i-1)*2000.0 ! go from surface to 200km

        call call_vcd(z_key,z,lon,lat,hires_key,date_key,juliandate,&
   localtime,dset,EUV_scena,albedo_scena,varE107,perturb_key,perturb_seed,&
   perturb_gw_length,extvar_keys,zon_wind,mer_wind,vert_wind,temp,pres,dens,&
   extvar,seed_out,ier)

        profile(1,i)=dens ! store density

      enddo
!
! ... some code here...
! ... moved on to a different time or place 'far' from previous one
! ... such that perturbations should be reset
!
     perturb_seed=seed_out  ! change seedin to regenerate perturbations
     ! build the new density profile
      do i=1,100

        z=(i-1)*2000.0 ! go from surface to 200km

        call call_vcd(z_key,z,lon,lat,hires_key,date_key,juliandate,&
   localtime,dset,EUV_scena,albedo_scena,varE107,perturb_key,perturb_seed,&
   perturb_gw_length,extvar_keys,zon_wind,mer_wind,vert_wind,temp,pres,dens,&
   extvar,seed_out,ier)

        profile(2,i)=dens ! store density

      enddo
```

2. Compile the main program (e.g. `test_vcd.c`) with your C or C++ compiler:
   ```
   > pgcc test_vcd.c VCD_var.o VCD.o julian.o
   -I/path/to/netcdf/include -L/path/to/netcdf/lib
   -l netcdf
   ```
   you will likely need to add other libraries to ensure good Fortran/C/C++ compatibility, but these are extremely compiler (and platform) dependent; check your compiler's manual for instructions.

10 Calling the CALL_VCD subroutine from Python

An exemple of a python script calling the Fortran subroutine `call_vcd` is given in the `vcd/python` subdirectory.

Note that in order to interface the VCD software with python, one must first create the corresponding python interface, using the provided `fvcd_gfortran.sh` script, adequately adapted as explained in the `README` file in the directory.

The provided `test_vcd.py` script illustrates how one can implement a call to the VCD from python.