

This cloud model calculates particle sizes for several major cloud species commonly found in brown dwarf, giant planet, and terrestrial planet atmospheres.

Copyright (C) 2004 By Curtis S. Cooper

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

The choices made for the physics of this particular 1D cloud treatment, which is based on the microphysical timescale approach of W.B. Rossow (1978, *Icarus* v. 36, 1-50), are described in detail in the Cooper et al. (2003) paper:

Cooper, C.S., Sudarsky, David, Milsom, J.A., Lunine, J.I., and Burrows, A. (2003), "Modeling the Formation of Clouds in Brown Dwarf Atmospheres," *Astrophysical Journal*, v586 n2, April 1, 2003.

Acknowledgements. For all your help in writing this program, many thanks to Drew Milsom and Jonathan I. Lunine Thanks also to Christopher Sharp for allowing us to use your chemical equilibrium code, which is the basis for the input tables for the condensation curves used herein. Thank you also to David Sudarsky and Adam Burrows for providing the HD 209458b atmosphere model used in the sample programs.

Using the Cloud Model

1 Directories and Files

1.1 Makefile

Builds the module from the source code as well as the two sample programs. The Makefile is thoroughly documented. Please read the top of it, which contains the variables used to control the compilers, flags, etc., before using this program! Most problems compiling the source can readily be fixed by a few minor modifications to the Makefile. Uses GNU Make (<http://http://www.gnu.org/software/make/manual/make.html>) by Richard Stallman.

To build a Makefile target, just invoke GNU Make with the target name.

These are the basic targets of the Makefile:

all : bring all objects and PROGRAMS up to date wrt sources.

run : compile, link, and then run the main program.

clean : clean out all object file and main program only.

clobber : clean, plus remove *.dep, *.mod, * .

really_clean : Full clean, including source code documentation.

rebuild : clean, then make all.

debug : make all, then run debugger defined by DEBUGGER.

obj : compile but don't link; i.e., bring objects up to date.

tags : tags used by Vim and Emacs for efficient source code navigation.

guide : builds a PS and PDF version of the Guide.tex file.

view_guide : read output Guide.ps with ghostview.

view_pdf : read output Guide.pdf with acroread.

docs : Doxygen produces very nice HTML source code documentation for this code as it appears on the web page. Install it! It makes browsing the source code comments a much easier task.

browse_docs : 'make docs', then view them with favorite browser.

1.2 data/

Contains all data input into the module.

free_params..txt*—Contains data on the condensable species read in by module *global_cloud*.

**.cond*—The condensation curves of the different species in two columns, temperature in Kelvins and pressure in *dyne cm⁻²*.

1.3 src/

Contains the module source code (.c) (but not the examples).

global_cloud.c—Main cloud module. Use source code documentation to determine code inputs and outputs. Module is used by calling `int global_cloud(...)`. Return values of function `short int global_cloud` are explained in the source code documentation.

vapor.c—Methods for obtaining the saturation vapor pressure or condensation curve of the condensable species the code can handle.

interp.c—General helper methods to perform the interpolation (bilinear).

fileio.c—Helper methods to do File I/O.

1.4 inc/

Contains include files (.h) for module: *global_cloud.h*, *vapor.h*, *interp.h*, *fileio.h*. See the source code documentation for details.

1.5 examples/

sample_C.c, *sample_f77.f*— Sample programs in different programming languages of the use of the module. Currently, *global_cloud.c* has been tested without problems on GNU C and C++, GNU Fortran 77, and the Intel C/C++ and Fortran compiler suites. Difficulties in calling the program using the Portland Group compiler suite and on Sun's native

C/Fortran compilers were observed. Inter-language operation should work fine, but may require slight modifications owing to the non-portable nature of the process.

II. Compiling

The routines of the code are all ANSI C (1989) compatible; since they do not use any non-ANSI compatible features, it should compile on any ANSI C compiler. For standards conformance checking on GNU C, I recommend the `-std=c99` flag (C99 accepts C++ style comments) or `-ansi`, which is equivalent to `-std=c89` (ANSI C Standard, 1989). I also typically specify `-pedantic` to more rigorously enforce the standard.

Or, you can compile everything with an ANSI/ISO C++ compiler, which should work fine. In that case, however, you will need to use `-lstdc++` during the linkage phase or use `C++` to direct the linkage process. C++ *must* be used to direct the linkage process when non-C features of C++ are used to ensure proper construction and destruction of non-local static objects.

If you're planning to use the routine in a C or C++ program, you should `#include "global_cloud.h"` in the source file of the calling function so that the prototype for `'short global_cloud(...)'` is available for the compiler to perform strong type checking of the input parameters. Then, dimension (i.e., allocate under the assumption that they will be accessed by `global_cloud` from `0..nr-1`, where `i = 0` is the first atmospheric layer and `nr-1` is the last) all input and output arrays, set the values of the inputs parameters, and call `global_cloud(...)`.

For Fortran 77 callers, type checking is not available at this time, though I may implement a Fortran 90+ interface file mirroring the procedure's C prototype for strong type checking in Fortran 90+. To call the function from Fortran (any dialect), just use an `'external global_cloud'` specification at the top of the calling routine (or `'external :: global_cloud'`, as the syntax goes in F90+). Be sure to dimension the return type of the function as `'integer'`! Then, dimension all inputs and

output arrays (the Fortran convention of using 1..nr is OK, since you're simply passing a pointer to the C routine and A[0] in C is equivalent to A(1) in Fortran) and simply call the function using a 'return_value = global_cloud(...)' statement, where 'return_value' is dimensioned as an integer type variable. In so doing, the wrapper routine will be called, which will automatically pass all subroutine parameters to the real routine 'short global_cloud(...)'.

III. Extending to new species and other atmospheres

The samples use a now out-of date atmospheric T(P) profile of the transiting extrasolar giant planet HD 209458b. Here, T is in [K] and P in [dyne cm^{-2}]. To use this model on other planetary atmospheres, just use an appropriate T(P) profile for the planet. Note that the code also requires the convective velocity [cm s^{-1}] at each T(P) level in the convective part(s) of the profile (zero in radiative layers).

To extend to more cloud species, simply add condensation files and provide the necessary microphysical parameters in either the homogeneous or heterogeneous parameters file. This will require a straightforward modification of the module code to parse the extended parameter files.

Or, if you send me the thermodynamic data for new condensables, I can merge them into the source code myself, which is what I would prefer so that they can be used by others as well.

The condensable species provided in this version include gehlenite, forsterite, iron vapor, water, and ammonia.

IV. How it works

This version of program of the 1D cloud model is based primarily on the compiled formulas of two approaches:

Cooper et al. (2003), "Modeling the Formation of Clouds in Brown Dwarf Atmospheres," ApJ, April 1, 2003, vol. 586 no 2

Rossow, "Cloud Microphysics: Analysis of the Clouds of Earth, Venus, Mars, and Jupiter," *Icarus*, 36: 1-50, 1978

I have also taken note of revisions to this theory, beginning initially with the subsequent LPL/Steward 1989 cloud paper:

Lunine, J.I. et al., "The effect of gas and grain opacity on the cooling of brown dwarfs", *Astrophysical Journal*, 338:314-337, 1989 March 1.

The code, in its present form, essentially repeats the calculations of Jonathan Lunine's 1989 cloud code but adds physics appropriate to homogeneous nucleation not included in the original version.

The basic algorithm of the code is to compute particle sizes for each of the T/P levels in the atmosphere model, specified as a list of temperatures and pressures in the .pt file (see the example profile for the planet HD209458b). The calculation is performed independently, and does not include the effects of particle mixing between the various cloud levels. We use the .cv file to determine if the region is convective.

If it is convective, we must include the effects of convective upwelling, computed by considering the upward flux at the brown dwarf's effective surface temperature, which is 1270 K for HD209458b (Lunine et al., 1989). If it is not convective, we simply assume the eddy updraft timescale is longer than the timescale of all of the other cloud processes.

Then, starting with a miniscule particle size of 1 micron, we iteratively "build" the cloud particle until the rate for fallout exceeds the rates of the various growth processes: coagulation, condensation, and coalescence. The precise microphysics of each of these processes are described in (Rossow, 1978). Once the rate for rainout is greater than the rates governing cloud particle formation, we say that the cloud particles have attained their maximum size for that level. The code then stores the cloud particle size thus attained, along with the various microphysical timescales, and proceeds to the next cloud layer.

When the cloud particle size and timescales for coalescence, condensation, coagulation, sedimentation, and eddy updraft have been

thus computed for every T/P level specified, the code exits and outputs the results. Therefore, on a plot vs. temperature or pressure within the atmosphere, one can visualize the variation in the relative effectiveness of each cloud microphysical process, as well as the maximum particle size that would be important in the clouds that form. The results of the code may be useful for computing grain opacities if the shape of the cloud particle distribution can be determined. Then, the relative sizes of the important grains for photon scattering can be input with facility into a Mie scattering radiative transport program.