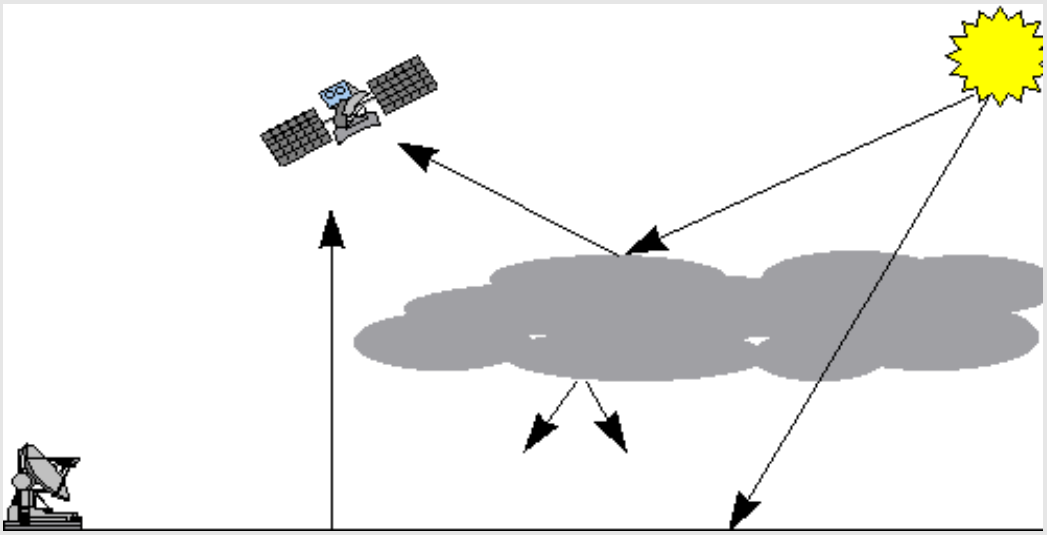
Streamer

Version 3.0

User's Guide



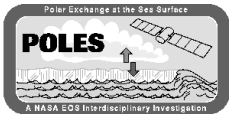
Streamer

User's Guide

Jeffrey R. Key

NOAA/NESDIS
Madison, Wisconsin

August 27, 2002



Part I: Usage 5

INTRODUCTION 6

Input and Output Overview 6
Limitations, Problems, Peculiarities 6
Purpose of This Guide 7
Questions 7
People and Acknowledgments 7
What's New 8
Disclaimer 8
A Note on "Cautions" 8
Referencing Streamer in Publications 8
Comparison with Other Models 9

OBTAINING AND BUILDING STREAMER 11

The Documentation 11
The Program Distribution 11
Cross-Platform Compatibility 12
Other Files 12

RUNNING STREAMER: INPUT 14

Input Overview 14
Specifying Options with the $OPTIONS Command 16
Specifying Data with the $SETDATA and $CASE Commands 21
Setting Default Values 30
Band Weights File 30
Cloud Properties File 31
Surface Bidirectional Reflectance File 32
Spectral Intervals 34
Input Notes 35

OTHER COMMANDS 42

The $REPLACE and $EXCHANGE Commands 42
The $CPRINT Command 45
The $COMMENT Command 46
Interactive Mode: $ASSIGN, $LOAD, $PRINT, $RUN, and $EXIT 46

INPUT QUICK REFERENCE 48

OUTPUT 50

Descriptive Output File 50
Output Notes 51
User-Controlled Output Variable Descriptions 53

SAMPLE INPUT AND OUTPUT 56

Input: Fluxes 56
Output: Fluxes 57
Output: Radiances 58

UTILITIES 60

Streamer as a Subroutine 60
Visible-to-broadband Conversion 61
Cloud and Aerosol Optical Properties 61
Wavelength to Band Conversion and Vice Versa 61
Web Browser Interface 61

WEB VERSION OF THIS DOCUMENTATION 62
SAMPLE APPLICATIONS 65
PROGRAM STRUCTURE 68

Part II: Reference 70

COMPONENT SUMMARY 71
FLUX AND RADIANCE QUANTITIES 72
FLUXES 72
RADIANCES 74
GASEOUS ABSORPTION 76
STANDARD PROFILES 78
AEROSOLS 81
CLOUD PROPERTIES 84
CLOUD OPTICAL PROPERTIES 84
CLOUD THICKNESS AND WATER CONTENT 91
COMBINING AEROSOL AND CLOUD OPTICAL PROPERTIES 91
SURFACE REFLECTANCE 93
SPECTRAL ALBEDO MODELS 93
USER-SPECIFIED BIDIRECTIONAL REFLECTANCE FUNCTIONS 95
BIDIRECTIONAL REFLECTANCE MODELS 95
DEFAULT BRDF MODELS FOR VARIOUS SURFACE TYPES 99
SOLAR FLUXES 100
BAND WEIGHTS 101
REFERENCES 104
REVISION HISTORY 107

PART I: USAGE

1 INTRODUCTION

Streamer is a radiative transfer model that can be used for computing either radiances (intensities) or irradiances (fluxes) for a wide variety of atmospheric and surface conditions. Its interface is very flexible and easy to use. *Streamer*'s major features are:

- Fluxes (irradiances) may be computed using two or more streams, either broadband or narrow band. For more than two streams, a discrete ordinate solver (DISORT) is used.
- Radiances (intensities) may be computed for any polar and azimuthal angles, using 4 or more streams. TOA (top-of-atmosphere) albedos or brightness temperatures are output along with the radiances.
- Upwelling and downwelling shortwave and longwave, and net fluxes, cloud radiative effect ("cloud forcing"), and heating rates can be computed.
- There are 24 shortwave and 105 longwave bands.
- Gas absorption is parameterized with overlapping gases. Gaseous absorption may be turned on or off.
- Liquid and ice cloud optical properties, five aerosol optical models, and four aerosol vertical profiles are part of the model database. Clouds may be specified as some combination of particle size, water content, optical and geometrical thicknesses, or the optical properties and phase function can
- be input. A variety of different ice particle shapes (column, aggregate, etc.) are available. Mixed-phase clouds (liquid and ice or ice and ice) can also be specified.
- There are seven standard atmospheric profiles. Either standard or user-defined profiles can be used, or total column amounts of water vapor, ozone, and/or aerosols can be specified. Standard profiles include tropical, mid-latitude, subarctic, and arctic.
- Each computation is done for a "scene", where the scene can be a mixture of up to 10 cloud types occurring individually, up to 10 overlapping cloud sets of up to 10 clouds each, and clear sky, all over some combination of up to three surface types.
- Various built-in surface types may occur within the scene: open ocean (sea water), meltponds, bare ice, snow, vegetation (four types), and dry sand. Spectral albedo and bidirectional reflectance models (BRDF) are included, or BRDF data can be input.
- A simple user command language provides looping structures for up to ten variables at a time, variables can be reassigned, and output can be customized. There is an interactive (interpreted) mode and a Web interface.

INPUT AND OUTPUT OVERVIEW

Processing is controlled by a set of options and input data in a file. The options define the characteristics common to all the "scenes" or "cases" (e.g., a pixel or observation in time), while the data provides information for each individual case. Output includes surface and top of the atmosphere radiative fluxes, surface albedo, and cloud radiative effect for flux calculations, or radiances and TOA albedo or brightness temperature. Either of two files may be written: one with labeled results, and/or one that is user-customizable.

LIMITATIONS, PROBLEMS, PECULIARITIES

- Unless the cloud phase function is input by the user, it is approximated using the asymmetry parameter and the Henyey-Greenstein function. This is fine for fluxes but can result in significant

errors when radiances are being computed.
- No atmospheric refraction or spherical geometry is included so that results for solar zenith angles greater than about 70 degrees are subject to error.
- The only gases considered are water vapor, oxygen, carbon dioxide, and ozone.
- The flux/radiance that *Streamer* computes for the scene is the sum of the fluxes/radiances computed for each cloud part (individual clouds and overlap) weighted by their area fraction.
- Shortwave calculations are not done when the solar zenith angle is 90 degrees or more. No shortwave calculations are done beyond a wavelength of 4 μm; no longwave calculations are done at wavelengths less than about 3.4 μm.
- While different surface types can occur within a scene, they do not occupy different portions of the scene; i.e., their reflectance signatures are used to create a weighted average albedo for the entire scene.

PURPOSE OF THIS GUIDE

The purpose of this User's Guide is to provide instructions for using *Streamer*. Information on the internal data, parameterizations, and procedures is contained in the Reference guide.

QUESTIONS

Streamer may be obtained via anonymous ftp as described in the next section. If you have questions or bug reports, contact:

Jeff Key
NOAA/NESDIS
1225 W. Dayton Street
Madison, WI 53706
e-mail: jkey@ssec.wisc.edu

Are you on the mailing list? If you have requested information via e-mail then you are. If not, and you plan to use *Streamer*, please send me your complete postal and e-mail addresses.

Continued work on this program is largely unfunded. I'll be happy to answer questions about things that are not in the User's Guide, and will fix bugs in a reasonably short period of time. Keep in mind, however, that I may not be able to provide an immediate response.

PEOPLE AND ACKNOWLEDGMENTS

Streamer has its roots in the program *strats* by **Si-Chee Tsay** and much of the code and the gas absorption data from that program have been incorporated into the original *Streamer*. The functionality of *strats* was extended by **Axel Schweiger**. The transformation of that program into *Streamer* was done by **Jeff Key**.

Scientific programming assistance from **Mark Rehder** and **Rongqian Yang** is much appreciated. The HTML interface was developed by **Victor Nguyen**. **Julie Haggerty** provided the band weights for the KT19. **Jennifer Francis**, **Marcel Haeflige**r, and **James Pinto** did much of the testing of *Streamer*. Thanks to **Bob Stone** for many valuable discussions and for help with the original ice cloud Mie calculations. **Bryan Baum** provided the data for the smoke aerosol model. **Yinghui Liu** created the

MODIS band weights

Two solvers are used in *Streamer*: a 2-stream solution by T. Ackerman and the discrete ordinate code by Stamnes et al. If you have questions concerning DISORT, and only DISORT, contact one of the authors listed in **disort.doc**. The 2-stream solver is completely unsupported in its current form.

Streamer was developed under funding from the NASA EOS interdisciplinary project POLES. Further development is supported by NOAA.

WHAT'S NEW

Version 3 includes DISORT version 2, the capability of specifying a surface bidirectional reflectance function or a built-in BRDF model, a facility to input cloud optical properties including a phase function, new cloud optical property models including eight ice particle shapes, the option to compute cloud water content from optical and geometrical thicknesses, an interactive mode of execution with new commands, a smoke aerosol model, and much more! See the revision history in the *Reference Guide* for detailed information.

DISCLAIMER

This program is distributed as "freeware". Except when otherwise stated in writing the program is provided "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. In no event unless agreed to in writing will the author or any other party who may modify and/or redistribute the program be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program. This includes, but is not limited to, the loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs.

A NOTE ON "CAUTIONS"

There are a number of cautions given throughout this manual. These are not meant to discourage you from using *Streamer*, but rather to inform those users with little experience in radiative transfer of things that this model does not do well. Ultimately you have to decide if what you are trying to do is reasonable, but these cautions should help steer you away from those aspects of the model that have the greatest uncertainty.

REFERENCING *STREAMER* IN PUBLICATIONS

Streamer is comprised of code and data from a variety of sources. Some of it is original, some of it is not. While *Streamer* is a unique compilation, it does not constitute a new approach to radiative transfer and has therefore not been published in any atmospheric science journal. So, if you are presenting results obtained using *Streamer*, how do you reference it? The most complete way would be to reference the individual pieces: the discrete ordinate solver described in Stamnes et al. (1988), the two-stream method following Toon et al. (1989), the water cloud optical properties from Hu and Stamnes (1993), the ice cloud optical properties from Ebert and Curry (1992), the gas absorption data described

in Tsay et al. (1989), etc. See the Component Summary section of the *User's Guide: Reference* for complete details. If you want a single reference, use

Key, J. and A.J. Schweiger, 1998, Tools for atmospheric radiative transfer: Streamer and FluxNet, *Computers & Geosciences*, 24(5), 443-451.

or simply reference this *User's Guide*:

Key, J., 2001, Streamer User's Guide, Cooperative Institute for Meteorological Satellite Studies, University of Wisconsin, 96 pp.

COMPARISON WITH OTHER MODELS

A variety of radiative transfer models exist, both for the calculation of radiative fluxes and the simulation of radiances measured by satellite sensors. However, those used for the calculation of radiative fluxes are generally components of climate models (cf., Ellingson et al., 1991) and are not well documented or easy to use. For this reason they will not be discussed further in this paper. Radiative transfer models that are well documented, reliable, and available to the scientific community include LOWTRAN (Kneizys et al., 1988), MODTRAN (Snell et al., 1995), and 6S (Vermote et al., 1994). While other general-purpose models exist, these three have been widely used in remote sensing applications. They are all medium or high spectral resolution band models and incorporate thorough treatments of gas absorption. However, the cloud models in LOWTRAN/MODTRAN are not very easy to modify, and the two-stream approximation for multiple scattering can result in significant errors under certain conditions. The 6S model does not include clouds but is very flexible for clear sky satellite simulations. None of these models computes fluxes directly, and the user interfaces are somewhat crude. Some of the major similarities and differences between these models are listed in Table 1.

Table 1: Comparison of a few common radiative transfer models.

	Streamer	MODTRAN/MODTRAN	6S
Numerical approximation method(s)	Discrete ordinates and two-stream	Two-stream, including atmospheric refraction; discrete ordinates also in MODTRAN-3	Successive orders of scattering
Spectral resolution	24 shortwave bands; 20 cm^{-1} bandwidth in longwave	20 cm^{-1} (LOWTRAN); 2 cm^{-1} (MODTRAN)	10 cm^{-1}, shortwave only
Clouds	Flexible specification of cloud physical properties; multiple ice cloud particle types; user-specified optical properties	Eight cloud models; user-specified optical properties	No clouds
Aerosols	Six optical models, some user control	Four optical models	Six optical models plus user-defined
Gas absorption*	Principle gases only	Principle and trace gases	Principle and trace gases
Atmospheric profiles	Standard and user-specified	Standard and user-specified	Standard and user-specified
Surface characteristics	Lambertian and BRDF, built-in spectral albedo models and user-specified BRDF	Lambertian, no built-in models	Lambertian spectral albedo models built-in; bidirectionally reflecting surface possible
Primary output parameter	Radiance/reflectance/brightness temperature or flux	Radiance	Radiance/reflectance
User interface	Input file with command language; interactive mode; web interface	Formatted input file	Input file

*In this table, principle gases are H_2O, O_3, CO_2, and O_2. Trace gases include, among others, CH_4, N_2O, and CO.

2 OBTAINING AND BUILDING *STREAMER*

Streamer may be obtained for implementation under UNIX or other operating systems. With a few exceptions the source code is the same for all, but the method of getting the files and building the executable program varies.

THE DOCUMENTATION

The files are **userman.ps, userman.pdf, refman.ps** and **refman.pdf** in the **docs** subdirectory of the main **streamer** subdirectory. These files are Postscript and PDF versions of the manual that can be printed on a postscript printer. You should also be able to view them with a Postscript or PDF viewer (e.g., Ghostview and Acrobat Reader). Because of size limitations with some systems, you may need special printing options; e.g., **lpr -s userman.ps** on a UNIX machine. It's about 2 Mb and 75 pages, so be patient. The manual can be obtained via anonymous ftp as follows:

1. **ftp stratus.ssec.wisc.edu**
2. Use **anonymous** as the username and your e-mail address as the password
3. **cd pub/streamer/docs**
4. **bin**
5. **get userman.ps** (and/or **userman.pdf**)
6. **get refman.ps** (and/or **refman.pdf**)
7. **get htmldocs.tar**
8. **quit**

The file **htmldocs.tar** contains versions of the User's and Reference Guides that can be viewed with a web browser. You can install it anywhere, though you should put a copy of it with the web interface, as described in the *Utilities* section of this manual.

THE PROGRAM DISTRIBUTION

On your machine, create a directory for *Streamer* and switch to it. The *Streamer* program code can then be obtained via anonymous ftp as follows:

1. **ftp stratus.ssec.wisc.edu**
2. Use **anonymous** as the username and your e-mail address as the password
3. **cd pub/streamer**
4. **bin**
5. **get streamer.tar**
6. **quit**

Next, extract the program files:

tar xvf streamer.tar

You can then delete the file **streamer.tar**. Change to the **progs** subdirectory. Rename or copy the makefile for your operating system from **Makefile**.*x* (where *x* is "sun", "hp", "sgi", "ibm") to simply **Makefile**. Then build the program with:

make all

Otherwise, you must compile and link all the *.**f** files as appropriate for your computer system. Makefiles for other machines will be provided as they become available. You should check your compiler documentation for options that might increase execution speed. For Sun Fortran, for example, there is a "-fast" option which nearly halves the computation time (this has been added to **Makefile.sun**).

Binary executable files are available for some operating systems in the **bin** directory. Switch to it before quiting ftp if you do not want to build *Streamer* from scratch, and download the appropriate file. Rename the file for your operating system to simply "streamer".

The *Streamer Web Interface* allows the new user to create *Streamer* input files with a web browser. A tar file in the **gui** subdirectory of the distribution contains the program files. The browser interface must be installed by your system administrator. Have him/her extract the tar file in the **gui** subdirectory and read the **README.install** file. Note: The interface may be updated independent of the model, so check for newer versions in the **gui** subdirectory at the ftp site.

CROSS-PLATFORM COMPATIBILITY

The code is for the most part generic and should run on any computer. The only significant machine-dependent part of *Streamer* concerns machine precision. For runs using more than two streams the DISORT solver is called. DISORT calls the function **d1mach** which returns information on machine precision. The function **d1mach** is expected by the Makefiles to be in the file **d1mach.f**. For Linux (Intel), Sun (Sparc), SGI, HP, and IBM environments, their make files SHOULD use the appropriate d1mach file, copying it to **d1mach.f**. For other operating systems/architectures you will will need modify or rename one of the available files. See the file **README.d1mach** in the **progs** directory.

An additional problem will occur if your implementation of Fortran does not have the **getarg** and **iargc** functions. This is rare, and may only be a problem with DEC VMS. If the functions are not present you must edit **options.f**. These two functions allow you to specify the input file name on the same line as the invocation of the *Streamer* executable (see below). Without them you must comment out the appropriate code, the process of which is explained in **options.f**. Under Windows the function calls are slightly different and you will need to comment/uncomment certain lines in **options.f**.

Streamer has been compiled on a Windows PC using Microsoft's Fortran Powerstation (v4.0), a 32-bit compiler. The executable is available as described above. Either Windows 95/98 or Windows NT is required; the executable will not run under Windows 3.1 or MSDOS alone.

Besides Microsoft Windows, the only other non-Unix operating system that *Streamer* has been tested on to date is VMS on a DEC Alpha. No make files are provided for VMS.

OTHER FILES

The input and output test files are in **testio**, and have extensions of **.inp**, and **.des**. You should run the test cases and compare your results to those provided. You'll need to copy the input files to ones with new names and change the output file names that are specified in the input files so that the originals are

not overwritten. The file **streamer.def** is a sample defaults file. See the *Input* section for information on setting default values.

A few bandweights files are available in the **bandweights** directory for the AVHRR on NOAA 12 (**avhrr12.wts**), HIRS/2 (**hirs2.wts**), GOES-8 imager (**goes8_imager.wts**), and MODIS (**modis.wts**). The HIRS and MODIS weights are based on the half-amplitude band widths; i.e., no response functions were used. The MODIS weights are based on very early specifications and should be considered experimental.

3 RUNNING *STREAMER*: INPUT

The standard way to run *Streamer* is from the command line, giving an input file as an argument:

streamer <input-file-name>

The input file contains a numerical description of the surface and atmospheric conditions for which radiative transfer calculations will be done, as described in the next sections. The specification of these input options and data is the most important, and most time consuming, part of learning to use *Streamer*.

Use "**streamer -h**" to see what command line options are available. For example, you can specify a default values file and the input file with:

streamer -d <defaults-file-name> <input-file-name>

The command

streamer -s <input-file-name>

can be used to echo the values of the input file variables as they are read, which is useful in debugging input problems.

If **getarg** and **iargc** are not available and you modified **options.f** appropriately (e.g., for the DEC Alpha-VMS operating system), type **streamer** then follow the prompts.
Streamer can also be run interactively, where a simple command interpreter is invoked by:

streamer -i

Finally, *Streamer* can be run through a web browser interface. The interface must be installed by your system administrator. Have him/her extract the tar file in the **gui** subdirectory of your distribution and read the **README.install** file. See the *Utilities* section for more information.

The computation time for fluxes (2 streams) is reasonable. For radiances, however, it can take a very, very long time to get results. The computation time is a function of the numbers of levels, streams, spectral bands, cloud types in the scene, and ESFT terms for gaseous absorption. Only the last of these is beyond your control, short of turning off gas absorption.

INPUT OVERVIEW

Streamer is command-driven, where the current command set includes:

OPTIONS
SETDATA
CASE
REPLACE
EXCHANGE
CPRINT
ASSIGN
RUN
LOAD
PRINT
COMMENT or “;”

Commands and their data are placed in a file, the name of which is given with the command to run the model, e.g., **streamer testflx.inp**. Commands must start in the first column of a line in the input file. They can be upper, lower, or mixed case. The leading "$" is optional. It is used in the discussions below for clarity, but is not required. For example, "$PRINT", "$print", "PRINT", "print", and "PrInT" are equivalent. Commands that require arguments must be followed by either a space or a comma; e.g., "print zen" and "print,zen" are both acceptable.

The $OPTIONS command indicates that a block of options should be read. The $SETDATA and $CASE commands signify that data for a new scene (e.g., a pixel or cell) are to be read. These three commands are described in this section. The $REPLACE and $EXCHANGE commands are used to reassign variable values from the previous scene's data, and can be used to initiate loops. Replacements done using $REPLACE are in effect until the next set of data are read with $SETDATA or $CASE but replacements done using $EXCHANGE are only in effect for that statement. $ASSIGN is used for simple variable assignments like $REPLACE. Specific variables can be printed for each case without having to modify the source code by using $CPRINT. Comments can be added to the input file with $COMMENT. $REPLACE, $EXCHANGE, $CPRINT, and $COMMENT are described later.

$CASE, $REPLACE, and $EXCHANGE are "executable" commands; they cause the model to run on variable values that they assign. $OPTIONS, $SETDATA, $ASSIGN, $CPRINT, and $COMMENT only set up data; they do not cause model execution. Not surprisingly, $RUN causes the model to run.

CAUTION

There are many options and variations on how the input stream is structured. Read this documentation very carefully and watch for parameters whose presence/absence and values depend on other parameters. There are many traps for bad input, but there are also many parameters that are not checked.

SPECIFYING OPTIONS WITH THE $OPTIONS COMMAND

Streamer does its computations based on data specified for each "case" or "scene" (e.g., a pixel or cell), further controlled by a set of options that apply to each scene. Some of these variables were set as options rather than data because by nature they apply to all cases; e.g., the output file name. Others were chosen as options because they will likely apply to all cases in a given run of *Streamer*; e.g., the number of streams, whether or not gaseous absorption is included, or the standard profile used.

The $OPTIONS command and associated values may appear anywhere in the input file although it makes most sense to put it at the beginning. This block of data is optional but strongly recommended. If not present, default values will be used (see the section *Setting Default Values* below).

Variable names given below correspond to those in the routine that reads the options and throughout most of the *Streamer* code. The set of options must be preceded by a line starting with the string $OPTIONS; i.e., this is the first line and the title (below) is the second line. All options are free format except for the file name character strings, which must be left justified. However, each option variable or set of variables in the table below, i.e., each row of the table, constitutes a separate line in the input file. For example, the values of the *FLUXES* variable is on a line by itself; *NSTRSHORT* and *NSTRLONG* are on the same line. Comments starting with ";" can follow all required values on a line, but cannot be between required values on a line. See the test input files for examples.

FLUXES	Compute fluxes (.TRUE.) or radiances (.FALSE.)?
IR106	Both thermal emission and solar reflection occur in band 106 (around 3.7 μm). There may be times, however, when the user does not want thermal emission included in radiance calculations for this band. If thermal emission is not desired, then set *IR106* to FALSE, otherwise set it to TRUE. This only applies to radiance calculations and only when band 106 is used, otherwise it is ignored (although required for consistency). For flux calculations band 106 is considered as shortwave only. (See the band table in the *Spectral Intervals* section.)
CLDFORCE	Compute cloud radiative effect (“forcing”) (.TRUE.)? While this only makes sense for fluxes, it is required for consistency. If cloud forcing is not required, setting this to .FALSE. will halve the computation time for those scenes that are completely cloud covered because a clear sky flux must also be calculated. For any other case, there will be an insignificant decrease in computation time.
NSTRSHORT, *NSTRLONG*	The number of streams in computation for shortwave and longwave calculations, 2 or more for fluxes, 4 or more for radiances. Even numbers only, maximum of 48. Of course, both shortwave and longwave fluxes or radiances might not be desired, but both NSTR* values are required for consistency. [integer] Note: Using more than 2 streams for fluxes gives a more accurate solution (since in that case the discrete ordinate solver is used) but computation time increases dramatically.

NCOEF

Both of these apply only if *NSTRSHORT* and/or *NSTRLONG* are greater than 2. *NCOEF* is the number of phase function Legendre coefficients to use for cloud and aerosol phase functions, not including the zeroth moment (1 <= *NCOEF* <= 48 (maximum number of streams)). *NCOEF* can be greater than *NSTRSHORT* and *NSTRLONG*. It should generally be large; e.g., 24 or greater. If both *NSTRSHORT* and *NSTRLONG* are 2 then the value of *NCOEF* is not used but is still required. See *Input Notes* for more information.

GASABS

If gaseous absorption is to be included. If *GASABS*=.FALSE., then only scattering and absorption by clouds and aerosols is considered. (This speeds things up enormously.) Note, however, that profiles or column amounts must still be specified.

RAYLISHRT

Include Rayleigh scattering for shortwave calculations (.TRUE.)? No Rayleigh scattering is considered in the longwave (it is negligible).

ALBTYPE

If shortwave calculations are desired, otherwise not relevant (but still required). *ALBTYPE* controls how the surface albedo for each of the shortwave bands is determined. It can have these values:

0 -> You will supply the albedo or reflectance for each band.
1 -> A built-in spectral albedo model for the desired surface(s) will be used.
2 -> A built-in spectral albedo model will be used, but it will be scaled by a user-specified visible reflectance/albedo (see *rsurf* below).
3 -> A built-in spectral albedo model will be used, but it will be scaled by a user-specified broadband (clear-sky) albedo (see *rsurf* below).
4 -> This value is not used.
5 -> A built-in bidirectional reflectance function (parameterization) will be used for the surface type that you specify.
6 -> You will specify a built-in bidirectional reflectance function (parameterization) and the appropriate parameters.
7 -> You will supply bidirectional reflectance values in a file. See the *BRDFFILE* option below and the *Surface Bidirectional Reflectance File* section.

Values 5-7 are only valid if the number of streams is greater than two. These are explained in more detail in the *Input: Data* section.

EMISSTYPE

If longwave calculations are desired, otherwise not relevant (but still required). If .TRUE., then a single value for the surface emissivity will be read (below) and used for all longwave bands. If .FALSE., then you must specify (below) an emissivity for each band. It can have these values:

0 -> You will supply the emissivity for each band.
1 -> This value is not used.
2 -> This value is not used.
3 -> This value is not used.
4 -> You will supply a single emissivity to be used for all bands.

STDPROF, SPACE

STDPROF is the standard profile to be used as the default in the extension of your profile to space or when no profile is supplied. *STDPROF* [integer] is as follows:

1 -> Tropical
2 -> Mid-latitude summer
3 -> Mid-latitude winter
4 -> Subarctic summer
5 -> Subarctic winter
6 -> Arctic summer
7 -> Arctic winter

If your top level is significantly less than 100 km, and if you want calculations for the entire atmosphere, then set *SPACE*=.TRUE. and the profile will be supplemented above your top level. (See Notes/Profiles below.) If, however, you only want calculations between your top and bottom levels, then set *SPACE*=.FALSE.

AERMOD, AERVERT

AERMOD is the aerosol optical model to be used:

1 -> Tropospheric
2 -> Rural
3 -> Urban
4 -> Maritime
5 -> Arctic haze (for the shortwave; Tropospheric are used for the longwave)
6 -> Smoke

If no aerosols are to be used, set the aerosol optical depth to zero in the data for each case. A valid value of *AERMOD* is, however, still required.

AERVERT determines the vertical distribution of the aerosols. Four profiles are available:

1 -> Background tropospheric aerosols, 50 km visibility, and background stratospheric amounts.
2 -> High tropospheric loadings with 5 km visibility from 0-2 km vertically, 23 km visibility between 2 and 10 km; background stratospheric amounts.
3 -> Background tropospheric amounts as in #1; high volcanic stratospheric aerosols.
4 -> High tropospheric loadings and high volcanic stratospheric amounts.

See *Input Notes* for more information on specifying aerosol optical depth.

IZD, *ITD*, *IWV*, *IO3*, *ICTOP*, *ICTHK* *IWAVE*

Units for altitude, temperature, water vapor and ozone profiles, cloud top/bottom and surface temperatures, top/bottom height, cloud thickness, and wavelength/wavenumber specification. If atmospheric profiles will not be input, then *IZD*, *IWV*, and *IO3* values are not used, unless otherwise indicated in the next section. If band weights are to be read (below), *IWAVE* is ignored. [integer]

IZD - Height profile data in:
1 -> km
2 -> m
3 -> cm

ITD - Temperature (input profiles, cloud top, surface) in:
1 -> K
2 -> C

IWV - Water vapor profiles in:
1 -> Density in g m^{-3}
2 -> Dewpoint temperature (same unit as temperature)
3 -> Relative humidity in percent
4 -> Mixing ratio in g/Kg

IO3 - Ozone profile data as:
1 -> Density in g m^{-3}
2 -> Pressure in atm-cm/km
3 -> # density in cm^{-3}
4 -> Mixing ratio in g/Kg

ICTOP - Cloud Top/Bottom Height:
1 -> Cloud top pressure in mb
2 -> Cloud top height in km
3 -> Cloud bottom pressure in mb
4 -> Cloud bottom height in km

ICTHK - Cloud Thickness:
1 -> Physical thickness in mb
2 -> Physical thickness in km
3 -> Optical thickness at 0.6 μm, unitless
4 -> Optical thickness (unitless) and physical thickness in mb
5 -> Optical thickness (unitless) and physical thickness in km
6 -> Cloud will occupy the single layer with the specified top/bottom height

IWAVE - Wavelength or wavenumber:
1 -> Spectral range will be specified as wavelengths in micrometers
2 -> Spectral range will be specified as wavenumbers in cm^{-1}
3 -> Spectral range will be specified as *Streamer* band numbers
4 -> Spectral range will be specified as a channel number from the band weights file (see WEIGHTFILE below).

Note that if height, water vapor, or ozone profiles are not present then their unit specifications are meaningless (using 0 is a good convention) but must be present. Cloud units must be specified even if there will not be clouds in the input stream (once again, 0 can be used in this case).

OUTLEVS Specifies for which levels the fluxes/radiances, cloud radiative effect, and heating rates are output. Possible values are: 1=top level, 2=bottom level, 3=top and bottom levels, 4=all levels.

DESCRIP Output a file with textual descriptions of the numerical results?

DESCRIPFILE If *DESCRIP*=.TRUE., enter the name of the file that will contain output data and textual descriptions. The name must be 60 characters or less (no other characters can be in the first 60 columns). If *DESCRIP*=.FALSE. then the contents of the line are ignored (but it is still required).

USEROUT Output user-customizable data? If true, then it is expected that the user is either using $CPRINT or has added write statements to the routine **writeusr.f**, thereby printing only those variables of interest. See the Output section.

USERFILE If *USEROUT*=.TRUE., then enter the name of the file for your customized output. The name must be 60 characters or less (no other characters can be in the first 60 columns). If *USEROUT*=.FALSE. then the contents of the line are ignored (but the line is still required).

WEIGHTFILE If *IWAVE*=4 then provide the name of the file containing band weights for some or all of the *Streamer* bands. The purpose and structure of this file are described under *Input Notes, Bandweights*. If *IWAVE* is other than 4 then this line can be blank.

USERCLOUD Read user-specified cloud properties from a file? If .TRUE., a file will be read that contains the extinction coefficient, single scattering albedo, and either asymmetry parameter or the phase function for each wavelength or channel.

CLOUDFILE The name of the cloud properties file, if *USERCLOU*D=.TRUE. See the *Cloud Properties File* section for details. If USER*CLOUD*=.FALSE. then leave a blank line.

BRDFFILE The name of the BRDF file, if *ALBTYPE*=7. See the *Surface Bidirectional Reflectance File* section for details. If *ALBTYPE* is not 7 then you can leave a blank line.

SPECIFYING DATA WITH THE $SETDATA AND $CASE COMMANDS

The $SETDATA and $CASE commands signifies that all data for a scene; e.g., location, viewing geometry, spectral bands, clouds, etc., are to be read. In general, one or the other of these is necessary to run the model. If one of these data blocks is not present, default values will be used (see the section *Setting Default Values* below). The difference between the two is that $SETDATA does not cause *Streamer* to perform any radiative transfer calculations. It simply assigns values to all the variables, setting them up for a subsequent $REPLACE. The $CASE command, on the other hand, causes data to be read and the radiative transfer solver to be run. $SETDATA is only useful when $REPLACE or $RUN follows and if the $REPLACE contains a loop. If $CASE were used in that situation, results for one extra scene (the $CASE data) would be produced. Both of these commands occur by themselves on one line, followed by the data as described below.

In the following format descriptions for each record, "[ ]" means optional variables, "{}" are used for grouping, and "|" means "or". See the test input files for examples. Variable names correspond to those in the routine that reads the data. Each list of variables signifies a new line of input. For example, *rectitle* is the first line, *year, month, day, hour, lat, lon*, and *zen* are on the second line, etc. If BLANK is specified for an input record, the line is required but is ignored and may contain anything. Comments starting with ";" can follow all required values on a line, but cannot be between required values on a line. See the test input files for examples.

rectitle

rectitle	This line can be used as a title or comment for this case (80 characters maximum).

year month day hour lat lon zen

year	2- or 4-digit year (4-digit is preferred), e.g., 1992 or 92. If 2-digit, then values less than 50 represent 20xx; if 50 or more, then 19xx. [integer]
month, day	Two-digit month, e.g., 11, 6 [integer]
hour	Greenwich Mean Time on the 24 hr scale; e.g., 13.2 [real]
lat,lon	Latitude (positive in Northern Hemisphere) and longitude (0 to +-180, positive west of Prime Meridian) [real]
zen	Solar zenith angle (degrees). **If it is to be calculated based on the location and time, use a negative value.** [real]

IMPORTANT: The date variables are used to adjust the solar flux for Earth-Sun distance and to calculate the solar zenith angle (if necessary).

BLANK |
ntheta {theta(i), i=1..ntheta} nphi {phi(j), j=1..nphi}

This line may contain anything if *FLUXES*=.TRUE. If *FLUXES*=.FALSE. then:

ntheta Number of polar angles for which results are desired (maximum of 20). [integer]

theta() **Cosines** of the polar (zenith) angles, negative for downward radiances, positive for upward. These must be specified from smallest to largest and must NOT have any zero values. For satellite simulations, this may be the scan angle or the zenith angle. See I*nput Notes, Radiance Angles* for details. [real]

nphi Number of azimuthal angles for which results are desired (maximum of 37). [integer]

phi() The relative (not absolute) azimuthal angles between the sun and the satellite (0-180 degrees), where 0 means forward scattering (looking toward the sun) and 180 means backward scattering (with your back to the sun). [real]

wvstart, wvend* | *bstart bend* | *channel

wvstart, wvend

or

bstart, bend

Specify the starting and ending wavelengths (μm) or wavenumbers (cm^{-1}) [real] if the *IWAVE* option is 1 or 2, respectively. If *IWAVE* is 3, then specify the starting and ending band numbers [integer]. Only one continuous spectral interval may be specified. However, because this is band model, the actual wavelengths/wavenumbers used will probably be somewhat different than what is specified. See *Input Notes: Spectral Bands* and the *Spectral Intervals* section for more information.

If *FLUXES*=.TRUE., then you can obtain both shortwave and longwave results in one run by specifying an interval that crosses the boundary at 4 μm. In that case, the results will NOT be for one band that is both shortwave and longwave; the single specified band will be split into shortwave and longwave components.

If radiance calculations include the band around 3.7 μm, then whether or not thermal emission is included depends on the option *IR106*. Solar radiation will be included unless the solar zenith angle is 90 degrees or greater.

channel If *IWAVE=4* then bandweights are being applied and *channel* specifies the channel number that was used in the bandweights file; *bstart* and *bend* are determined internally from the data in the weights file. See the *Bandweights File* section below.

rsurf nsurfs {whichsurfs(i) surffrac(whichsurfs(i)), i=1..nsurfs} |

surfalb(bstart) ... surfalb(bend) |
whichsurfs(1) |
brdftype {brdfparams(i), i=1..nbparams} |
brdfscaler |
BLANK

Values on this line are **required** even if no shortwave calculations will be done. However, meaningful values are only required if shortwave calculations are to be done.

If *ALBTYPE* = 0:

surfalb()	Surface albedo values for each requested shortwave band from *bstart* to *bend*. Shortwave bands are 106 - 129. For example, if *bstart* is less than 106 and *bend* is greater than 106, enter albedos for bands 106 through *bend*.

If *ALBTYPE* =1, 2, or 3 then:

rsurf	If *ALBTYPE*=1 this value is ignored and the surface albedo in each band is based on the internal data base for each surface type in the scene. If *ALBTYPE*=2 then *rsurf* is the visible albedo or reflectance, which is used to scale the built-in albedos for the surface types present. If *ALBTYPE*=3 then *rsurf* is the broadband (0.3-4 μm) albedo, also used to scale the built-in albedos. Under cloud cover these will be adjusted for diffuse radiation (snow and freshwater only), and should therefore generally be clear sky values. See Input Notes for more details. [real]
nsurfs, *whichsurfs*(), *surffrac*()	Specify the number of surface types in the scene (*nsurfs*), 7 maximum. Then for each of the *nsurf* surface types present, specify its identifier *whichsurfs* [integer] and area fraction *surffrac* [real]. The currently available surface types are:

1 -> Open sea water
2 -> Freshwater
3 -> Melt pond on sea ice
4 -> Melting snow (large grain, 1000 μm with small amount of soot)
5 -> Fresh snow (small grain, 300 μm with small amount of soot)
6 -> Bare sea ice
7 -> Dry sand
8 -> Vegetation (a mean green vegetation surface)
9 -> Grass
10 -> Dry grass
11 -> Deciduous forest
12 -> Coniferous forest.

For example, this line might be:

0.7 2 1 0.3 6 0.7

indicating a surface reflectance of 0.7, two surface types present, the first being open sea water with an area fraction of 0.3 and the second being bare ice with an area fraction of 0.7.

If *ALBTYPE* = 5:

whichsurfs(1) Specify the surface type (one only), as described above and an appropriate BRDF model will be selected for that surface. BRDF models are available for generic vegetation (type 8, actually clover) and fresh snow (5).

If *ALBTYPE* = 6:

brdftype Specify the BRDF model/parameterization to use. Available BRDF models are:
1 -> Hapke: suppy ω, θ, *S*, and *h* for *brdfparams* below
2 -> Rahman: supply ρ, *k*,and Θ for *brdfparams* below
3 -> Roujean: supply *k0, k1*, and *k2* for *brdfparams* below

brdfparams() The parameters of the BRDF model. The number and nature of the parameters depends on the model. See *BRDF Models* in the *Reference Guide* for details.

If *ALBTYPE*=7 then input the BRDF values in a file (option BRDFFILE) and specify:

brdfscaler A scaling factor for the BRDF values (unitless). The input BRDF values will be multiplied by this value so it must be positive and should generally be in the range 0..2. Unless you're experimenting use a value of 1.0. The effect is to scale the overall albedo while maintaining the directional dependence of surface reflectance. However, this method of scaling is not strictly correct from a physics standpoint so use it with caution.

***tsurf emis* |**
***tsurf gemis(bstart) ... gemis(bend))* |**

Values on this line are **required**, even if no longwave calculations will be done. However, meaningful values are only required for longwave calculations; i.e., if only bands less than 106 were specified, or if bands 106 and higher are specified for radiance calculations and *IR106*=.TRUE.

If *EMISSTYPE* = 0 then

tsurf Surface temp, units specified by *ITD*. However, a value of either 999.0 or -999.0 can be used to indicate that the temperature of the profile level nearest the surface should be used for *tsurf*.

gemis() Enter one emissivity value for each of the longwave bands selected (which may include band 106), from *bstart* to *bend*

If *EMISSTYPE* = 4 then

tsurf Surface temp, units specified by *ITD*. However, a value of either 999.0 or -999.0 can be used to indicate that the temperature of the profile level nearest the surface should be used for *tsurf*.

emis Enter one emissivity value to be applied to each and every longwave band.

nclouds ntype(1) cldfrac(1) cldtemp(1) cldtop(1) cldthick(1), cldtau(1), nphases(1), cldphase1(1), cldre1(1), cldwc1(1), cldre2(1), cldwc2(1)

If nclouds > 1 then the following must be given for each additional cloud, <u>one set of values per line</u>, *i=2..nclouds:*

ntype(i) cldfrac(i) cldtemp(i) cldtop(i) cldthick(i), cldtau(i), nphases(i), cldphase1(i), cldre1(i), cldwc1(i), cldre2(i), cldwc2(i)

nclouds — The number of clouds in this scene (e.g., grid cell, pixel, or whatever); 50 maximum. [integer]

Then, if *nclouds* > 0, for each cloud i=1 to *nclouds*:

ntype(i) — The "name" (integer label) of the cloud; used to identify each or the *nclouds* clouds in the cloud overlap record and in $CPRINT and $REPLACE commands. Each name can only be used once in a case/scene. *ntype(i)* is typically 1, 2, ... [integer]

cldfrac(i) — Cloud fraction (0.0-1.0) [real]

cldtemp(i) — Cloud top/bottom temperature, units specified by option *ITD*. *ICTOP* determines if this corresponds to the top or bottom. If this value is either 999 or -999 then the cloud top/bottom temperature is determined from the top height/pressure and the profile. See Input Notes for further information on how clouds are inserted into the profile. [real]

cldtop(i) — Cloud top or bottom height, type and units as specified by *ICTOP* [real]

cldthick(i) — Cloud thickness in either mb or km as specified by *ICTHK*. The value is meaningless for *ICTHK*=3 and 6. If *ICTHK*=4 or 5 then both *cldthick* and *cldtau* are used and either *cldwc1* or *cldwc2* (below) is computed. If ICTHK=6 then the cloud thickness is set to the thickness of the layer that has *cldtop* as its top or bottom. [real]

cldtau(i) — Cloud optical thickness (visible). The value is meaningless for *ICTHK*=1, 2, or 6. If *ICTHK*=4 or 5 then both *cldthick* and *cldtau* are used and either *cldwcwat* or *cldwcice* (below) is computed. If ICTHK=6 then the optical thickness is determined from the layer thickness, water content, and effective radius. [real]

nphases(i) — Number of particle phases that will make up this cloud, currently either one or two. If the cloud is all liquid or all ice of a single habit, *nphases*=1. If the cloud will be mixed phase, either liquid and ice or ice and ice, *nphases*=2. [integer]

cldphase1(i) Cloud particle thermodynamic phase [integer] for the first (and possibly only) of the particle constituents where:

0 -> liquid ("water")
1 -> ice - hexagonal solid columns
2 -> ice - hexagonal hollow columns
3 -> ice - rough aggregates
4 -> ice - bullet rosettes, 4 branches
5 -> ice - bullet rosettes, 2-6 branches
6 -> ice - plates
7 -> ice - dendrites
8 -> ice - spherical particles

Types 1-7 are only available for shortwave calculations.

cldre1(i) Cloud particle effective radius for the first (and possibly only) cloud particle consituent (microns). *cldre1* should be in the range specified in the table below. If it isn't, a warning will be issued though processing will continue. See the *Reference Guide* for information on the relationship between the effective radius and maximum dimension of ice particle habits. [real]

cldwc1(i) The water concentration (or "content") for the first (and possibly only) cloud particle consituent (g m^{-3}). It should be in the range specified in the table below. If *ICTHK*=4 or 5 then *cldwc1* will be calculated. If *ICTHK*=4 or 5 and *nphases*=2 and *cldwc2* has a valid value then *cldwc1* will be calculated/recalculated. [real]

cldphase2(i) Cloud particle phase for the second constituent. Same values as for *cldphase1*. If there is only one particle type in the cloud (e.g., all liquid or a single ice habit, *nphases*=1), this value is not necessary and will be ignored if present.

cldre2(i) Cloud particle effective radius for the second constituent (microns). If there is only one particle type in the cloud (e.g., all liquid or a single ice habit, *nphases*=1), this value is not necessary and will be ignored if present. [real]

cldwc2(i) Water concentration (g m^{-3}) for the second particle constituent, if present. If *ICTHK*=4 or 5 and *nphases*= 2 and *cldwc2* doesn't have a valid value (e.g., <=0 or 999) then it will be calculated. If there is only one particle type in the cloud (e.g., all liquid or a single ice habit, *nphases*=1), this value is not necessary and will be ignored if present. [real]

As an example, this record might be:

```
2 1 0.3 260.2 850.0 999 5.0 1 0 10. 0.2 999 999 999
  2 0.2 999 400.0 999 1.0 1 0 5. 0.01 4 50. 0.03
```

which indicates that there are two clouds in the scene with "names" 1 and 2, having a combined area fraction of 0.5 of the scene, unless they overlap (next record). The cloud "1" is a liquid cloud with optical thickness 5, an effective radius of 10, a liquid water content of 0.2. The physical thickness will be calculated. Cloud "2" is a mixed-phase cloud, liquid and ice, whose temperature will be determined

based on the specified cloud top pressure (400). Its optical thickness is 1, the effective radius for the liquid portion is 5 with a water content of 0.01. The ice portion is made up of bullet-rosettes (4-branch variety) with an effective radius of 50 and a water content of 0.03.

NOTE: If user-specified cloud properties are read (if the option *USERCLOUD*=.TRUE.) then the values for particle phase, effective radius and water content are not used.

BLANK |

nover {ntypeover(i) [overtype(i,j), j=1,novertype(i)] overfrac(i), i=1..nover}

This line specified overlapping cloud sets. See the *Input Notes, Clouds and Cloud Overlap* section for more information. The line is **required** but may contain anything if *nclouds* is less than 2. Otherwise:

nover The number of sets of overlapping clouds (0-50). [integer] There may be up to 50 sets of clouds overlapping in a scene and each set may have up to 50 clouds.

Then, if *nover* > 0, for each set *i*=1 to *nover*:

novertype(i) The number of clouds in overlap set *i*. [integer]

overtype(i,j) The cloud types ("names" in the previous record) of each cloud in overlap set *i*, *j*=1 to *novertype(i)*. [integer]

overfrac(i) The amount of overlap in the set. For example, a cloud fraction of 0.6 for each of two clouds and an overlap fraction of 0.2 would give a completely cloudy scene with 20% of it covered by <u>both</u> clouds and 80% of it covered by each of the two clouds <u>individually</u> (40% each). [real]

As another example, you could set the cloud fraction for a cloud type 1 to 0.9, the cloud fraction for cloud 2 to 0.5, and the overlap fraction to 0.4. That results in 40% of the scene covered by both clouds, 50% by cloud 1 alone, and 10% by cloud 2 alone. For this example the overlap input line would be:

1 2 1 2 0.4

altchoice pchoice tchoice wvchoice ozchoice hzchoice n_in frwv fro3 frrhhz frco2 fro2 frwv2

altchoice, pchoice, tchoice, wvchoice, ozchoice, hzchoice

These variables control whether or not the altitude, pressure, temperature, water vapor, ozone, and aerosol ("haze") profiles are read. [integer]

< 0 -> A value less than 0 (e.g., -1) for ANY of these will cause the profiles from the previous record to be used. In this case NO PROFILE DATA OR COLUMN AMOUNTS WILL BE READ and therefore none should be given.

0 -> Total column amounts rather than the profiles will be read. This option does not apply to pressure or temperature, and if specified for those parameters it will be reset to 2. For altitude it indicates that the height of the first level (surface) will be read and the profile will be computed from the temperature and pressure profiles, which **must** be supplied.

1 -> The profile will be supplied by you.

2 -> Use the standard profile (specified in the options). In this case neither the profile or column amount will be read.

In summary, pressure and temperature can either be read or defaulted to the standard profiles. Altitude can be user specified, standard, or constructed from the user-specified temperature and pressure profiles and the user-specified surface altitude. Water vapor, ozone, and aerosol profiles can either be user-specified, standard, or the standard profile scaled by the total column amounts.

n_in

The number of levels for the input profiles. If none of the **choice* values are 1 then this can be any value (using 0 is a good convention). Otherwise, if *n_in* is positive then the profiles are expected to be specified from the top (highest atmospheric level) down. If *n_in* is negative then the level nearest the surface is expected first. The maximum number of levels is 100 (maximum number of layers is 99). [integer]

frwv, fro3, frrhhz, frco2, fro2, frwv2

Fractional amounts desired of water vapor, ozone, the relative humidity of haze, carbon dioxide, oxygen, and the water vapor continuum (range: 0 to < 100.0). These are multiplicative factors that apply to either the new or old profiles, respectively. When reusing the old profiles (see the $REPLACE statement), however, they will be applied to unscaled profiles; i.e., they do not carry over from adjustments to profiles in previous runs. Use a value of 1.0 if no scaling is desired; use a value of 0.0 to effectively "turn off" an individual gas. [real]

[alt_in(i)] [p_in(i)] [t_in(i)] [wv_in(i)] [oz_in(i)] [aer_in(i)], i=1..n_in

Atmospheric profiles will be read if any of the **choice* variables (above) are set to 1. If not using old profiles, read *nlev* records with all or some of these variables, depending on the **choice* variables in the previous record. Data are free format, one line for each level *i*, all parameters on the same line. Profiles are specified from the top of the atmosphere downward if *nlev* is positive; otherwise they must

be given from the surface up. **Restrictions**: if any of temperature, water vapor, ozone, or aerosol profiles are to be read, then either of pressure or altitude (or both) must be given. If the altitude profile is not given but the first level is (*altchoice*=0; see below), then the temperature profile must also be given.

Definitions:

alt_in()	Altitude profile (units as specified in the options).
p_in()	Pressure. [mb]
t_in()	Temperatures (units as specified in the options).
wv_in()	Water vapor profile (units as specified in the options).
oz_in()	Ozone profile (units as specified in the options).
aer_in()	Aerosol extinction coefficients [1/km], where the value at each level times the thickness of the layer that has this level as its top is the optical depth of the layer. The last (lowest) level is therefore not used. Extinction coefficients are for 0.6 μm.

BLANK |
[zfirst] [h2ocol] [O3] [tauhaze]

This line is required but may contain anything if all the **choice* variables have values other than 0. Otherwise, one or more of these are required in the following order:

zfirst	If the altitude profile was not supplied, give the altitude of the first (nearest the surface) level. (Units as specified in the options.) Applies only if *altchoice*=0 and *pchoice*=1 and *tchoice*=1. The altitude profile will be constructed from the temperature profile.
h2ocol	If the water vapor profile was not supplied, give the total precipitable water [cm]. Applies if *wvchoice*=0.
O3	If the ozone profile was not specified then give the total column amount [Dobson units]. Applies if *ozchoice*=0. (Note: Dobson units = milli atm cm; 1 DU = 2.687e16 molecules/cm^2 = 2.1416667e-6 g/cm^2)
tauhaze	If the aerosol profile was not supplied, then give the total aerosol optical depth at 0.6 um. [unitless] Applies if *hzchoice*=0.

Which ones appear is controlled by the values of the **choice* variables. For example, if *wvchoice* and *hzchoice* are both 0, the line may look like:

1.5 0.2

where *h2ocol*=1.5 and *tauhaze*=0.2.

Minimum and Maximum Effective Radius and Water Content

Habit/Phase	**Min Re**	**Max Re**	**Min WC**	**Max WC**
Liquid	2.50	65.00	5.00e-02	5.00e-01
Solid Column	5.96	84.22	6.33e-04	3.00e-01
Hollow Column	4.97	70.24	5.28e-04	2.50e-01
Aggregate	3.55	108.10	2.20e-04	4.89e-01
Rosette-4	2.77	45.30	1.71e-04	1.16e-01
Rosette-6	2.85	46.01	1.71e-04	1.16e-01
Plate	4.87	48.18	3.90e-04	2.68e-01
Dendrite	0.45	1.88	1.59e-05	7.98e-03
Spheres	5.00	130.00	1.50e-04	7.00e-02

SETTING DEFAULT VALUES

In general, $OPTIONS and $CASE or $SETDATA define all variable values for the model. The subroutine version of *Streamer* (see the *Utilities* section) does not use the input file concept for defining variable values, and not all variables are passed into the subroutine. Therefore, having the appropriate default values is important. Addtionally, it is entirely possible, though not recommended, to run *Streamer* without $OPTIONS or $CASE/$SETDATA commands, in which case the default values would be used.

Default values are defined in one of two ways: if a file named **streamer.def** is present in the directory from which **streamer** is run, or if a defaults file is specified on the command line when **streamer** is run (**-d** option), it is assumed to be a file defining default variable values and it will be read. If it is not present, then values assigned in the file **defaults.f** will be used. The **streamer.def** file has one (and only one) $OPTIONS block and one (and only one) $SETDATA block. If there is anything else in the file, the model might run, which is not the purpose of the defaults value file. See the sample file that came with the distribution. Concerning **defaults.f**, any changes made to that file will require rebuilding the executable code; i.e., recompiling the model.

BAND WEIGHTS FILE

Band weights can be used to approximate an instrument's spectral response functions. This is optional. It is accomplished by specifying a weight for each Streamer band that the response function covers in a separate file. The weight is defined as the proportion of the Streamer band that is covered by the instrument response function [0-1]. Since the height of the band is 1 (100% response), this proportion is a mean response for that band. The computed radiance or flux quantities for each band are multiplied by the weight for that band and the resulting weighted value is summed with those from the other bands specified for a scene.

The structure of the band weights file is as follows: The first line is ignored and may be used for a comment. The subsequent lines are free format and specify the weights for each band within continuous subintervals of the 129 bands, where the channel number [integer], beginning and ending band numbers [integer] of each subinterval are given, followed by the band weights [real]. The channel numbers must be integers in the range [1..MAXCHAN] (currently 1..50). The number of

Streamer bands that a channel can cover is MAXBANDS (currently 25).

If the interval is in the longwave; i.e., it starts in bands 1-106 (NOTE: band 106 includes thermal emission), then the central wavenumber of the subinterval must be provided ON A SEPARATE LINE following the weights. If this value is zero or negative, *Streamer* will use the midpoint of the interval as the central wavenumber for the calculation of brightness temperatures. The central wavenumber is only used if radiances are being computed, but some value must be present in any case [real]. Lastly, the satellite altitude (km) can be given as the only value on the last line in the file [real]. This value is optional. If present, it is assumed that radiance angles are scan angles and it is used to convert them to satellite zenith angles. See *Radiance Angles* in the *Input Notes* section and *Radiances* in the *Reference Guide* for more details.

As an example, here is the contents of the NOAA-12 AVHRR bandweights file. The last line is the nominal altitude of the satellite:

```
Streamer band weights for NOAA 12 AVHRR channels (1-5)
1   116   122
 0.0105 0.0032 0.0048 0.2161 0.9693 0.8516 0.0468
2   115   120
 0.0398 0.6554 0.8205 0.9920 0.4894 0.0042
3   105   107
 0.0126 0.6452 0.0003
 2674.8
4    42    50
 0.0174 0.1959 0.7441 0.8984 0.9670 0.9182 0.2936 0.0295 0.0171
 922.6
5    39    45
 0.0581 0.6387 0.9398 0.9893 0.6060 0.0142 0.0121
 839.3
850.
```

If a band weights file is used, the channel number (*channel*) is used to determine the starting and ending spectral bands for calculations. The bandweights available in the *Streamer* distribution are described in the *Reference Manual*.

CLOUD PROPERTIES FILE

While *Streamer*'s built-in cloud optical property models are ideal for most purposes, there may be applications for which a special set of optical properties are needed. This can be accomplished with the cloud properties input file, which is enabled with the *USERCLOUD* and *CLOUDFILE* option. The file provides the volume extinction coefficient, the single scattering albedo, and either the asymmetry parameter or coefficients of the Legendre polynomial expansion of the scattering phase function.

The first line of the file is ignored and can be used for a comment. The second line has two values: the number of wavelengths for which cloud optical properties are present and the number of Legendre coefficients of the phase function, not including the coefficient for the zeroth moment. If the number of Legendre coefficients is zero then it is assumed that the asymmetry parameter will be given rather than the phase function. The number of Legendre coefficients should be equal to *NCOEF* (see the Options section) and greater than or equal to the number of streams.

The next group of lines describe a particular cloud model; i.e., a set of optical properties for a cloud. The first line can be used for a comment and is ignored. Subsequent lines contain the wavelength (microns), volume extinction coefficient (km^{-1}), single scattering albedo, and either the asymmetry parameter or the Legendre coefficients for each wavelength. The Legendre coefficients must be provided starting with the first moment, not the zeroth moment (which will be set to unity). They are defined by $P(\mu)=SUM\{(2k+1)\ pmom(k)\ pk(\mu)\}$, $k=0,npmom$, where P is the phase function, μ is the cosine of the scattering angle, and pk is the kth Legendre polynomial. You supply *pmom*. For example, for the Henyey-Greenstein function the Legendre coefficients are 1, $3g$, $5g^{-2}$,..., where g is the asymmetry parameter The *pmom* values in your cloud properties file would be g, g^{-2}, ...

NOTE: This block <u>must</u> contain data for the visible (0.6 micron more or less) region. The visible extinction coefficient is used to determine the geometrical thickness from the visible optical thickness.

Additional optical properties data blocks can be provided to define other clouds (maximum of 50). Each will receive the integer "name" that corresponds to their location in the file, and can be referenced by the *ntype* variable in the *Streamer* input file. The first block of cloud data is *ntype*=1, the second is *ntype*=2, etc. The wavelengths must be the same for each cloud, so the number of lines for each must also be the same. The file structure is

<comment line>
nwv npmom
<comment line>
wv(1) ext(1) ssa(1) g(1) | pmom(1,1)...pmom(npmom,1)
...
wv(nwv) ext(nwv) ssa(nwv) g(nwv) | pmom(1,nwv)...pmom(npmom,nwv)
<comment line>
wv(1) ext(1) ssa(1) g(1) | pmom(1,1)...pmom(npmom,1)
...
wv(nwv) ext(nwv) ssa(nwv) g(nwv) | pmom(1,nwv)...pmom(npmom,nwv)

where *nwv* is the number of wavelengths, *npmom* is the number of Legendre coefficients, *wv* is wavelength, *ext* is the extinction coefficient, *g* is the asymmetry parameter, *pmom* contains the Legendre coefficients, "|" means "or", the choice depending on the value of *npmom*. The comment lines can contain anything or nothing, but must be present. Lines 2-5 above are for cloud "1"; lines 6-8 are for cloud "2". Up to ten different cloud sets can be input.

The number of coefficients *npmom* should normally be equal to the NCOEF option value. In any case, exactly NCOEF coefficients will be used. If *npmom* is less than NCOEF then the coefficients from *npmom* to NCOEF will be zero. If *npmom* is greater than NCOEF then your phase function will be truncated.

There is a sample cloud properties files in the **testio** directory of the program distribution.

SURFACE BIDIRECTIONAL REFLECTANCE FILE

Surface bidirectional reflectance can be incorporated in three ways: (1) by selecting a built-in "model" and specifying its parameters, (2) by selecting a built-in model with preset parameters for various surface types, or (3) by providing bidirectional reflectance function (BRDF) values in a file. The

method used is controlled by the *ALBTYPE* option. This section describes the structure of the BRDF file that is read when *ALBTYPE* is 7. The name of the file is given by the *BRDFFILE* option. The BRDF and the anisotropic reflectance factor (ARF) are related as ARF=PI*BRDF/ALBEDO. See the *Bidirectional Reflectance Distribution Functions* section of the *Reference Guide* for further details.

The first line of the file is ignored and can be used for a comment. The second line gives the number of solar zenith angles (maximum of 18) then the angles themselves (degrees). The third line gives the number of view zenith angles (maximum of 18) then the angles themselves (degrees). The fourth line gives the number of relative azimuth angles (maximum of 36) then the angles themselves (0..180 degrees). All angles must be given in monotonically increasing order. The fifth line begins the block of data for the first wavelength. It contains the wavelength in microns. The sixth line contains the first solar zenith angle in degrees. The seventh line begins the block of BRDF values, one line for each view zenith angle and one column for each relative azimuth angle. The file structure is (note: variables names below are not the actual names):

<comment line>
nzen zen(1) zen(2) ... zen(nzen)
ntheta theta(1) theta(2) ... theta(ntheta)
nphi phi(1) phi(2) ... phi(nphi)
wavelength
solar zenith angle
brdf(thetha(1),phi(1)) brdf(theta(1),phi(2))...brdf(theta(1),phi(nphi))
brdf(thetha(2),phi(1)) brdf(theta(2),phi(2))...brdf(theta(2),phi(nphi))
...
brdf(thetha(ntheta),phi(1)) brdf(theta(ntheta),phi(2))...brdf(theta(ntheta),phi(nphi))

Provide BRDF values for every solar zenith angle (lines from "solar zenith angle" through "brdf(thetha(ntheta,phi(1))...") and provide similar blocks of data for each wavelength. The data will be read until the end of the file so the number of wavelengths is not specified. Wavelengths must be in monotonically increasing order.

Note: This method of utilizing BRDFs can be computationally intensive, i.e., slow!

SPECTRAL INTERVALS

These are the starting wavenumbers of each interval. The end of each band is the beginning of the next.

Wavenumbers (cm^{-1})

Longwave:

Band	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
1:	20	40	60	80	100	120	140	160	180	200
11:	220	240	260	280	300	320	340	360	380	400
21:	420	440	460	480	500	520	540	560	580	600
31:	620	640	660	680	700	720	740	760	780	800
41:	820	840	860	880	900	920	940	960	980	1000
51:	1020	1040	1060	1080	1100	1120	1140	1160	1180	1200
61:	1220	1240	1260	1280	1300	1320	1340	1360	1380	1400
71:	1420	1440	1460	1480	1500	1520	1540	1560	1580	1600
81:	1620	1640	1660	1680	1700	1720	1740	1760	1780	1800
91:	1820	1840	1860	1880	1900	1920	1940	1960	1980	2000
101:	2080	2160	2240	2320	2400	*				

*(Band 105 ends at 2480)

Shortwave:

Band	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
106:	2500	2920	3440	4200	4700	6080	6520	7840	8400	9120
116:	10000	11540	12820	13300	14500	15620	17540	19240	20840	22720
126:	25000	27780	30300	33340	*					

*(Band 129 ends at 35710)

Microns (10000/wavenumber)

Longwave:

Band	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
1:	500.00	250.00	166.67	125.00	100.0	83.33	71.43	62.50	55.56	50.00
11:	45.45	41.67	38.46	35.71	33.33	31.25	29.41	27.78	26.32	25.00
21:	23.81	22.73	21.74	20.83	20.00	19.23	18.52	17.86	17.24	16.67
31:	16.13	15.62	15.15	14.71	14.29	13.89	13.51	13.16	12.82	12.50
41:	12.20	11.90	11.63	11.36	11.11	10.87	10.64	10.42	10.20	10.00
51:	9.80	9.62	9.43	9.26	9.09	8.93	8.77	8.62	8.47	8.33
61:	8.20	8.06	7.94	7.81	7.69	7.58	7.46	7.35	7.25	7.14
71:	7.04	6.94	6.85	6.76	6.67	6.58	6.49	6.41	6.33	6.25
81:	6.17	6.10	6.02	5.95	5.88	5.81	5.75	5.68	5.62	5.56
91:	5.49	5.43	5.38	5.32	5.26	5.21	5.15	5.10	5.05	5.00
101:	4.81	4.63	4.46	4.31	4.17	*				

*(Band 105 ends at 4.03)

Shortwave:

Band	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
106:	4.00	3.42	2.91	2.38	2.13	1.64	1.53	1.28	1.19	1.10
116:	1.00	0.87	0.78	0.75	0.69	0.64	0.57	0.52	0.48	0.44
126:	0.40	0.36	0.33	0.30	*					

*(Band 129 ends at 0.28)

INPUT NOTES

This section provides additional detail on the input variables.

COMPLEX SCENES

A "scene" can consist of clear sky, individual clouds, overlapping sets of clouds, and one or more surface types. It must be stressed that the treatment of such complex scenes is really very simple:

- The flux/radiance that *Streamer* computes for the scene is the sum of the fluxes/radiances computed for each cloud part (individual clouds and overlap) weighted by their area fraction.
- The different surface types that occur within a scene do not occupy different portions of the scene; i.e., their reflectance signatures are used to create a weighted average albedo for the entire scene.

The point is that no 3-dimensional effects are considered, and exactly which portion of the scene each cloud, overlapping cloud set, and surface type occupy is irrelevant.

SPECTRAL BANDS

When specifying the spectral range as wavelengths or wavenumbers (*wvstart, wvend*), the model will use those exact values only if the correspond exactly to the beginning and end of *Streamer*'s spectral bands. Otherwise, the wavelengths/wavenumbers used will be the be start of the band in which *wvstart* occurs, and the end of the band in which *wvend* occurs. If *wvend* is on the boundary between two bands, then the lower band number is used.

The interval 2480-2500 cm-1 (4.0-4.03 μm) is not included.

It is assumed that bands 1-105 have no solar component and that bands 107-129 have no thermal component. Only band 106 can have both, and only when radiances are being computed. Shortwave calculations are not done when the solar zenith angle is 90 degrees or more.

IMPORTANT!

The solar constant is always adjusted for Earth-Sun distance based on the date specified.

FLUX AND RADIANCE CALCULATIONS

In general, the greater the number of streams, the greater the accuracy of the computed radiances (intensities) and irradiances (fluxes) in the shortwave in particular. However, more streams requires more computing time. If two streams are specified then a 2-stream solver is used (Toon et al., 1989) and only fluxes can be computed. If more than two streams are specified then the discrete ordinate solver DISORT (Stamnes et al., 1988; Stamnes et al., 1994) is used.

With DISORT a balance can be found between accuracy and computation time by adjusting the parameters *NSTRSHORT, NSTRLONG,* and *NCOEF*. The Nakajima and Tanaka (1988) algorithm incorporated into DISORT version 2 permits a high degree of accuracy in shortwave, forward-scattering problems with as few streams as 8. However, it is recommended that *NCOEF* remain large; at least 24. Unfortunately, this increases the computation time. Even so, the time savings using fewer

streams outweigh the increase in computation time with larger *NCOEF*s, and the decrease in accuracy with a small *NCOEF* may not be acceptable. For example, comparable radiances can be obtained using *NSTRSHORT*=16 and *NSTRSHORT*=8 with *NCOEF*=24. However, the latter case runs 5 times as fast as the former. *NCOEF* is only meaningfull when *NSTRSHORT* and/or *NSTRLONG* are greater than two.

.

CAUTION

The cloud phase function is approximated using the asymmetry parameter and the Henyey-Greenstein function. This is fine for fluxes but can result in significant errors when radiances are being computed.

If radiance calculations are to be done over a range of angles, specifying all the angles with *ntheta*, *theta*, *nphi*, and *phi* in the input data block will be 2-20 times faster than looping over the angles with $REPLACE, depending on the number of angles. This is a result of the structure of DISORT, and not due to any inefficiency in $REPLACE.

RADIANCE ANGLES

The angle *phi* is the **relative** angle between the sun and the satellite;i.e., if you project a line onto the surface between the sun and the surface point observed, and another line between the satellite and the point observed, *phi* is 180 minus the angle between them. So *phi*=0 means that the satellite is looking into the sun (forward scattering) and *phi*=180 means that the satellite is looking away from the sun (backward scattering). Perhaps a better way to think about it is that *phi* is the angle from the forward scattering direction. This is a common convention not unique to *Streamer*, though the opposite is equally common!

The angle *theta* is the cosine of the polar (zenith) angle for which radiance computations are to be done. In degrees, 0 means straight up and 180 is straight down, which gives *theta* values of 1 and -1, respectively.

For satellite simulations, the angle of the emerging radiance, *theta*, needs to be adjusted. In a plane-parallel sense, this is the same as the satellite scan angle (or "look" angle or "view" angle or whatever you want to call it). It is close to, but not the same as, the scan angle for a **curved** earth, which is a function of the satellite altitude and the radius of the earth. With the NOAA polar orbiters, for example, which have a nominal height above the surface of 850 km, a scan angle of 10 degrees implies a *theta* of 11.4, while a scan angle of 50 degrees implies a *theta* of 60.3.

When you specify values for *theta* and you're doing satellite simulations, are your values for the satellite zenith angle or the scan angle? If a bandweights file is used and if the satellite altitude is given in the file then *Streamer* assumes that your *theta* values are the scan angles. It will then do the conversion automatically; i.e., the scan angles will be converted to zenith angles for the plane parallel calculations. If there is no bandweights file or if the bandweights file does not contain a satellite altitude, then the *theta* values are assumed to be satellite zenith angles and no conversion is done. The conversion equations are given in the *Flux and Radiance Quantities* section of the *Reference Manual*.

SURFACE TYPES AND ALBEDO

The following discussion pertains to the cases where a surface bidirectional reflectance distribution

function (BRDF) is **not** used. See *Input/Surface Bidirectional Reflectance File* for information on specifying BRDFs.

Surface types may be any mixture of snow, ice, meltponds, open (sea) water, vegetation, and sand. The type of snow is determined by the month:October through May - Snow is fresh; grain size=300 μm; soot=0.3e-6 ppmw; June through September - Snow is old; grain size=1000 μm; soot=0.3e-6 ppmw.

The albedo scheme is as follows. If *ALBTYPE*=0, then you provide the albedo in each band and the surface type fractions are not used. Otherwise, the built-in albedo values for each surface type will be weighted by the area fraction that you specify. If *ALBTYPE*=1 then these built-in values are the ones used in the radiative transfer calculations. If *ALBTYPE*=2 then these are scaled by the "observed" (input) 0.6 μm reflectance (*rsurf*). If *ALBTYPE*=3 then the built-in albedos are scaled by the "observed" broadband albedo (*rsurf*). The scaling is done so that the built-in 0.6 μm or broadband albedo is equal to the observed albedo; i.e., the shape of the curve over the shortwave bands is maintained, but it is scaled (by the ratio of built-in to observed albedo) up or down. The assumption is, therefore, that the modeled (built-in) albedos are incorrect in their amplitude (and that your observed value is correct) but that the band-to-band relationships are correct. Be aware that it is entirely possible (and easy) to cause the adjusted albedos to exceed unity. This will happen if your "observed" albedo is much larger than the built-in albedo.

The visible surface reflectance value, if specified, should correspond to the *Streamer* band that contains 0.6 μm. We refer to this here as simply “0.6 μm” or "visible".

For sand reflectance data the values in bands 126-129 are simply copies of the band 125 reflectance. The values in bands 106-108 are copies of the value for band 109. This is only an approximation, necessary because data for these wavelengths were not available.

Under clouds with optical depths greater than 0.1 (arbitrary) the built-in albedo values for diffuse sky conditions are used when ALBTYPE = 1, 2, or 3.

While different surface types can occur within a scene, they do not occupy different portions of the scene; i.e., their reflectance signatures are used to create a weighted average albedo for the entire scene.

AEROSOLS

Aerosols are specified in terms of their optical property model and their vertical distribution. In the simplest case these two parameters are chosen in the $OPTIONS section of the input file and *hazechoice* is set to 2 in the input data section. The total optical depth is then computed from the built-in data (see *Input: Options* and also the *Part II: Reference*). Two other possibilities exist, however. If one of the built-in profile shapes suits your needs but you want a different total optical depth, then use the *tauhaze* input variable and *hazechoice=0*. If you also want to control the profile shape, then an extinction profile can be input with *hazechoice=1*.

CLOUDS AND CLOUD OVERLAP

The cloud fraction specified for each cloud includes that part that is overlapping with another cloud or clouds, if any, so that the non-overlapping fraction of any given cloud is its total fraction (*cldfrac*) minus the sum of all its overlapping fractions. For example, suppose you have defined cloud types 1,

3, and 4 and have specified the following cloud fractions for each: 0.5, 0.3, 0.6. Clouds 1 and 3 overlap is 0.2 and clouds 1 and 4 overlap is 0.3. Therefore, none of cloud 1 alone exists in the scene; the non-overlapping part of cloud 3 is 0.1, and the non-overlapping part of cloud 4 is 0.3. The total cloud coverage in the scene is 0.9. Overlapping clouds cannot occupy the same layer. "Layer" here is not necessarily the same as a layer in the profile since a cloud may span more than one profile layer.

Cloud thickness works as follows: if cloud optical thickness is input then cloud physical (or "geometrical") thickness (in both mb and km) is computed using the specified effective radius and liquid water concentration and the appropriate parameterization of cloud optical properties. If cloud physical thickness is given then cloud optical thickness is computed in the same manner. If, however, either the cloud bottom would extend below the surface or the top would extend above the profile top, a warning is issued, the bottom/top is reset to the profile bottom/top, and thicknesses (including optical) are recomputed. The cloud optical thickness should be a value in the visible (0.6 μm) portion of the spectrum.

If both the optical thickness and the geometrical thickness are specified (*ICTHK*=4 or 5) then the cloud water content will be calculated. However, for mixed-phase clouds (*nphases*=2), specifying both the optical and geometrical thicknesses also requires that one of the two water contents be specified (*cldwc1* or *cldwc2*) so the other can be calculated. So if *ICTHK*=4 or 5 and *nphases*=2, then the first water content will be calculated if the second content is greater than zero but not 999, or the second water content will be calculated if the first water content is greater than zero but not 999. If both are greater than zero then the first water content will be calculated (arbitrary decision). WARNING: In the case of mixed-phase clouds with one water content specified, it is entirely possible for the two thicknesses to be inconsistent with the water content such that the calculated water content is zero or negative. In that case a fatal error is issued. Water content is not calculated if user-specifed cloud optical properties have been read (if USERCLOUD=.TRUE.).

The last cloud thickness option (ICTHK=6) is to not give the thickness at all (though a value is still required), impling that the cloud should occupy the layer that has the specified cloud top/bottom pressure/altitude as one of its boundaries. The cloud thickness is calculated from the top/bottom pressure/altitude to the next level. In general the top/bottom should correspond to a level in the profile but this is not required.

.

CAUTION

The conversion of optical to physical thickness can be problematic.

If cloud thickness is specified as an optical thickness, *Streamer* requires that you provide a **visible** optical thickness. The conversion of optical thickness to physical (geometrical) thickness, and vice versa, therefore requires that the visible optical properties be used. For longwave calculations with ice clouds this may pose a problem. At present there is only one ice particle type in the longwave (spherical) but 7 in the shortwave. If you specify a particle type other than spherical, and if you are prescribing optical rather than physical thickness, and if you are doing both shortwave and longwave calculations, then that particle type will be used in determining the physical thickness. So there is something of a disconnect in that the longwave radiation calculations are based on a spherical particle but the cloud physical thickness is based on some other particle shape. Specifying a spherical particle

only partially solves the problem, because the shortwave optical properties for spherical ice particles may not be realistic. If only longwave calculations are being done, then only spherical particles are allowed. Until longwave optical property models for other ice particle types are developed, there is no good solution to this problem. Note that this only affects longwave calculations.

When a cloud is inserted into a profile the atmospheric values at the cloud top and bottom of the cloud (temp, pressure, ozone, etc.) replace those of the nearest levels in the original profile. The nearest level is determined by the cloud top pressure or altitude. These values will be essentially the same (there may be minor differences due to interpolation) regardless of whether the cloud top or bottom was specified in the input data, unless a cloud top temperature is given that is significantly different from the temperature in the profile at the specified cloud top/bottom pressure or altitude. Exceptions to the nearest level rule occur when the cloud is thinner than a layer. If a thin cloud (less than one layer in thickness) straddles a level and both the top and bottom are closer to the straddled level than to the ones below and above, then if the cloud top was specified it gets the index of the level that's straddled and the cloud bottom gets the next lower level index. If the cloud bottom was specified then it gets the index of the level that's straddled and the cloud top gets the next higher one. Therefore, in this case (and only this case) the placement of the cloud depends on whether the cloud top or bottom was specified. If a thin cloud is completely within a layer then the cloud bottom gets the index of the bottom of the layer and the cloud top gets the index of the top of the layer, as expected. Figure 3-1 gives a simple example of inserting a cloud into a profile.

Note: While the relative humidity within clouds is usually near saturation, it is not reset in the model. In other words, the relative humidity of the profile is not changed, except for interpolation as described above.

With overlapping clouds placement can be tricky. Clouds are inserted into the profile one at a time without regard to the levels of any previously inserted cloud. In other words, there is nothing stopping the placement routine from changing the bottom of the cloud first inserted to some new set of values matching the cloud currently being inserted. Make sure there are an adequate number of levels in the profile.

If you want to have two or more adjacent (no vertical space between them) clouds then you should (1) make sure that your profile has an adequate number of levels, preferably at points very close to the cloud tops and bottoms, and (2) specify cloud thickness in mb or km rather than using optical thickness, making sure that the tops and bottoms coincide exactly with levels in the profile. If you don’t want the cloud insertion to change the temperature profile then use 999 for the cloud top temperature. That way the cloud top will be located based on its pressure/altitude and the temperature from the profile at that level will be used for the cloud top.

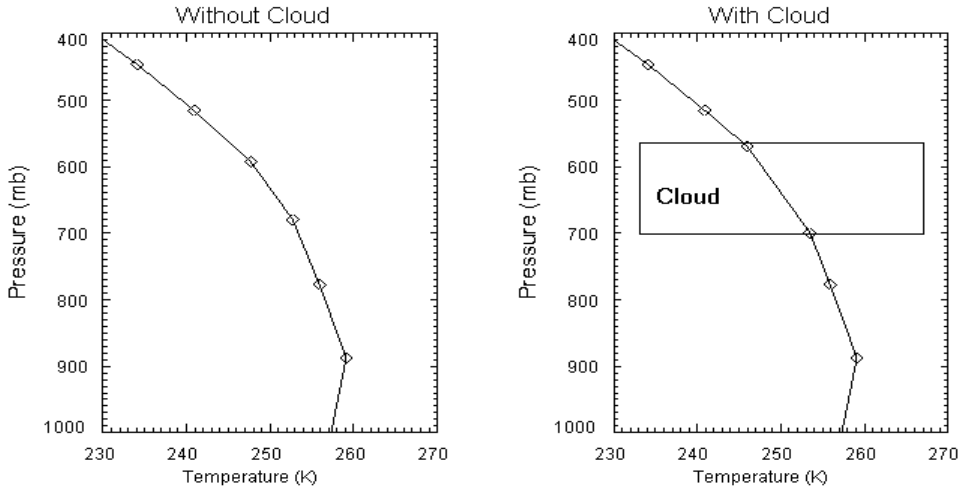


Fig. 3-1. A simple example of a temperature profile with and without a cloud. The cloud was specified by it's cloud top pressure and thickness, and it's top temperature is determined from the profile. The number of profile levels is the same in both cases.

IMPORTANT!

If cloud top temperature is specified, make sure that it is reasonably close to the profile temperature at the specified vertical position. With overlapping clouds, insuring that the top of one cloud is not higher than the bottom of the cloud above it is the user's responsibility.

BAND 106: 3.4 - 4.0 μm

This band has created a number of programming problems, since both reflected and emitted energy are possible. For flux calculations it is considered a shortwave band, but for radiances the user may want to examine only the reflected solar radiation, only the emitted thermal component (at night), or both. Therefore, either or both of the surface albedo and emissivity will be required. Each of these should equal one minus the other, but if the internal albedo models are used then the albedo may not correspond to the desired emissivity. So, to simplify things a bit, if either solar only or solar plus thermal radiation are desired in band 106, then albedo will be used in the calculations, radiances or fluxes. However, if thermal radiation is included for radiance calculations, i.e., if IR106=.TRUE. or if the solar zenith angle is at least 90 degrees, then the emissivity must also be input.

PROFILES

The profiles define the layers used in the model. Profiles are specified from either the TOA down to the surface or the surface to TOA, depending on the sign of *nlev*. The temperature difference across a layer should be less than 10 K, although this restriction will be lifted when DISORT version 2 is publicly released.

The maximum number of levels is currently 100. If *SPACE*=.TRUE., then some number of levels will probably be added. The total number of levels is the number of user levels plus the number added from the standard profile. See the standard subarctic profile levels in the reference guide to estimate your total number of levels when using *SPACE*=.TRUE..

In either case of *SPACE*, total precipitable water, aerosol amount, and total column ozone (if provided) correspond to the specified atmosphere. For example, if your profile extends from 0-20 km, *SPACE*=.FALSE. and no water vapor profile is provided, then specify total precipitable water for 0-20 km only. If *SPACE*=.TRUE. then your column amounts will be distributed over the entire profile from your surface level to the top level set by *Streamer*.

The scaling factor *frrhhz* is used to scale the relative humidity, but this only plays a role in selecting the aerosol optical properties. It does not change the total column water vapor amount; use *frwv* to scale that. In a sense this goes against the conservation of mass, because the relative humidity profile used in determining aerosol optical properties can be different from the actual relative humidity profile. But it allows the user to specify aerosols as being more or less hygroscopic, which can be useful.

4 OTHER COMMANDS

THE $REPLACE AND $EXCHANGE COMMANDS

These commands are used to replace input values from the previous case, and to loop over a range of values for one or more variables (e.g., cloud optical depth or surface temperature). Replacements done with $REPLACE are in effect until changed again further along in the input stream, while those done with $EXCHANGE are only in effect during the execution of that statement. In other words, any value assignments made in an $EXCHANGE are forgotten after $EXCHANGE is executed, and variable values resort to their previous state. Variable values that are in place when $REPLACE or $EXCHANGE are encountered are those from the last $CASE or $SETDATA, modified by any $REPLACEs that were done. So normally $REPLACE and $EXCHANGE will not be used until after a complete set of data values has been defined with $OPTIONS and either $CASE or $SETDATA. However, you can also define a set of default data by modifying the subroutine **set_defaults** (**defaults.f**) and recompiling the code. Then and only then is it reasonable to use $REPLACE to assign a partial list of variable values without a preceding $CASE or $SETDATA.

The form of these two commands are identical; the following examples use $REPLACE. The line begins with "$REPLACE", either upper or lower case. "$REPLACE" is followed by a space then one or more replacements separated by semicolons or commas. The format of the line is:

$replace *var1*=*val1*; *vari*=*vali* ...

where *var* is an identifier (variable name; see the list below) of either a scalar; e.g., *ntheta* or an array. The value that *var* is to be assigned is *val*, which is either a scalar (single value), a list of array values, or a loop structure. Variable value assignments must be made using "=", and must be separated by either a semicolon or comma (";" or ","):

$replace zen = 60.; lat=72.3, lon=0.0

No line may be more than 240 characters (only 240 will be read), including the string "$REPLACE". However, placing a backslash ("\") at the end of the text in a line signifies a continuation onto the following line in the file. The continuation character is **not** a valid separator between replacements; i.e., ";" or "," must precede it. Up to 10 continuation lines can be used. There is no limit to the number of consecutive $REPLACE or $EXCHANGE statements in the input file.

ARRAYS

Array elements can be assigned by either specifying an array subscript (one-dimensional only) or by specifying a list of values enclosed in braces. Array indices must be integer constants (e.g., 1, 2, 3, ...); no variable names can be used. For example,

$replace cldthick(1) = 5.

assigns the value 5.0 to the first element of the array **cldthick**. To replace elements in an array starting with the first, but without having to specify an assignment for each subscript value, use the variable name, and equals sign, then a list of values **separated by commas** (not semicolons) and enclosed in

curly braces. For example, if *nphi*=4 and you want to assign new values to all the elements of phi, you could do so with:

$replace phi = {10, 30, 60, 90}

which is equivalent to

$replace phi(1)=10; phi(2)=30, phi(3)=60; phi(4)=90

If the number of values in the list is less than the number expected (*nphi* in this case), the remaining elements will be unchanged. Such an underloaded list is valid and useful. Note that all value assignments start with the first element of the array. The list is **not** checked for overloading, so if there are more values than the array should contain, the results are unpredictable.

For the surface fraction array (*surffrac*) and the cloud properties arrays (*cldthick*, *cldfrac*, *cldtemp*, *cldtop*) the index that is used is the "name" of the surface or cloud type, not the position of the surface or cloud in the list 1..*nsurfs* or 1..*nclouds*. For example, if there are two surfaces (*nsurf*=2), of types 2 and 5 (*whichsurfs*(1)=2, *whichsurfs*(2)=5), then you would change the fraction of the second one with *surffrac*(5)=x, not with *surffrac*(2)=x. Similarly, if there are three clouds (*nclouds*=3) with types 2, 3, and 1 (*ntype*(1)=2, *ntype*(2)=3, *ntype*(3)=1), and you want to change the thickness of the second one, you would do so with *cldthick*(3)=x, not *cldthick*(2)=x.

LOOPS

Loop structures can take either of two forms:

***var* = (*start_value*, *end_value*, *increment*)**

or

***var* = [*val1*, *val2*,..., *valn*]**

The first case is similar to the loop structure found in most programming languages, though the increment value in a loop must be specified; there is no default. The increment value may be positive or negative, integer or real. The second form specifies explicitly all values that the variable is to be assigned. It is equivalent to a series of $REPLACE lines, each reassigning *var1* once; e.g.,

$REPLACE *var1*=*val1*
$REPLACE *var1*=*val2*
...
$REPLACE *var1*=*valn*

Note the syntax differences between these two loop forms. In the first form three and only three values are expected. In the second form any number of values may be present (maximum: 100). In both cases the values must be **separated by commas** (not semicolons).

If two or more loops are specified, then they are nested, with the rightmost loop varying most rapidly. For example:

$replace zen=65.0; cldre(1)=(2, 10., 1.); cldthick(1)=[0,5,10,100]

This line replaces the zenith angle specified in the previous case by 65.0, and loops over both cloud thickness and cloud effective radius for cloud number 1 (both **cldre** and **cldthick** are arrays), varying thickness more rapidly since it occurred second (as in a nested loop structure); i.e., for every value of **cldre**, **cldthick** takes on values of 0, 5, 10, and 100. Up to 10 loops are allowed in a single $REPLACE.

Lastly, it is possible to execute two or more loops in parallel, assuming that they each have the same number of values. This is accomplished by enclosing the loops in vertical bars "|". For example,

$replace tsurf=280.0; |cldre(1)=[7,12,20]; cldthick(1)=[5,10,100];| zen=[50.,60.,70.]

In this example, **cldre** and **cldthick** will change together (they are not nested loops), first with value pairs of (7,5), then (12,10), then (20,100). But for each of these pairs of values, **zen** will loop through its three values of 50, 60, and 70 degrees. **tsurf** will have the value 280 throughout. The first "|" must immediately precede a variable name; the second "|" can be either before or after the comma or semicolon separator. *Only one parallel loop structure is allowed in a $REPLACE*.

In summary, here are the special symbols and their meanings:

{***val1, val2,..., valn***}	Assignment of array values
(***start_value, end_value, increment***)	An incremental loop
[***val1, val2,..., valn***]	A loop listing
| (...); [...]; ...|	A parallel array structure

REPLACEABLE VARIABLES

Only the following variables can be reassigned via $REPLACE:

year, month, day, hour, lat, lon, zen,
ntheta, theta(1..ntheta), nphi, phi(1..nphi),
bstart, bend, wvstart, wvend, channel
rsurf, nsurfs, whichsurfs(i), surffrac(whichsurfs(i)); for *i* in the range *1..nsurfs*
brdftype, brdfparams(), brdfscaler, surfalb(bstart)...surfalb(bend)
tsurf, emis, gemis(bstart)...gemis(bend)
nclouds, cldfrac(ntype(i)), cldtemp(ntype(i)), cldtop(ntype(i)), cldthick(ntype(i)), cldtau(ntype(i)),
nphases(ntype(i)), cldphase1(ntype(i)), cldre1(ntype(i)), cldwc1(ntype(i)), cldphase2(ntype(i)),
cldre2(ntype(i)), cldwc2(ntype(i)); for i in the range 1..nclouds
frwv, fro3, frrhhz, frco2, fro2, frwv2
*zfirst, h2ocol, o3, tauhaze (*only if the **choice* variables are appropriately set*)*

and these options:

NSTRSHORT, NSTRLONG, NCOEF, STDPROF, AERMOD, AERVERT, ALBTYPE,
IZD, ITD, IWV, IO3, ICTOP, ICTHK, IWAVE, IR106, GASABS, RAYLISHRT, SPACE,
DESCRIP, USEROUT, descripfile, userfile

For the logical variables *IR106, GASABS, RAYLISHRT, DESCRIP, USEROUT,* and *SPACE* use .TRUE. and .FALSE. (case insensitive) in $REPLACE statements. Character variable values (e.g., *descripfile*, *userfile*) must be enclosed in single or double quotes.

CAUTION

Beware of dependencies.

This caution is particularly true for *bstart, bend, and channel*, which may require changes in *tsurf*, *surfalb*, *gemis*, and *IR106*. For example, in a $CASE block *bstart* and *bend* were 106 and 129, respectively, and you specified "missing" values for surface temperature or emissivity thinking that they wouldn't be used. But if in a subsequent $REPLACE *bstart* and *bend* are changed to 1 and 105, then there will not be a valid surface temperature or emissivity for the calculations. So, if albedo, surface temperature, and emissivity will be needed anywhere in the input file, be sure that valid values are input with $CASE or $SETDATA.

THE $CPRINT COMMAND

The $CPRINT command (for "case print") is used to print one or more variables after each case is processed with $CASE, $EXCHANGE, or $REPLACE. In the $CPRINT command variables are separated by "," or ";" as in $REPLACE. Up to ten continuation lines are allowed. Scalar and array variables can be specified, but array indices, if present, must be integer constants; i.e., variable names as indices are not allowed. Strings can also be printed but are limited to 40 characters including enclosing single or double quotes (required). No quotes can appear within the string. No arithmetic expressions are allowed with $CPRINT. To use $CPRINT you should set the *USEROUT* and *USERFILE* options appropriately.

Here's an example:

```
$CPRINT 'bstart, bend: ', bstart, bend, /NEWLINE, \
'Re, albedo, zenith angle: ', cldre, satalb(1,2), zen
```

All variables in a $CPRINT will be output in a single record, up to 500 chars, with two exceptions:

1. If array indices (subscripts) are not given then the entire active part of the array is printed. If it is a one-dimensional array then it is printed as part of the current record. If it is a two- or three-dimensional array then it is written one line per level if the vertical level is one of the dimensions or one line per polar angle if for radiance-related quantities (*averad, satalb, sattb*).
2. The /NEWLINE option (which may be upper or lower case but must start with a slash) can be used like a variable name to force a carriage return in the output.

Default formats for each variable will be used, and all values will be separated by at least one space. The option /TAB can be used to add five spaces between variable values. "/NEWLINE" and "/TAB" can be shortened to "/N" and "/T" if desired. They all can be uppercase or lowercase.

The variables that can be output using $CPRINT are listed in the section "User-Controlled Output Variable Descriptions" (page 53, variables without "*"). For array variables that have the vertical level as one of their subscripts (e.g., *cldfrc, averad*), there is one special subscript form. Since the number of levels may not be known a priori (if *SPACE*=.TRUE.) you can use a negative subscript for the level,

where -1 means the surface, -2 means the next level up from the surface, etc. So,

$CPRINT averad(1,1,1), averad(1,-1,1)

prints the average scene radiance at the first polar and azimuthal angles that you specified (first and second indices), but first for the top atmospheric level and then for the surface.

All output from $CPRINT goes to *USERFILE*. Therefore, output using the user-changeable subroutine in **writeusr.f** and output from $CPRINT cannot both occur in the same run. However, if *USEROUT* is set to true in the options then the **writeusr** subroutine will be called for each case processed until a $CPRINT is encountered. So, if you intend to use $CPRINT to control your output then it makes sense to place it in the input stream before the first $CASE, $REPLACE, or $EXCHANGE. If you use more than one $CPRINT in the input stream the one most recently encountered is the one used.

If *USEROUT* was not set to true in the options (and therefore no *USERFILE* was opened) and $CPRINT is encountered, all output from $CPRINT will go to the default file **USER.OUT**. As with user-customizable output using **writeusr.f**, the use of $CPRINT does not preclude the use of the *DESCRIP* option.

Also see the $PRINT command below.

THE $COMMENT COMMAND

$COMMENT isn't really a command in that no action is taken. When encountered in an input file, the line is simply ignored and the next command is read. Anything can appear on this line. Only one line of text is allowed per $COMMENT; i.e., no continuation lines are allowed. If your text exceeds one line, use $COMMENT at the beginning of each. The semicolon (";") can be used in place of the string "$COMMENT" and must be the first character on the line.

Comment lines can be placed wherever a command (e.g., $CASE, $REPLACE) is expected. They cannot interrupt a $CASE, $SETDATA, or $OPTIONS data block. However, comments, with or without a leading semicolon, can be placed after all expected data on a line since all characters after the data are ignored. But if you extend the data "line" to multiple lines, e.g., if you have five clouds and specify their properties on separate lines, do not put comments within this data block.

$COMMENT differs from the *rectitle* of $CASE and $SETDATA in that comments are not sent to any output file.

Blank lines can also be used wherever a comment is allowed, which is wherever a command is expected.

INTERACTIVE MODE: $ASSIGN, $LOAD, $PRINT, $RUN, AND $EXIT

All of the above commands are available in interactive mode (invoked from the command line with "streamer -i"), plus a few more. The $ASSIGN command can be used to assign values to variables. It is like $REPLACE and $EXCHANGE except that (a) no loops are allowed and (b) the model is not executed after the assignments. If a loop structure is present, a fatal error will be issued. Like

$REPLACE the assignments are permanent. The syntax is identical to that of $REPLACE. For convenience, the actual command string "$ASSIGN" or "ASSIGN" is not required; i.e., the line(s) only needs to contain assignments. For example, the following two statements are equivalent:

$assign tsurf=280.0, zen=50.
tsurf=280.0, zen=50.

$LOAD is used to load a set of default values. Its syntax is "load <file name>" where <file name> is the name of a file containing only $OPTIONS and $SETDATA or $CASE blocks. Any other commands in the file will be ignored. $CASE and $SETDATA are treated equally, as $LOAD does not cause the model to run. Using $LOAD as the first command in interactive mode is equivalent to starting *Streamer* with a defaults file specified on the command line, i.e., "streamer -i -d <file name>". However, $LOAD can be used at any time in the session.

$PRINT has the same syntax and capabilities as $CPRINT, but $PRINT results in the immediate display of its arguments whereas $CPRINT only produces output after each case is processed (the "C" in $CPRINT is for "case"). If $CPRINT is used in interactive mode then output is sent to both a file and the screen for each case. If $PRINT is used in non-interactive mode then it will print to the screen the values of its arguments at that point in the processing. $PRINT and $CPRINT operate independently.

$RUN causes the model to run. Use it after making assignments with $ASSIGN and/or loading data with $LOAD.

$EXIT (or $QUIT) is used to terminate an interactive session. $HELP is another interactive command, but the information provided is minimal.

All of the above commands are permitted in an input file, though they are of limited utility there and were implemented for use in interactive mode. Note that $SETDATA followed by $RUN is equivalent to a $CASE statement without $RUN; $ASSIGN followed by $RUN is equivalent to $REPLACE.

Here's an example of an interactive session:

```
stratus % streamer -i

 Streamer, v3.0

 No 'streamer.def' defaults file.  Using internal default values.
streamer> load ../testio/streamer.def
streamer> print zen, bstart, bend
  -99.00   1 129
streamer> zen = 50.
streamer> cprint zen, swflxd(-1)
 WARNING >>> $PRINT ENCOUNTERED BUT NO FILE OPEN.  USING USER.OUT
streamer> run
 Processing case       1
   50.00  548.70
streamer> exit
 Streamer done.
```

5 INPUT QUICK REFERENCE

This section lists the input variables in their proper order. These variables are described completely in the previous sections. Each line below corresponds to a line of input. "BLANK" signifies a blank line.

Options:	$OPTIONS FLUXES IR106 CLDFORCE NSTRSHORT, NSTRLONG NCOEF GASABS RAYLISHRT ALBTYPE EMISSTYPE STDPROF, SPACE AERMOD, AERVERT IZD, ITD, IWV, IO3, ICTOP, ICTHK,IWAVE OUTLEVS DESCRIP DESCRIPFILE or BLANK if DESCRIP=.FALSE. USEROUT USERFILE or BLANK if USEROUT=.FALSE. WEIGHTFILE or BLANK if IWAVE<4 USERCLOUD CLOUDFILE or BLANK if USERCLOUD=.FALSE. BRDFFILE or BLANK if BRDFFILE=.FALSE.
$CPRINT (Example)	$CPRINT var1; var2(i,j); var3; \ "VARIABLE #4: ", var4(i,-1,j)

$SETDATA or $CASE for Each Scene	$SETDATA or $CASE rectitle year, month, day, hour, lat, lon, zen BLANK | ntheta, theta(1..ntheta), nphi, phi(1..nphi) wvstart, wvend | bstart, bend | channel surfalb(bstart..bend) if ALBTYPE=0 | rsurf, nsurfs, whichsurfs(i), surffrac(i), i=1..nsurfs if ALBTYPE=1,2,3 | whichsurfs(1) if ALBTYPE=5 | brdftype, brdfparams() if ALBTYPE=6 | brdfscaler if ALBTYPE=7 tsurf, gemis(bstart..bend) if EMISSTYPE=0 | tsurf, emis if EMISSTYPE=4 nclouds, ntype(i), cldfrac(i), cldtemp(i), cldtop(i), cldthick(i), cldtau(i), nphases(i), cldphase1(i), cldre1(i), cldwc2(i), cldphase2(i), cldreice(i), cldwcice(i), i=1..nclouds BLANK | nover, novertype(i), overtype(i,j),…,j=1,novertype(I), overfrac(i),…, i=1..nover if nclouds >= 2 altchoice, pchoice, tchoice, wvchoice, ozchoice, hzchoice, nlev, frwv, fro3, frrhhz, frco2, fro2, frwv2 If not using old profiles then for each level i=1..n: [alt_in(i)] [p_in(i)] [t_in(i)] [wv_in(i)] [oz_in(i)] [aer_in(i)] BLANK | [zfirst] [h2ocol] [O3] [tauhaze] If column amounts are to be specified
$REPLACE or $EXCHANGE (Example)	$REPLACE var1=val; |var2=(start,end,inc), \ var3=[val1,val2,val3,val4],| var4=[val1,val2,val3]; \ var4(i)=val5
$COMMENT	$COMMENT <text> ; <text>

6 OUTPUT

Three types of output are possible:

1. descriptive output containing values and short textual descriptions,
2. user-customizable, via $CPRINT or
3. user-customizable, via the **writeusr.f** subroutine.

The descriptive output and/or either of the user-customizable output options may be chosen in a given *Streamer* run. See the *USEROUT* option for customizing your own output.

DESCRIPTIVE OUTPUT FILE

The following list is what is contained within the descriptive output file.

HEADER INFORMATION:

Streamer version number
Input File Name
Whether or not band weighting is applied.
Initial Options (case-by-case changes are not noted in the file):
Number of Streams, shortwave and longwave.
Number of phase moments coefficients and the method of computation (if the number of streams is greater than 2).
Whether or not gaseous absorption is included.
Whether or not Rayleigh scattering is included (shortwave only). What standard atmosphere is used, if any.
Whether or not a surface BRDF is being used.

THEN FOR EACH CASE (SCENE):

Case Number in Input File
Title for this record
Bandnumber and wavenumber start and end
Year, month, day, hour, latitude, longitude, zenith Angle (degrees)
Atmospheric Profiles (unscaled) of: Height [km], pressure [mb], temperature [K], water vapor density [g m-3], relative humidity [%], ozone [g m-3], and aerosol extinction coefficient [km—1], total column amounts (trapezoidal integration) for the entire scaled ("base"; i.e., input or standard) profiles of:
water vapor path (precipitable water) [g/m^2]
ozone path [g/m^2; path = Dobson-units * 0.021416667]
aerosols (optical depth) [unitless]
Scaling factors (as input): water vapor, ozone, haze relative humidity, carbon dioxide, oxygen, water vapor continuum
Cloud/clear-types in scene (13=clear)

CLOUD CHARACTERISTICS: type, top index, (level in profile with 1 being the top), bottom index, physical thickness [km], pressure thickness [mb], optical thickness (depth), fraction of scene (not

including the overlapped portion, if any), overlap sets (names and fraction), top temperature, top pressure and height, effective radius, liquid/ice water content, phase (ice/liquid)

IF FLUXES=.TRUE., THEN:

SURFACE CHARACTERISTICS: clear sky fraction, fraction of each surface type, input surface albedo at 0.6 um, surface temperature, broadband surface albedo by type (if the area fraction for a given type is 0, then the albedo will be 0; this may happen if one cloud overlaps another completely), broadband all-sky surface albedo. Albedo values will not be present in the descriptive file if only longwave calculations are done.

FLUXES: Downwelling direct, diffuse, and total shortwave, downwelling longwave, upwelling shortwave and longwave, net irradiances [W m-2], heating/cooling rates [K/day] for each layer with its top at the specified level. The levels output are the ones chosen with *OUTLEVS*, one line per level. Both shortwave and longwave will be output, but if only one of the two was specified, then values for the other will be 0.

CLOUD RADIATIVE EFFECT (if requested): top-of-atmosphere and surface, shortwave and longwave. If there were no clouds in the input stream, cloud forcing will be 0. [W m-2]

ELSE IF FLUXES=.FALSE., THEN:

SURFACE CHARACTERISTICS: clear sky fraction, fraction of each surface type, input surface albedo at 0.6 um, surface temperature, the clear-sky albedos or emissivities for each band.

RADIANCES: Integrated (spectrally) for each polar angle, azimuthal angle, and level [W m-2 sr-1]. The sign of the polar angle determines whether the printed radiance is upwelling (positive polar angle) or downwelling (negative polar angle). There is one line per level/polar angle/azimuth angle combination, for the user-specified output level (*OUTLEVS*). Top level albedo and/or brightness temperature (depending on the bands) will be output for each angle if upwelling radiance was computed.

OUTPUT NOTES

Since there are as many profiles as there are cloud/clear types in a scene, the only one that is output is the input or standard profile (the "base" profile), possibly extended to *SPACE*. They will not contain clouds unless they did initially.

Total column amounts are computed from the "base" profile; i.e., either the one input, the standard profile, or the one constructed based on the input column amounts. Inserting a cloud into the profile should not have any noticeable effect on the column amounts used for computations. The aerosol profile consists of extinction coefficients that represent the average value of the layer that has the given level as its top. Therefore, the lowest (surface) value is meaningless.

The flux/radiance results given for the entire scene is the weighted average of the fluxes/radiances computed for each part of the scene: clear sky, individual clouds, and cloud overlap sets. No 3-dimensional modeling of scattering effects is done. The fluxes/radiances for each part of the scene are weighted by their area fractions and summed.

Albedos for flux runs are for all-sky conditions (cloudy and clear), and are computed from the flux results. For radiance runs, the albedos or emissivities output for each band are either those that were input, or those based on the built-in data, depending on the value of *ALBTYPE*.

The values for cloud phase, effective radius, and water content are meaningless if user-specified cloud optical properties were read; i.e., if USERCLOUD=.TRUE.

In the case of overlapping clouds, the fraction of the scene occupied by the nonoverlapping part of individual cloud is computed and given in the output. Each overlapping pair constitutes another "type". These types will be used in the output of surface albedo by type.

Some variables whose values are missing, not calculated, or not applicable for the options specified may be given values of 0 in the output, especially the raw data file. In some cases 0 may even be a legitimate value. So, keep in mind what options are active when examining the data.

For radiances, negative polar angles represent downwelling radiances. However, -0.0 doesn't exist on many computers, so look at the angle cosines to distinguish between upwelling and downwelling radiances at 0 degree angles. Remember also that azimuth angles of 0 mean that you are facing the sun (forward scattering).

TOA albedos (e.g., sensor simulations) will be computed and output regardless of where the your top level actually is (i.e., what altitude), but since they use the solar flux as the downwelling radiation, they only make sense for the top of the atmosphere. The TOA solar fluxes (W m-2) for bands 106-129 are given in the Reference Guide.

If you get negative radiances when the actual radiances are near zero, try changing the angles and/or the number of streams. This may also be related to the delta-M truncation method used in DISORT.

In general, you shouldn't be concerned with a message about underflows that you might see on the screen (not in the output file) after running *Streamer*.

USER-CONTROLLED OUTPUT VARIABLE DESCRIPTIONS

These variables are used in the subroutines **writeout**, **writeusr**, and **writeprt** and most can be specified in $CPRINT/PRINT. Only **writeusr** is meant to be modified by the user. Lines with "*" in the first column contain variables that cannot be printed using $CPRINT/PRINT. If a variable is an array, the useful range of its dimensions, though not necessarily its declared size, is given in square brackets. Note that the output variable names are not always the same as the input name for a given parameter. For example, the input variable for cloud thickness, be it optical or physical, is *cldthick* but the output variables are *cldtau, zcthick,* or *pcthick,* as appropriate.

Options:

* version	Version number string
nstrshort, nstrlong	Number of streams for shortwave and longwave calculations
NCOEF	Number of coefficients for moments of phase function
ALBTYPE	Albedo type
weightfile	Name of the band weighting file [character*60]
STDPROF	Standard profile to be used as default
AERMOD	The aerosol optical model
AERVERT	The aerosol vertical profile model
IZD, ITD, IWV, IO3, ICTOP, ICTHK, IWAVE	Input unit specifications
USERCLOUD	Use cloud optical properties from a file?
cloudfile	Cloud properties file name
brdffile	BRDF file name

Data:

nrec	Record (case or scene) number
* rectitle	Record title [character*80]
wvstart, wvend	Starting and ending wavelengths (μm) or wavenumbers (cm^{-1}), depending on *IWAVE*
bstart, bend	Starting and ending band numbers
channel	The channel number if a band weights file was specified.
year, month, day, hour, lat, lon, zen	Day, hour, latitude, longitude, zenith angle (degrees)
nlev	Number of vertical levels in the profiles
alt, press, temps, wv, rh, ozone, aerosol	Profiles of altitude (km), pressure (mb), temperature (K), water vapor ($g\ m^{-3}$), relative humidity (%), ozone ($g\ m^{-3}$), and aerosol extinction (km^{-1}) [1..*nlev*]. These will not have been adjusted for any clouds present in the scene; i.e., they are the input profiles after unit conversions. They are not available until after running a case.
h2ocol, o3, tauhaze	Total water path (precipitable water; $g\ m^{-2}$; path = cm * 10000), total column ozone (ozone path; $g\ m^{-2}$; path = Dobson-units * 0.021416667), total haze (aerosol) optical depth. *h2ocol* and *o3* are scaled by *frwv* and *fro3*, respectively.

frwv,fro3, frrhhz,frco, fro2,frwv2	Fractional amounts of gases
ntypetotal	Total number of 'types' (e.g., cloud and surface) in scene. If there is any clear sky in the scene, or if the cloud forcing calculation was requested (*CLD-FORCE*=.TRUE.), then *ntypetotal* equals the number of clouds plus one. Otherwise, it is the number of clouds.
ntype	"Names" of each cloud, indexed as [1..*nclouds*] for the individual clouds and [*MAXCMODS..MAXCMODS+nover*] for the overlap pairs
nclouds	The number of individual clouds in the scene, not including overlapping pairs
* ctopnum, * cbottnum	Profile top and bottom levels for a cloud [1..*nclouds*]
zcthick, pcthick	Cloud physical thickness in km and mb [1..*nclouds*]
cldtau	Cloud optical thickness [1..*nclouds*]
cldthick	Cloud physical thickness [1..*nclouds*]
cldtauband	Cloud optical depth for each individual cloud and for spectral band [*bstart..bend*,1..*nclouds*]
cldfrac	Cloud or clear sky fraction [1..*nclouds*]
cldtemp	Cloud top temperature (K) [1..*nclouds*]
cldpress	Cloud top pressure (mb) [1..*nclouds*]
nphases	Number of cloud particle types in a cloud [1..*nclouds*]
cldphase1, cldphase2	Cloud particle phase or habit [1..*nclouds*]
cldre1, cldre2 cldwc1, cldwc2	Cloud particle effective radius (microns) and water concentration (g m^{-3}) for each of two possible particle types [1..*nclouds*]
cldwp	Cloud water path (g m^{-2}) [1..*nclouds*]
* phasetxt	Individual cloud effective radii, water contents, and a string for 'ice' or 'liq' phase [1..*nclouds*]
	NOTE: The above cloud arrays are indexed by *ntype*(); e.g., *cldtau*(*ntype*(1)) is the optical depth of the first cloud whose "name" is *ntype*(1). So while the useful size of these arrays is [1..*nclouds*] the actual subscripts to use are [*ntype*(1)..*ntype*(*nclouds*)].
nover	The number of overlapping cloud pairs
overfrac	Fractional coverage of an overlapping cloud pair [1..*nover*]
novertype	The number of overlapping clouds in each set [1..*nover*]
overtype	The "names" of the clouds that make up the overlap sets [1..*nover*,*1*..*nover-type*]
nsurfs	Number of surface types present in the scene
whichsurfs	The surface types (integer "names") present [1..*nsurfs*]
surffrac	Area fraction of each surface type [1..*nsurfs*]; indexed by *surffrac*(*which-surfs*(*i*))
emis	Surface emissivity, valid if *EMISSTYPE* is 4.
gemis	Surface emissivity in each longwave band [1..105], if *EMISSTYPE*=0.
tsurf	Observed (input) surface temperature (K)
rsurf	Observed (input) surface reflectance, 0.6 μm or broadband, depending on *ALBTYPE*.
brdftype	The surface BRDF model selected
brdfparams	The parameters of the selected BRDF model
brdfscaler	Scaling factor (multiplier) for user-supplied BRDF values
surfalb	Surface albedo in each shortwave band [106..129] based on the input albedo and the built-in models, not on the computed fluxes.

bandalb	The integrated surface albedo over the bands *bstart*..*bend* weighted by the relative solar flux in each band. This is based on internal model data or user-specified band albedos, not on computed fluxes.
alb06_adj, bbalb_adj	The adjusted/recomputed visible and broadband surface albedos based on the internal albedo data, if *ALBTYPE* > 0. If *ALBTYPE*=2 then *alb06_adj* is the same as *rsurf*, otherwise it is the computed 0.6 μm albedo. If *ALBTYPE*=3 then *bbalb_adj* is the same as *rsurf*, otherwise it is the computed broadband (bands 106 to 129) albedo. *bbalb_adj* is not computed from the fluxes but is a clear-sky albedo (if that is what was input). If computations are being done for the entire shortwave spectrum then *bbalb_adj* will equal *bandalb*.
albtotal	Surface albedo for each type [*albtotal*(1..*ntypetotal*)] and the average over the total scene [*albtotal*(*ntypetotal*+1)], computed from the upwelling and downwelling fluxes. *albtotal*(*ntypetotal*+1) corresponds to clear sky, if present or if *CLDFORCE*=TRUE.
avedirc, aveflxu, aveflxd	Direct (*avedirc*), and diffuse up and down fluxes at each level, shortwave and longwave. [1..*nlev*,1..2] where the second index is 1=shortwave, 2=longwave. (W m^{-2})
swflxd, netflx	Total downwelling shortwave flux (direct plus diffuse); net radiative flux. [1..*nlev*] (W m^{-2})
dtdt	Heating/cooling rate. [1..*nlev*] (K/day)
cldfrc	Cloud forcing at the surface and TOA, shortwave and longwave [1..*nlev*,1..2], where the second index is 1=shortwave, 2=longwave. (W m^{-2})
averad	Scene average intensity (radiance) at each polar and azimuthal angle, at each level. [1..*ntheta*,1..*nphi*,1..*nlev*] (W m^{-2} sr^{-1})
ntheta, nphi	Number of polar and azimuthal angles
theta, phi	Polar angles (cosines) and azimuthal angles (degrees) for radiances [1..*ntheta*] and [1..*nphi*]
satalb,sattb	Either the TOA (satellite) albedo or brightness temperature, depending on the bands. For radiance calculations only. [1..*ntheta*,1..*nphi*] (Unitless, K)
solflux	Band-weighted and integrated solar flux, not adjusted for solar zenith angle (W m^{-2})

7 SAMPLE INPUT AND OUTPUT

The sample input and output given below are examples only. The output corresponds to the input and is similar to that contained in the test files. Input for radiance calculations is similar to that for fluxes so is not shown here. See the files **testio** directory for input and output examples.

INPUT: FLUXES

```
;  This file specifies the calculation of shortwave and longwave fluxes.  It uses
;  an input profile as well as standard profiles.  It illustrates the use
;  of the REPLACE command with two different looping structures.
OPTIONS
.TRUE.                        ; Compute fluxes (or radiances)?        (FLUXES)
.FALSE.                       ; Include thermal emission in band 106? (IR106)
.TRUE.                        ; Compute cloud forcing?                (CLDFORCE)
2 2                           ; Number of streams, short and long     (NSTR*)
0                             ; Number of Legendre coeff.             (NCOEF)
.TRUE.                        ; Include gaseous absorption?           (GASABS)
.TRUE.                        ; Include Rayleigh scatter (shortwave)? (RAYLISHRT)
2                             ; Surface albedo control                (ALBTYPE)
4                             ; Surface emissivity control            (EMISSTYPE)
5 .TRUE.                      ; Std prof; extend input profile to 100 km?
5 1                           ; Aerosol model and profile
1 2 3 1 1 3 3                 ; Height, temp, wv, oz, cloud units, band spec.
4                             ; Output levels control
.TRUE.                        ; Descriptive output desired?
testflx.des
.FALSE.                       ; User-customized output?
<file name>
<file name>
.FALSE.                       ; Read cloud optical properties?
<file name>
<file name>                   ; BRDF file name
CASE
Clear sky only, April conditions
 92  4 28 22.0  72.88  144.50 -99.0
                                      ; Viewing geometry
 1 129
 0.75 2 1 0.05 5 0.95
 -25.15 0.99
 0
                                      ; Cloud overlap
1 1 1 1 0 0 13 1.0 1.0 1.0 1.0 1.0 1.0
  14.101  125.4  -50.9   2.0
  10.948  204.9  -56.2   7.0
   8.486  303.5  -60.2  35.0
   7.864  335.1  -57.1  43.0
   6.651  404.8  -50.0  47.0
   6.041  444.1  -45.5  43.0
   4.860  528.4  -36.8  58.0
   3.702  622.4  -26.8  39.0
   2.618  721.7  -19.7  67.0
   1.592  828.2  -16.2  72.0
   0.973  898.3  -10.9  60.0
   0.577  945.8  -12.2  67.0
```

```
   0.000 1022.0  -23.2  78.0
300.0 0.25
CASE
Cloudy sky, 2 clouds, April conditions, shortwave only
 92  4 28 22.0  72.88  144.50 -99.0
                                                    ; Viewing geometry
 106 129
 0.75 2 1 0.05 5 0.95
 -25.15 0.99
 2 1 0.4 -19.15 725.0 999 5.0 1 0 6.0 0.1 999 999 999
   3 0.6 -60.15 300.0 999 1.0 1 1 20.0 0.01 999 999 999
 0                                                  ; Cloud overlap
-1 -1 -1 -1 0 0 13 1.0 1.0 1.0 1.0 1.0 1.0

SETDATA
Some clear sky, some cloudy sky, 2 clouds, April conditions, longwave, loops
 92  4 28 22.0  72.88  144.50 -99.0

 1 105
 0.75 2 1 0.05 5 0.95
 -25.15 0.99
 2 1 0.4 -19.15 725.0 999 5.0 1 0 6.0 0.1 999 999 999
   3 0.6 -60.15 300.0 999 1.0 1 1 20.0 0.01 999 999 999
 0
2 2 2 2 2 2 0 1.0 1.0 1.0 1.0 1.0 1.0

REPLACE cldtau(2)=(0, 8, 4), cldrewat(1)=[8., 10.]
```

OUTPUT: FLUXES

The following output corresponds to only part of above input and is not a complete output file.

```
Streamer, v2.1pB (preliminary, Beta)

 Streamer sample input file - Fluxes
 Input File: testflx.inp
  No spectral weighting.
  INITIAL OPTIONS (Later changes will not be noted):
   Number of Streams, Shortwave:  2, Longwave:  2
    Gaseous absorption included.
    Rayleigh scattering included.
    Default profile: Subarctic Winter
    Default aerosol optical model: Arctic
    Default aerosol vertical profile: Background trop. and strat.

 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 Case Number in Input File:     2
 Cloudy sky, 2 clouds, April conditions, shortwave only
 Band number range: 106 - 129
 Spectral Interval:  2500 1/cm (  4.00 um) to 35710 1/cm (   .28 um)
 Year: 92, Month:  4, Day: 28, Hour:  22.00
 Lat:  72.880, Lon: 144.500, Zenith Angle (degrees):  58.82
[Profiles not shown]
 Total Column Amounts (scaled) -
    Water Vapor:  3671.07 g/m^2
    Ozone:   6.43 g/m^2
    Aerosols Optical Depth:  .25 (unitless)
```

```
 Scaling Factors - w.v., O3, haze RH, CO2, O2, w.v. continuum:
     1.00  1.00  1.00  1.00  1.00  1.00
 Cloud/clear types (models) in scene (13=clear):   1  3 21
 INDIVIDUAL CLOUD CHARACTERISTICS:
 Model Top Bott Zthick Pthick  Frac  Tau  Ttop   Ptop    Re    WC   Phase
         Index    (m)   (mb)             (K)    (mb)   (um)(g/m^3)
 ------------------------------------------------------------------------
    1  18  19   187.1   18.4    .40   5.0 254.0  725.0   6.0  .100  Liq
    3  12  13   800.0   40.6    .60   1.0 213.0  300.0  20.0  .010  Ice
    (Note: Above fraction does not include overlapping portion, if any.)
 SURFACE CHARACTERISTICS:
   Clear Sky Fraction:  .00
   Surface Type Fractions -
      Sea Water:  .05, Meltponds:  .00, Snow:  .95, Bare Ice:  .00
      Vegetation:  .00, Dry Sand:  .00, Freshwater:  .00
   Observed (Input) Surface Albedo (0.6 um):   .750
   Broadband All-sky Surface Albedo, by type:
       1:  .645   3:  .617  21:  .608
   Broadband All-sky Surface Albedo:  .628
 ALL-SKY FLUXES, CLOUD RADIATIVE EFFECT (W/m^2), HEATING RATE (degrees K/day):
     DirSW    DiffSW  TotalSW     LW     DiffSW     LW              Heating
      Down     Down     Down    Down       Up       Up      NET       Rate
 ------------------------------------------------------------------------------
  1  692.26      .00   692.26      .00   404.66      .00   287.61      .925
  2  692.24      .01   692.25      .00   404.65      .00   287.60     1.049
  3  692.05      .11   692.16      .00   404.63      .00   287.53     1.285
  4  691.86      .20   692.06      .00   404.61      .00   287.45     2.338
  5  691.34      .38   691.72      .00   404.58      .00   287.13     2.823
  6  690.08      .76   690.84      .00   404.54      .00   286.30     1.981
  7  687.86     1.57   689.44      .00   404.44      .00   285.00     1.334
  8  683.67     3.63   687.30      .00   404.28      .00   283.02      .946
  9  676.07     7.88   683.95      .00   404.04      .00   279.91      .642
 10  655.85    21.34   677.19      .00   403.04      .00   274.14      .407
 11  640.80    31.35   672.14      .00   401.85      .00   270.29      .510
 12  620.93    43.62   664.55      .00   400.23      .00   264.31     3.656
 13  299.06   308.48   607.55      .00   356.97      .00   250.58      .541
 14  289.41   312.41   601.82      .00   355.72      .00   246.10      .708
 15  283.24   314.65   597.89      .00   355.10      .00   242.79      .876
 16  269.10   318.79   587.89      .00   353.88      .00   234.02      .934
 17  253.98   322.29   576.27      .00   352.68      .00   223.60     1.270
 18  236.09   325.01   561.11      .00   352.48      .00   208.63     1.140
 19   39.15   448.94   488.09      .00   293.86      .00   194.23      .948
 20   36.83   442.43   479.26      .00   292.91      .00   186.35     1.035
 21   34.96   438.49   473.45      .00   292.94      .00   180.51      .729
 22   32.55   434.06   466.61      .00   292.69      .00   173.93      .000

 Cloud Radiative Effect - Level:  1, Shortwave:  -46.9, Longwave:     .0
 Cloud Radiative Effect - Level: 22, Shortwave:  -44.1, Longwave:     .0
```

OUTPUT: RADIANCES

[... HEADER, PROFILE, SURFACE AND CLOUD DATA SIMILAR TO THAT FOR FLUXES ...]

```
 INTEGRATED SCENE RADIANCES (negative downward, positive upward):
 Lev  Polar Angle(deg)  Azim Angle(deg)  Radiance(W m-2-sr)
        (cosine)
-------------------------------------------------------------
```

```
 1    60.00 ( 0.50)      45.00           0.3793E+02
 1    60.00 ( 0.50)     135.00           0.3045E+02
 1    29.54 ( 0.87)      45.00           0.3229E+02
 1    29.54 ( 0.87)     135.00           0.3046E+02
 1     0.00 ( 1.00)      45.00           0.3058E+02
 1     0.00 ( 1.00)     135.00           0.3058E+02
22    60.00 ( 0.50)      45.00           0.2747E+02
22    60.00 ( 0.50)     135.00           0.2747E+02
22    29.54 ( 0.87)      45.00           0.2747E+02
22    29.54 ( 0.87)     135.00           0.2747E+02
22     0.00 ( 1.00)      45.00           0.2747E+02
22     0.00 ( 1.00)     135.00           0.2747E+02

TOP LEVEL ALBEDO (NOT ADJUSTED FOR SOLAR ZEN ANG):
Polar Angle(cos)  Azim Angle(deg)  Albedo
-----------------------------------------
    0.50             45.00         0.424
    0.50            135.00         0.340
    0.87             45.00         0.361
    0.87            135.00         0.340
    1.00             45.00         0.342
    1.00            135.00         0.342
```

8 UTILITIES

STREAMER AS A SUBROUTINE

While the user interface with provides tremendous flexibility and control, there may be applications where *Streamer* needs to be integrated into existing Fortran code. For example, simple paramerizations of radiative fluxes currently used in sea ice models could be replaced with a radiative transfer model. In applications such as this, *Streamer* has to be available as a subroutine, where all basic data normally given in the input file are passed in as a list of arguments to the subroutine.

The file **substrmr.f** contains *Streamer* as a subroutine. To use it, the following must be observed:

- Declare all input and output variables. The easiest way to do this is to copy the variable declarations from **subtest.f**, a sample calling program, into your calling program.
- Set default values for options and input variables by modifying the subroutine **set_defaults** in the file **defaults.f** or using the defaults file **streamer.def**. See the *Input/Setting Default Values* section of this guide. Only those variables that you are not passing in values for, but whose values are needed, are important. This is critical, as most options are not arguments to the **streamer** subroutine, and the conventional input file mechanism of *Streamer* is not used (to keep the code small).
- Pass in/out the variables that are listed in the formal argument list ("subroutine streamer(...))". As with Streamer, if a variable is not needed for a particular type of calculation (e.g., viewing geometry for fluxes), then those variables do not have to have valid values. All input and output variables are described in the User's Guide.
- Call this subroutine once before any calculations are desired. The purpose of this call is to set the default variable values. Nothing else will be done on the first call.
- Don't change the parameter units from one call to the next.
- Don't use the Fortran I/O unit numbers 17, 18, 19, 20, or 21.
- Two of the three standard Streamer output mechanisms (descriptive file and the **writeusr** subroutine) are still available, but the output units (files) must be closed in the calling program. Copy the close statements from **subtest.f** into the calling program if any of the output files are used (i.e., if either of the output flags in the $OPTIONS block is TRUE).
- Compile all the Fortran files listed in the first part **Makesub.sun**, a makefile that can be used to create a library of all the subroutines. It may need to be modified for your operating system.
- If desired, test the model by building an executable with **subtest.f**, run it, and compare the results to those generated with the stand-alone version of *Streamer* with **testio/testsub.inp** as input. The data in **testsub.inp** are the same as those in **defaults.f.**

NOTES

1. The values of the input variables will not be changed **except** for the first (setup) call. On that call the values of the input variables on entry will be ignored and the variables will be assigned values specified in the **set_defaults** procedure (**defaults.f**). It is recognized that some input variables that are transformed may be useful in the calling program (e.g., *h2ocol*), but no facility for returning the new values is provided. They can, however, be output via the standard *Streamer* output mechanisms.
2. A value of -1 for the *choice variables, which means that previous profiles should be used, is meaningless here. The profiles that are passed in will automatically be used.
3. Not all possible output variables are given in the argument list; e.g., the constructed profiles *alt*,

press, etc. are not present.

VISIBLE-TO-BROADBAND CONVERSION

The FORTRAN program **getalb.f** can be used for displaying either the visible or broadband albedo, as well as the band-by-band albedos, for any of the built-in surface models. See the documentation at the beginning of the file. If you obtained the binary distribution for your operating system, or if you built the binaries with "make all", then **getalb** should be present. Otherwise, you'll have to build it with "make getalb".

CLOUD AND AEROSOL OPTICAL PROPERTIES

The FORTRAN programs **getcloud.f** and **gethaze.f** can be used for the cloud and aerosol optical properties, i.e., extinction coefficient, single-scattering albedo, and the asymmetry parameter. See the documentation at the beginning of the file. If you obtained the binary distribution for your operating system, or if you built the binaries with "make all", then **getcloud** and **gethaze** should be present. Otherwise, you'll have to build it with "make getcloud" and "make gethaze".

WAVELENGTH TO BAND CONVERSION AND VICE VERSA

The FORTRAN program **getband.f** can be used to convert wavelengths or wavenumbers to *Streamer* band numbers or to convert band numbers to wavelengths or wavenumbers. Tables in this *Guide* provide the same information as the **getband** utility.

WEB BROWSER INTERFACE

A web browser interface to *Streamer* is available. The interface allows the user to create a basic *Streamer* input file by selecting values with buttons, pull-down menus, and text fields. Figure 8-1 shows the main and data definition pages. It should be particularly useful to the new user. However, it does not provide access to all of *Streamer*'s features, and is limited in the following ways:

- This interface can only read input files that it generated.
- Only one OPTIONS block and one CASE or SETDATA block is generated.
- Only one REPLACE statement can be specified.
- The web server might not be the same computer on which the **streamer** program is located, in which case you won't be able to run the model from the interface.

The browser interface has to be installed on your local web server by the systems administrator. There are Perl scripts that have to go in the web server's **cgi-bin** directory, and a number of other files that should go in a subdirectory of the web server's main document directory. How the interface is set up and used depends on how the web server and network are configured. In the best case the web server is configured to permit each user to create a subdirectory of the main web interface directory, and the **streamer** program can be run on the same computer as the web server. In this case the interface can be used to create and save *Streamer* input files, run streamer, and view the results. In the worst case users do not have accounts on the computer running the web server, they cannot create directories accessible by the web server, and the model isn't on the web server. In that case the interface can still be used to create input files but not to run the model with them. To run the model the user must: (1) create an

input ("settings") file with the web interface, (2) save it on the web server, (3) view it with the browser and use the "Save As..." function to save it as a plain text file on your local computer, (4) move the settings file to whatever computer the model is on, if it's not on your local computer, and finally (5) run **streamer** there via the command line. It sounds complicated but it really isn't. Have you systems administrator read the *Obtaining, Building, and Running Streamer* section of this guide.

WEB VERSION OF THIS DOCUMENTATION

An HTML (hypertext markup language) version of the *User's* and *Reference Guides* are available for viewing with your web browser. Get the file **htmldocs.tar** and extract the files it contains. You can do this anywhere and then bookmark it in your browser. But you should also have a copy of the files put on the web server with the web interface (see above), in the working directory set up by the systems administrator. That way the manual can be accessed from the web interface.

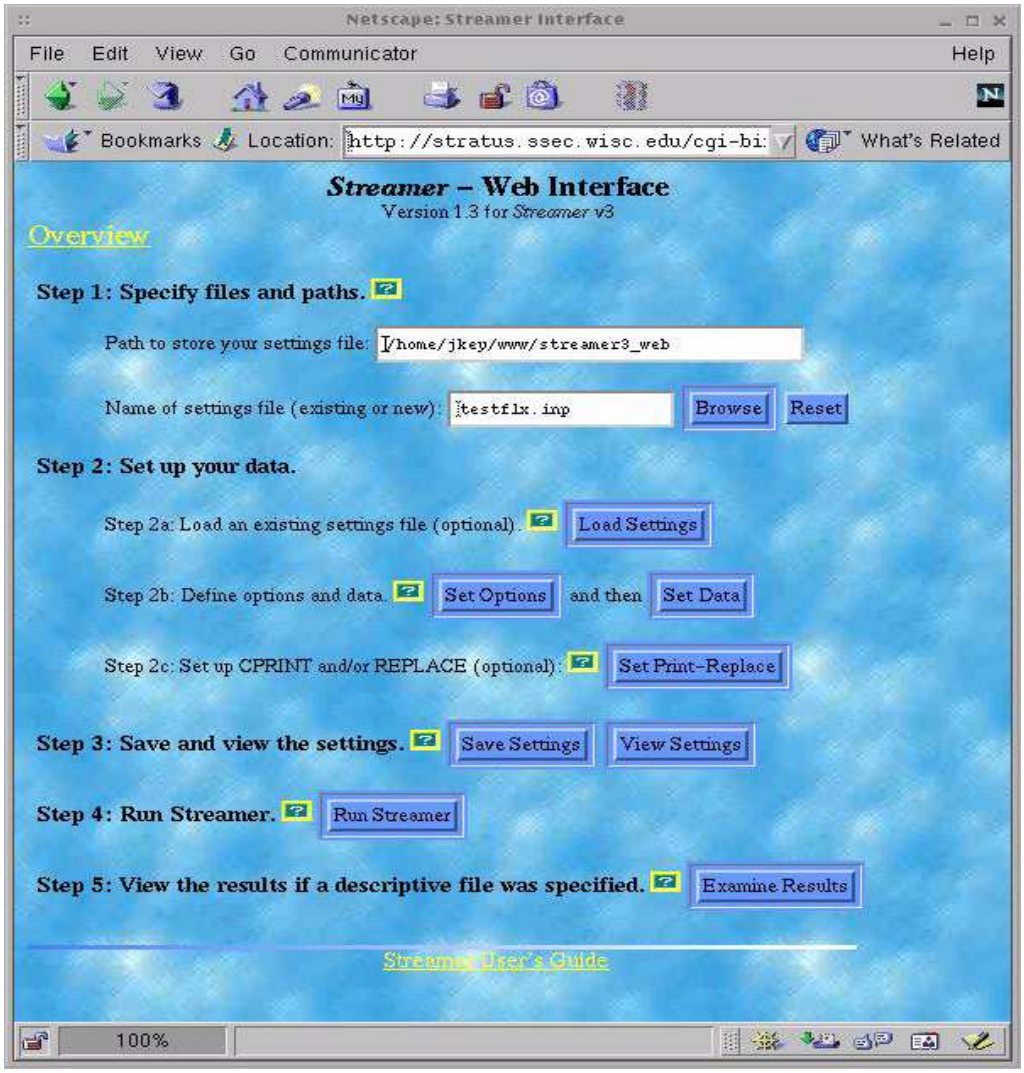


Fig. 8-1. The main and data web browser pages for the web interface.

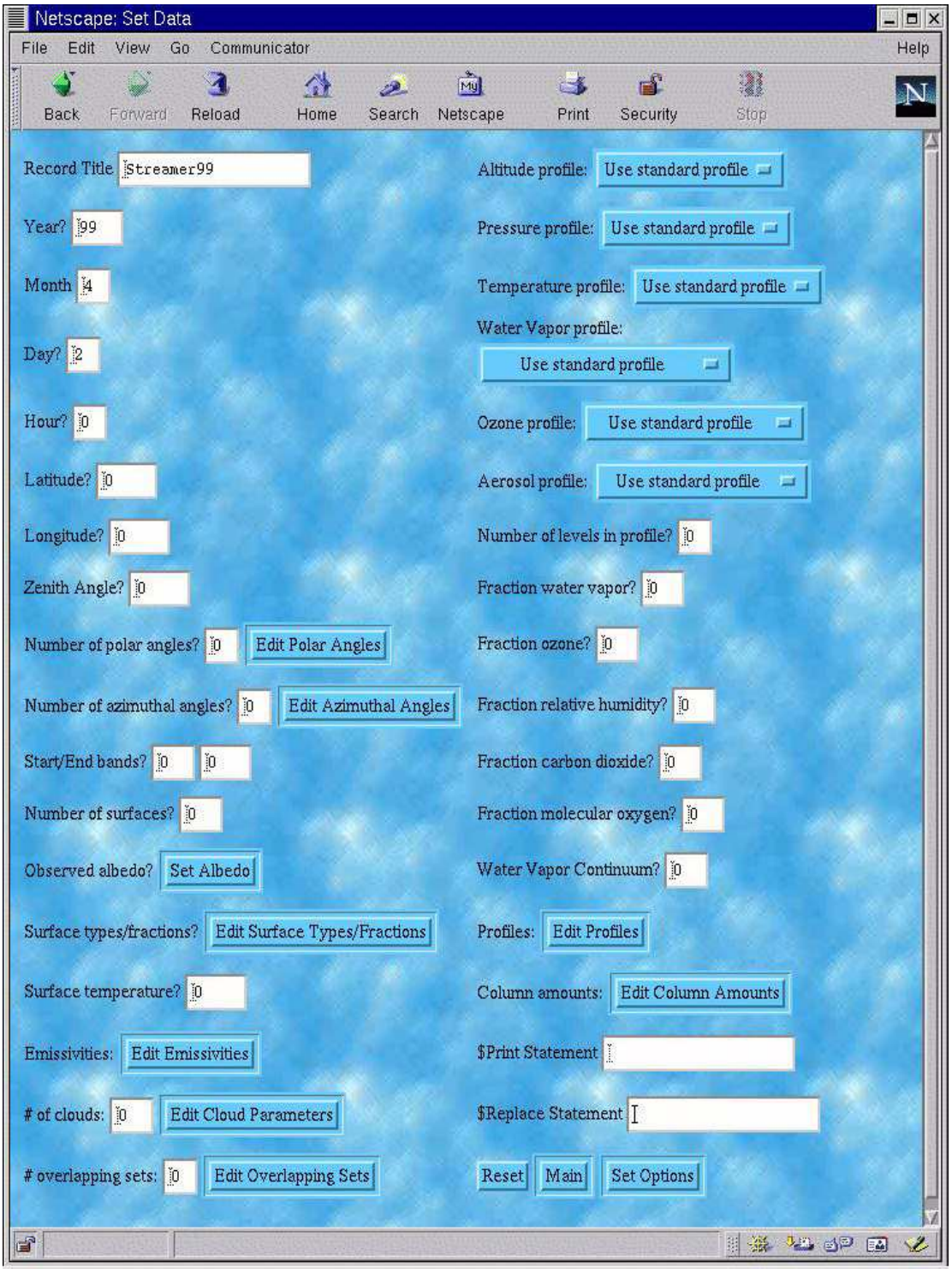

Netscape: Set Data
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Record Title Streamer99
Year? 99
Month 4
Day? 2
Hour? 0
Latitude? 0
Longitude? 0
Zenith Angle? 0
Number of polar angles? 0
Edit Polar Angles
Number of azimuthal angles? 0
Edit Azimuthal Angles
Start/End bands? 0 0
Number of surfaces? 0
Observed albedo?
Set Albedo
Surface types/fractions?
Edit Surface Types/Fractions
Surface temperature? 0
Emissivities:
Edit Emissivities
# of clouds: 0
Edit Cloud Parameters
overlapping sets: 0
Edit Overlapping Sets
Altitude profile: Use standard profile
Pressure profile: Use standard profile
Temperature profile: Use standard profile
Water Vapor profile:
Use standard profile
Ozone profile: Use standard profile
Aerosol profile: Use standard profile
Number of levels in profile? 0
Fraction water vapor? 0
Fraction ozone? 0
Fraction relative humidity? 0
Fraction carbon dioxide? 0
Fraction molecular oxygen? 0
Water Vapor Continuum? 0
Profiles:
Edit Profiles
Column amounts:
Edit Column Amounts
$Print Statement
$Replace Statement
Reset
Main
Set Options

Fig. 8.1, cont.

9 SAMPLE APPLICATIONS

This section contains a few application using *Streamer*. Figure 9-1 gives downwelling shortwave and longwave fluxes as a function of cloud physical thickness and cloud fractional coverage in the scene. Figure 9-2 shows TOA reflectances in AVHRR channels 2 and 3 as a function of cloud optical depth, droplet effective radius, viewing and illumination geometry, and surface reflectance. The data used to generate this plot can be used in the retrieval of cloud effective radius and optical depth from AVHRR observations. Figure 9-3 shows downwelling longwave fluxes over the Arctic that were computed using cloud and atmospheric parameters derived from the TOVS (Tiros Operational Vertical Sounder). Figure 9-4 shows a comparison of downwelling longwave radiance at the surface as measured by the AERI instrument with that calculated by *Streamer*.

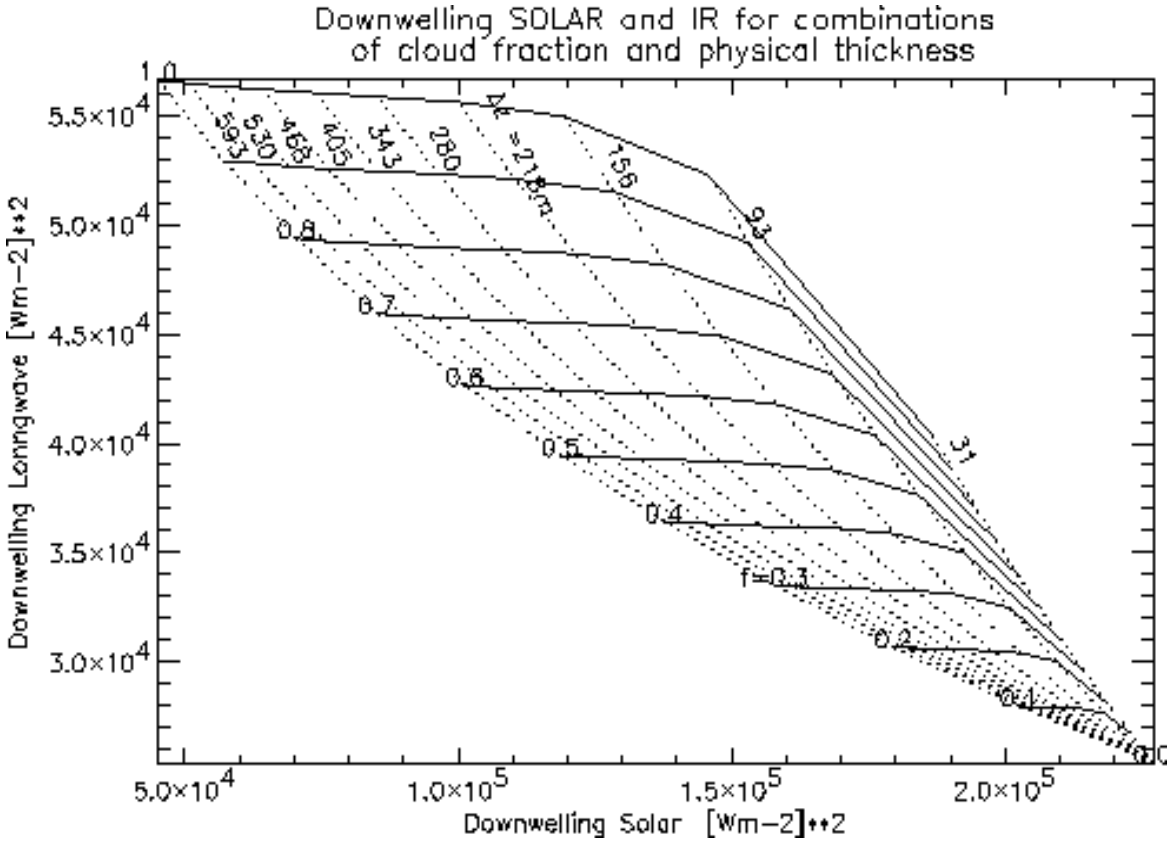


Fig. 9-1. Downwelling shortwave and longwave fluxes at the surface as a function of cloud fraction and physical thickness. (Courtesy of A. Schweiger)

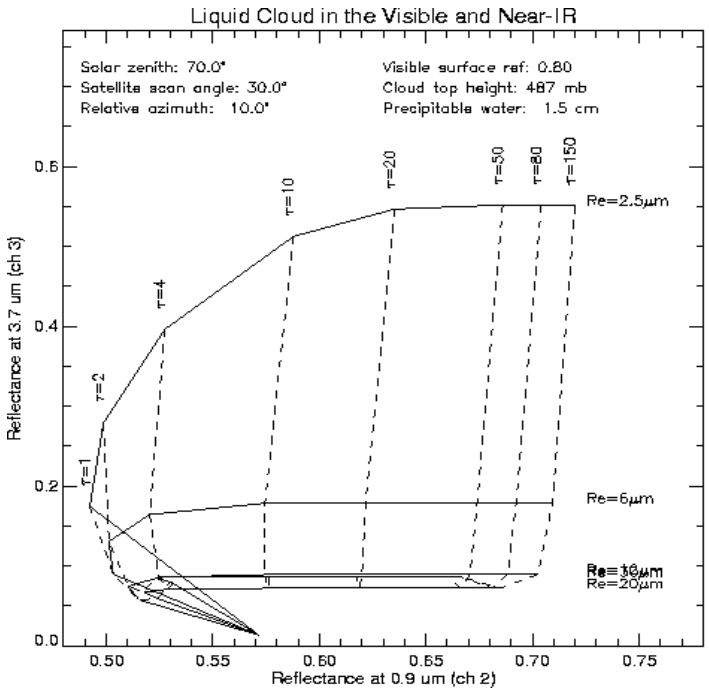


Fig. 9-2. Top of atmosphere reflectance in AVHRR channels 2 and 3 as a function of cloud droplet effective radius and visible optical depth.

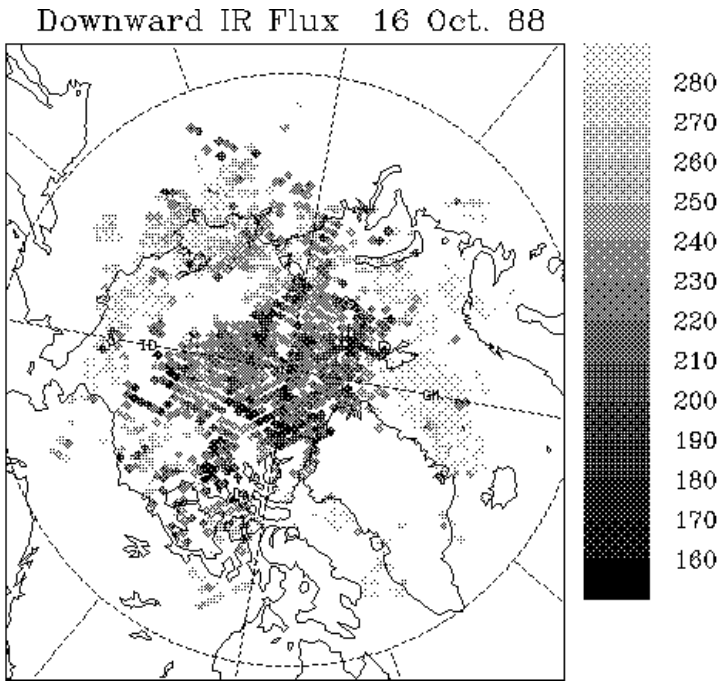


Fig. 9-3. Downwelling longwave fluxes at the surface computed using TOVS-derived profiles and cloud properties. (Courtesy of J. Francis)

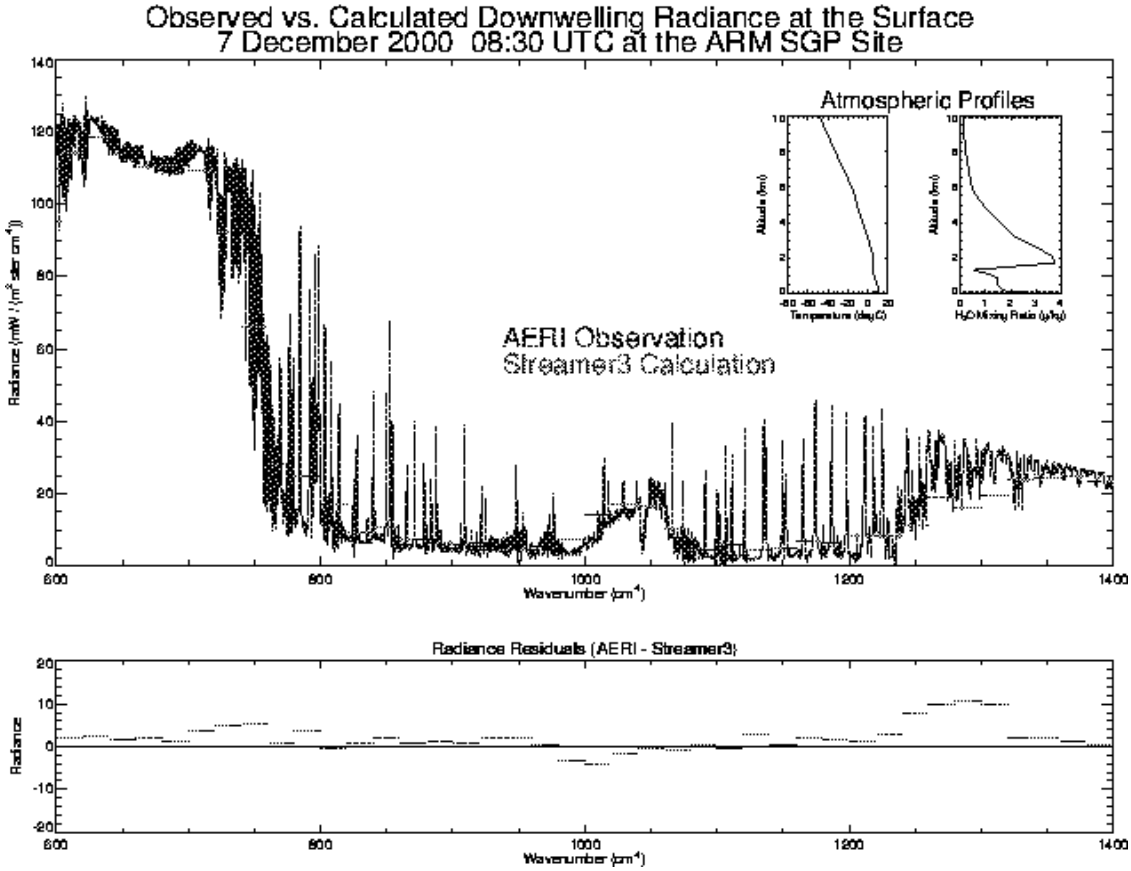


Fig. 9-4. Comparison of downwelling longwave radiance at the surface as measured by the AERI instrument with that calculated by *Streamer*. Observations were taken at the ARM SGP site in northern Oklahoma. The temperature profile used in the calculations is from a radiosonde and the moisture profile is from Raman lidar. The calculations used 8 streams with 24 coefficients, a rural aerosol model with an optical depth 0.09, total precipitable water of 1.06 cm, and total column ozone of 380 DU. (Courtesy of D. Turner)

10 PROGRAM STRUCTURE

Following is a list of the main subroutines that comprise *Streamer* and the files that contain them.

streamer	Main program. Loops over all scenes.	streamer.f
getinfln	Reads the options/data file name	options.f
getcommand		getcomm.f
readoptions	Reads the options	options.f
parse_replace, substitute	Process $REPLACE input lines	replace.f
readweights	Reads the bandweights	readuser.f
readcloudprops	Reads user-specified cloud properties	readuser.f
readbrdf	Reads user-specified surface BRDF	readuser.f
save_values	Saves and restores variable values.	savevals.f
parse_print	Process print commands	print.f
readdata	Reads the user data for each scene	readdata.f
checkoptions	Some error checking on the options	options.f
checkdata	Does some error checking on the data	readdata.f
scan2zen	Converts satellite scan to zenith angle	zenith.f
zenith	Computes solar zenith angle	zenith.f
specint	Determines spectral bands; wavelength conversions, etc.	specint.f
surf_alb	Determines the surface albedo in shortwave bands	surface.f
vis_ratio_bands	Scales the internal albedos	
broad_ratio_bands	Scales the internal albedos	
snow*, bareice, frswater, brgwater, sand, veggie grass, drygrass, decidous, conifer	Returns the modeled/observed albedos for snow ice, meltponds, open water, sand, and vegetation; some as a function of solar zenith angle	
tau2z, z2tau, ztau2wc, usertau2z, userz2tau	Converts cloud physical <-> optical thickness, and both (together) to water content.	tau2z.f
profiles	Constructs or completes profiles	profiles.f
insert_cloud	Inserts a cloud into the profiles	insrtcld.f
specint	Determines spectral bands from wavelength/ wavenumber or vice versa	specint.f
strats	Loops over the spectral intervals	strats.f
absorber_amount	Returns the gas absorber amounts at each level	absamnt.f
optical_props	Gets the cloud and aerosol optical properties	optics.f
cloudparm	Returns cloud optical properties from built-in models	cldparm.f
watcld, icecldshort, icecldlong	Parameterizations of optical properties	cldparm.f
usercloudparm	Returns cloud optical properties from user-specified models	cldparm.f
hazeparm	Returns haze bulk properties	hazeparm.f
trophaze, urbanhaze, ruralhaze, oceanhaze, arctichaze	Parameterizations of optical properties	hazeparm.f
solar_flux	TOA solar flux, adjusted for Earth-Sun distance	solflux.f

get_esft	Returns the exponential sum fitting gas terms	getesft.f
get_rayleigh	Returns the Rayleigh scattering coefficients	getray.f
layer_tauscat	Computes the layer optical properties	tauscat.f
compute	LOOPS over the gases and the ESFT terms	compute.f
toon2str	The 2-stream solver	toon.f
disort	The discrete ordinate solver	disort-d.f
writeprt	Output via the $CPRINT command	writeprt.f
writeout	Output to descriptive data file	writeout.f
writeusr	User-controlled routine; a stub	writeusr.f

The built-in data, such as surface albedo, gas absorption, cloud parameters and profiles, are contained in the following files.

prmstmnt.inc	Variables setting array sizes.
specint.inc	Spectral intervals (bands), used in a variety of subroutines
profdata.inc	Standard profiles, used in profiles
esftdata.inc	Gas absorption, used in get_esft
rayleigh.inc	Rayleigh scattering coefficients for the shortwave bands
solflux.inc	Solar fluxes
cldparm.f	Cloud optical property parameterizations
watcld.inc	Water cloud optical properties.
icecld.inc	Ice cloud optical properties.
hazeparm.f	Haze optical property parameterizations
surface.f	Surface reflectance data, used in surf_alb