kCARTA: An Atmospheric Radiative Transfer Algorithm using Compressed Lookup Tables

Sergio De Souza-Machado, L. Larrabee Strow,
Howard Motteler and Scott Hannon

Physics Department
University of Maryland Baltimore County
Baltimore, MD 21250 USA

Copyright 2002
University of Maryland Baltimore County
All Rights Reserved

v1.11,1.12,1.14 November 7, 2010

Sergio De Souza-Machado: sergio@umbc.edu
L. Larrabee Strow: strow@umbc.edu

Contents

List of Tables

List of Figures

ABSTRACT

kCARTA is a radiative transfer code for a non-scattering Earth's atmosphere. It can be used to output monochromatic gas optical depths, layer-to-space transmittances and radiances. In addition, it can also do clear sky radiative transfer computations, or radiative transfer computations in the presence of scattering.

In the case of clear sky radiative transfer, kCARTA can compute and output analytic Jacobians with respect to temperature and gas amount. As well as the layer Jacobians being output, kCARTA dumps out a separate file which contains the $surface$ *temp* and *columngas* jacobians (for a x0.1 column gas amount perturbation).

The user can easily change satellite viewing angles and surface boundary conditions. As the absorptions coefficients have been precomputed, the spline interpolations required to compute the optical depths for arbitrary gas profiles can be done very rapidly, as can the the spline derivatives needed for the Jacobians. This makes kCARTA very fast. kCARTA v1.03+ allows the user to subdivide the lower atmosphere layering structure, and then clump together the upper atmosphere layers, thereby allowing the program to calculate radiances for an uplooking instrument more accurately. The namelist files required to steer a run are easy to create. We hope that the ease of use, range of features and speed of kCARTA, make it a useful tool.

In addition to the main kCARTA source code, some packages need to be picked up and linked/compiled with the *kcarta* code. One is $KLAYERS$, which allows an user to input a radiosonde or retrieved point profile, and output a layer averaged profile that kCARTA can use. Another is the RTP package, which is a AIRS Level II file format. To maintain compatibilty with the AIRS products, kCARTA allows the user to completely or partially interface with the RTP file format, which can contain the layer averaged profile and associated atmosphere definitions (and convolved radiances).

We have written and tested a twostream $kSCATTER$ code, which allows fast scattering computations even in the presence of a solar beam. More well known scattering packages have also been integrated into the code, such as $DISORT$ and $RTSPEC$. To produce the Mie scattering parameters required by both these codes, we have packaged $RTSPEC$, as it contains the code necessary (*sscatmie.f*) to produce these parameters for ice and water clouds. In addition, for general aerosols, we have interfaced the $OPAC$ database, which contains general aerosol properties, with *sscatmie.f* Along with all this, we provide the Fortran and Matlab source code that enables on to read the output from kCARTA.

In addition, we have recently begun implementing NonLTE computations into kCARTA. As the code essentially performs a line by line spectroscopic computation for the layers in question, this significantly slows down the code! Currently the code is optimized for the CO2 4.3 um branch, and closely parallels the GENLN3 code. In addition, we have already included first order linemixing for the R branch of the strongest $\Sigma - \Sigma$ band (vib center $\simeq$ 2349 cm^{-1}), in the 2380 - 2430 cm^{-1} temperature sounding region, along with a “chi function” to bring it more into agreement with the full kCARTA line mixing.

This document is very much a work in progress. Some major omissions include references, significant examples of kCARTA output, and comparisons of kCARTA output to observed spectra. These omissions will be rectified in the future. Please give us your feedback on both the code and the documentation! This manual should work for v1.11+.

1 Introduction

kCARTA stands for "kCompressed Atmospheric Radiative Transfer Algorithm." This is an infrared, "monochromatic" radiative transfer algorithm written for a one dimensional non-scattering Earth atmosphere. For a downward looking instrument, in a clear sky, the surface term and layer emission terms are automatically included in the radiative transfer calculation. In addition, reflected thermal and solar terms can also be included :

$$R(\nu) = R_{surface}(\nu) + R_{layer\ emission}(\nu) + R_{thermal}(\nu) + R_{solar}(\nu) \qquad (1)$$

where the terms are the surface, layer emissions, reflected thermal and solar respectively. The reflected thermal term is computed accurately by determining (monochromatically) the layer to ground cumulative sum of absorption coefficients, and then using an optimum diffusive angle based on a parameterization of angle as a function of this cumulative sum. This makes the inclusion of reflected thermal in the code quick and accurate. By differentiating the radiance equation with respect to a layer gas amount or temperature, the radiance Jacobian is obtained. Dropping the surface and reflected thermal terms enables kCARTA to compute the radiance measured by an upward looking instrument as well. The program can either assume a plane parallel atmosphere, or include effects on the satellite viewing angle due to the curvature of the earth.

The absorption coefficients used by the code are computed using a database of look-up tables. The look up tables are compressed, using a Singular Value Decomposition (SVD) technique, to produce our kCompressed database. The point spacing of the current database is 0.0025 cm^{-1}, which is an average over five points spaced at 0.0005 cm^{-1}. To compute the absorption coefficients for an arbitrary profile, the look-up tables are (cubic) spline interpolated in temperature, and scaled in gas absorber amount. These splines allow us to easily compute the analytic temperature derivatives, from which we can compute temperature Jacobians. kCARTA also dumpus out an adiditional file which has radiances computed assuming the surface temperature and gas column amounts have increased by 1K and a factor of 0.1 respectively (this second file can be read in using *readkcBasic.m*).

Taking into account the view angle correction, if τ_i is the optical depth due to gas G at layer i, given by $\tau_i = q_i K_i/\mu i$ where the symbols respectively stand for optical depth (dimensionless), gas amount (kmoles/cm2), gas absorption (cm2/kmol) and $cos(viewangle)$, then the finite diference colum gas jacobian is given by the difference between the new and unperturbed radiances $\delta r = r_{new} - r - 0$ when the gas amounts in layers 1 to N are perturbed by a fraction δ. The (finite differences) column jacobians can be obtained from the (gas) layer analytic jacobians using

$$\delta r = \frac{\partial r}{\partial q_1}\delta q_1 + \frac{\partial r}{\partial q_2}\delta q_2 + ... + \frac{\partial r}{\partial q_N}\delta q_N$$

or

$$\delta r = J_1 \delta q_1 + J_2 \delta q_2 + ... J_N \delta q_N$$

Usually we take a constant perturbation to the column ie $q_l \rightarrow q_1(1+f)$ where $f \ll 1$. Then $\delta q_l \rightarrow f q_l$ and $\delta r = f\{J_1 q_1 + J_2 q_2 + ... + J_N q_N\}$. For example, for a 2 layer atmosphere the upwelling radiance (without background thermal or solar terms)jacobian terms J_l is

$$\begin{aligned} r = \quad & \epsilon B(T_s) exp(-q_1 K_1/\mu_1) exp(-q_2 K_2/\mu_2) + \\ & B(1)(1 - exp(-q_1 K_1/\mu_1)) exp(-q_2 K_2/\mu_2) + \\ & B(2)(1 - exp(-q_2 K_2/\mu_2)) \end{aligned}$$

from which the layer jacobian terms J_i reduce to

$$\begin{aligned} J_1 = \frac{\partial r}{\partial q_1} = \quad & -\tfrac{K_1}{\mu_1} \epsilon B(T_s) exp(-q_1 K_1/\mu_1) exp(-q_2 K_2/\mu_2) + \\ & -\tfrac{K_1}{\mu_1} B(1)(exp(-q_1 K_1/\mu 1) exp(-q_2 K_2/\mu 2) \end{aligned}$$

$$\begin{aligned} J_2 = \frac{\partial r}{\partial q_2} = \quad & -\tfrac{K_2}{\mu_2} \epsilon B(T_s) exp(-q_1 K_1/\mu_1) exp(-q_2 K_2/\mu_2) + \\ & -\tfrac{K_2}{\mu_2} B(1) exp(-q_1 K_1/\mu 1) exp(-q_2 K_2/\mu 2) + \\ & -\tfrac{K_2}{\mu_2} B(2) exp(-q_2 K_2/\mu 2) \end{aligned}$$

The output order is same as in regular jac file ie gases followed by STEMP.

The current database spans 605 cm^{-1} to 2805 cm^{-1}, broken up into chunks that are 25 cm^{-1} wide. One hundred pressure layers are used to generate the database, from 1100 mb down to 0.005 mb. These pressure layers are the same as those used for the AIRS (Atmospheric InfraRed Sounder) Fast Forward Model, for which kCARTA is the "Reference Forward Model.". The temperatures in the spectroscopic database are from the U.S. 1962 Standard Profile, as well as ten temperature offsets (in increments of ± 10K) on either side of the Standard Profile. The current spectroscopic compressed tables use the HITRAN98 database for both line-parameters and cross-sections. The full and first-order CO_2 linemixing is from refining the modeling undertaken by David Tobin. It should be more accurate than that currently in GENLN2. in addition, we have used the latest O2 and N2 continuum models (see Lafferty and J.-M. Hartmann et al in Applied Optics 1996, 1997). Other updates to spectroscopy include the "local" water lineshape as defined by CKD, and the CKDv0, 2.1, 2.3 and 2.4 water continuums; in the future we will also add in the χ function developed by D. Tobin.

kCARTA is not limited to these pressure levels/layers. $klayers.x$ is a code, supplied separately from the kCARTA distribution, that changes a user supplied point profile to a kCARTA layers averaged profile. The default behaviour is to use to predefined 100 AIRS layers. However, the savvy user can use $klayers.x$ to change the pressure levels; these changes then percolate back to kCARTA.

To maintain compatibility with the AIRS products, for which kCARTA has been developed, the RTP file format and associated libraries need to be linked into kCARTA. These files can contain the AIRS profile and the atmosphere definitions (such as surface pressure and temperature, solar reflectance and surface emissivity). kCARTA allows the user to either completely or partially use the information in these files.

WARNING : If you wish to use RTP, your compiler needs to be able to support structures. Currently Absoft, PDF and SGI do, but g77 does not

In addition, we are adding on more features onto the existing radiative transfer algorithm. Currently, it can compute the radiances for either an upward or downward looking instrument, in a clear atmosphere. The top of the atmosphere, and the earth's surface, can be arbitrarily set within the pressure level extremes. v1.04 allows the user to compute the upward and downward fluxes. Two well known scattering packages, one developed by Frank Evans *et. al RTSPEC* and the other by Knut Stamnes et al *DISORT* has been worked into the kCARTA algorithm. In addition we have written our own twostream scattering code *TWOSTREAM*.

We are also working on a perturbative solution to the radiative transfer equation, but it has stalled (till Mathematica can solve one of the integrals). In addition we have used the PCLSAM (Parameterization of Cloud Longwave Scattering for use in Atmospheric Models) implementation of Chou, Lee, Tsay and Fu's method.

To run any of these scattering codes, one needs the MIE scattering tables generated by F. Evans "sccatmie" code. While DISORT is very well tested, it is much slower than RTSPEC. RTSPEC runs very fast, almost as fast as nonscattering kCARTA, but cannot include the effects of a solar beam. *TWOSTREAM* is a twostream code that runs fast and can include solar beam effects. *PCLSAM* also

runs very fast and allows a soalr beam.
WARNING : If you wish to use DISORT, then your compiler must have the autodoubling option, so as to turn REAL variables into DOUBLE PRECISION variables. If not, then edit file "scatter_disort_code.f" and change all occurences of real to double precision

The kCARTA Makefile produces two different executables.

- *kcarta.x* contains all the features of kCARTA (optical depths, clear sky radiative transfer, clear sky jacobians and fluxes, *SCATTERING* radiances and fluxes. However, this version uses a *LOT* of memory
- *bkcarta.x* contains only the most basic features of kCARTA (optical depths and clear sky radiative transfer). This version uses very little memory.

The speed and features of the code make it an appealing alternative to other existing "line by line" codes such as GENLN2 and LBLRTM. The accuracy of the database has been extensively compared to GENLN2. kCARTA should contain the latest spectroscopy/lineshape information. Furthermore, the code now allows the user to input a set of externally computed absorption spectra, at various kCARTA chunks, for one gas. The transmittances computed by kCARTA are smooth and well behaved, which will allow people to develop fast-forward models.

kCARTA is comparatively easy to use. After first picking up and compling the *RTP* and *KLAYERS* packages, kCARTA can be compiled, with the right array sizes, paths to include files and libraries being set in files INCLUDE/kcarta.param, INCLUDE/pre_defined.param, INCLUDE/post_defined.param and Makefile. Run time options are set from one driver namelist input file. We now have gone away from the GENLN2 style driver input files. kCARTA is now namelist file driven. However, for users that still wish to use the older GENLN2 style driver input files, we have provided a translator code between the old input file and the new namelist (only supported upto v1.09).

The driver namelist file contains information such as start/stop frequency, atmosphere defining information, gas ID's to be used, and so on. A layer profile file will also be read in at this point. Once the kCARTA run is complete, supplied FORTRAN or MATLAB files can be used to read in the data.

kCARTA has been written so that it can process one profile at a time. (Note that while an RTP file can contain more than one profile, kCARTA can only process one of the profiles per run). Once this profile has been read in, a well written driver input namelist file can be used to perform many tasks in one run. For example, it can be used to compute radiances for many different combinations, such as with or without solar radiation, with or without background thermal radiation, different surface temperatures or spectrally varying emissivities, different start/stop pressures and so on. For each of these combinations, clear sky radiance Jacobians can be computed and output. In the same run, kCARTA can be used to output layer-to-space transmittances for various gas combinations, such as all gases combined, all gases except water, all gases except ozone and so on. The user can also choose to output the optical depths of any of the gases read in from the profile. Once a kCARTA run has ended, the user can read in the data using the supplied readers, or modify the readers to convolve the data over wanted instrument response functions.

2 Installing and running kCARTA

This is for the user that wants to install and use kCARTA as quickly as possible.

2.1 Distributing kCARTA

The distribution is divided into three parts :

- Main tarfile *kcartaYYY.tar* where YYY is the version number. This will contain the entire source code distribution (which includes the radiative transfer code kCARTA, and some Matlab and Fortan readers). There are also some additional data files, and the documentation. This is about 80Mb

- Bugfix tarfile *kcarta_bugfixYYY.tar* where YYY is the version number. This will contain the bugfixes, mainly to the radiative transfer code kCARTA. This tar file is about 20Mb

- kCompressed Database : about 600Mb, supplied on CDs. We supply two versions, the big endian or the little endian versions

2.2 Installing kCARTA

Having obtained the above three, the user can now proceed to install kCARTA:

- Untar *kcartaYYY.tar* : this will create a main subdirectory, named kCARTA, as well as many subdirectories containing the source code, scripts, data files and so on.
- Untar *kcarta_bugfixYYY.tar* : This will overwrite the relevant source code files with the latest ones that are deemed bugfree.
- Install datafiles : Under the KCARTA/DATA subdirectory, the user can install (or provide symbolic links to) the *CompDataBase* and *WaterDataBase* files provided in the diskettes containing the kCompressed Database
- Create a WORK directory : Create directory WORK under the KCARTA directory

2.3 Compiling the packages

Now the source code compilation can begin. There are three main sets of code that have to be compiled. We will decribe the compilation assuming the user is going to use the 100 AIRS pressure layers/levels, instead of changing the pressure levels. Before compiling each of the three packages described below, the user should edit the relevant Makefile so that his/her preferred F77 compiler will be called. We have tested the package mainly on Unix style SGI and Absoft machines. The flags for these compilers are

```
# SGI Fortran
# ------------
# SGI compiler options
# -u  : turn off implicit typing of variables
# -g  : generate debugging information (turns off optimization)
# -C  : do run time subscript range checking
# -w0 : inform about unused variables
# -O3 : heavy optimization
# -64 : 64-bit objects (libraries must match)

# Linux with Absoft Fortran
# --------------------------
# Absoft compiler options
# -f    fold all names to lower case
# -N109 fold all names to upper case
# -W    wide source file
# -w    suppress warning messages (absoft is more fussy than SGI or g77)
# -A    alignment warnings
# -C    check array bounds
# -O    some optimizations
# -N3   add record info to unformatted files
# -s    static allocation
# -N2   force intrinsic double functions
# -N113 force double precision
# -N114 force untyped variables as warnings
```

If Absoft is used, since it is quite fussy, there might be a few warning messages produced during the compilations.

- SRC symbolic link : At the main KCARTA subdirectory, make sure that the "SRC" symbolic link points to "SRCv1.11"
- BLAS library : If your computer has a BLAS library, you can skip this part. Else go to the KCARTA/LIB/blas.ref subdirectory, and type "make" (at present the Makefile assumes a LINUX Absoft F77 compiler.
- klayers Go to KLAYERS/Src. Edit the "Makefile" so that the compiler options are those that exist on your machine. Type "make". This should produce a file klayers.x.
- kcarta The kCARTA Makefile can make one or all of three different executables. For the basic distribution that will satisfy most people, only bkcarta.x is produced. Go to KCARTA/SRC. Edit the "Makefile" so that the compiler options are those that exist on your machine. Edit the little endian vs big endian part of the "kcarta.param" file, so that the correct paths are set for the data files (ensure that the kCompPath,kCO2Path, kCOusin_CO2Path and kWaterPath strings point correctly to the database files, while kCKDPath,kSolarPath and kXsecFile point to the correct be vs le files). Type "make". If everything compiles ok, file bkcarta.x will appear in the KCARTA/BIN directory
  - *kcarta.x* contains all the features of kCARTA (optical depths, clear sky radiative transfer, clear sky jacobians and fluxes, *SCATTERING* radiances and fluxes. However, this version can use a *LOT* of memory

  - *bkcarta.x* contains only the most basic features of kCARTA (optical depths and clear sky radiative transfer). This version uses very little memory.

- readers with fseek Go to KCARTA/UTILITY. If your compiler supports the “fseek” subroutine, type “make” at the prompt. If everything compiles ok, file readkcBasic.x will appear in the KCARTA/BIN directory. Note that Absoft F77 can act funny with the fseeks, so it might be better to use the g77 compiler here. If this is the case then edit files *readkcBasic.f* and *readbinary.f*, replacing iDummy = fseek(,...) with CALL fseek(,...) with as well as removing FSEEK from the local variables declarations

- readers without fseek Go to KCARTA/UTILITY/READwoFSEEK. If your compiler does not support the “fseek” subroutine, type “make” at the prompt. If everything compiles ok, file readkcBasic.x will appear in the KCARTA/BIN directory.

2.4 Checking the kCARTA installation

If all this has been accomplished successfully, the user is now ready to try running kCARTA. Change to the KCARTA/SCRIPTs directory and type

basic.sc sondeprofile outputfile

Here *sondeprofile* is the name of the input radiosonde profile. This input profile will be processed through *klayers.x*, producing a layer averaged profile *kcarta.op.rtp* in the KCARTA/WORK subdirectory. After that, *bkcarta.x* runs, reading in namelist file *basic.nml* (see below) and profile *kcarta.op.rtp*. The output of *bkcarta.x* is stored in a large binary file, *outputfile*. This binary output file consists of a header, which contains information that determine the overall size of the file. After this comes the actual binary data. Since kCARTA works in chunks of 10000 points, all the outputs for the first 10000 points are written out, followed by the outputs for the next 10000 points and so on.

As an example to use this script, type

basic.sc BASIC/USStandardProf_NEW ../WORK/out.dat

To read the contents of this file, one can use either the *readkcBasic.x* (Fortran) or the *readkcBasic.m* (Matlab) readers. These readers go ahead and "unchunk" the data, as shown in the figure below.

As would be evident from reading the more extensive documentation, kCARTA works in chunks of 10000 points. For example, suppose the user wants 100 mixed paths to be output from 605 to 705 cm^{-1} . These mixed paths, 1-100, in region 605-630 cm^{-1} are processed and output first. These are followed by the 100 mixeed paths in region 630-655 cm^{-1}, and so on.

However, most users would want the data in the following format. Mixed path 1, from 605 to 705 cm^{-1}, followed by mixed path 2 in this complete wavenumber interval, and so on, uptil mixed path 100. As seen from the Figure 1, the *readkcBasic.x* (Fortran) and the *readkcBasic.m* (Matlab) readers "reshape" the matrices that kCARTA outputs, to be in this more convenient format.

basic.sc calls *readkcBasic.x* and translates file $../WORK/out.dat$ (on the left) to a very simple binary file $../WORK/outfile.bin$ (on the right of the figure). The only possible problem is that to make this Fortan reader work fast, we have used the nonstandard "fseek" option. This seems to work fine with SGI Fortran and g77 on Linux, but causes grief if used with Absoft.

2.5 Checking the kCARTA installation

The format of the final binary file (shown on the right in the figure above) is very simple :

- iSetMin : 1 integer, stating starting index (= 1)
- iSetMax : 1 integer, stating ending index (= 10000 x iNumChunks)
- iTotal : 1 integer, telling how many outputs per chunk
- Freqs : 10000 x iNumChunks, giving the kCARTA wavenumbers (reals)
- Data : (10000 x iNumChunks) x iTotal, giving the data (reals)

If $10000xiNumChunks = iX$ then the $Freqs$ are output in one Fortran data block. Similarly the $Data$ is in iTotal rows, each of which contain iX elements. So the whole file can be read using a simple program such as

```
integer iOUN,iSetMin,iSetMax,iTotal,iX,iR,iFr,iRows,iChunks
```

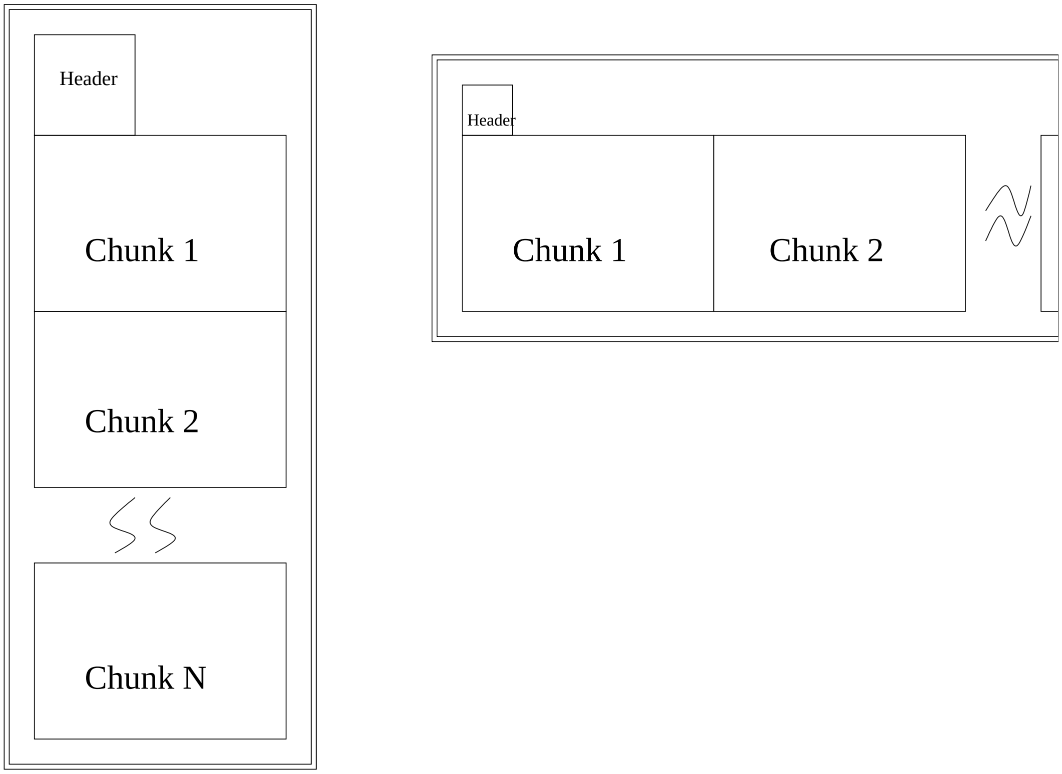


Figure 1: Translating the output file.

```
real raWaves(10000*iChunks),raaData(iRows,10000*iChunks)

read(IOUN) iSetMin
read(IOUN) iSetMax
read(IOUN) iTotal
iX = iSetMax - iSetMin + 1
read(IOUN) (raWaves(iFr),iFr = 1,iX)
DO iR = 1,iTotal
  read(IOUN) (raaData(iR,iFr),iFr = 1,iX)
  END DO
```

Typically, for a 100 layer atmosphere, iTotal is about 500 (five sets of mixed paths), while iX is an integer multiple of 10000 points (eg for a typical AIRS channel, this would probably span one or two kCARTA chunks, which would be about 10000 or 20000 points).

3 A Walk Through a kCARTA Session

kCARTA is packaged so that it is easy to set up and start using. In order to run the code, the user has to unzip and untar the distribution; this automatically sets up the correct directory structure. The file README.1ST describes the steps necessary to compile the source code for kCARTA and for the various UTILITY files we include with the package. After this is done, a script file can be run, so that the user can check to see if his/her setup obtains the same test results as we supply in the COMPARISON subdirectory.

At present, the 600 Mb kCompressed Database is distributed on tape. In the future, we plan to distribute the kCARTA package on CD in two parts; the SVD compressed database and a zipped tar file of the source code. The user has to put the database into the DATA/CompDataBase, DATA/WaterDataBase subdirectories. At this point, the user can ensure that he/she has the correct summary of the files in the database. This is achieved by a run of program compdatabase.x, which produces a parameter file comp97.param summarizing the database. compdatabase.f is one of the utility programs supplied along with the kCARTA package, and is found in the UTILITY subdirectory. Typing “make” in that subdirectory will compile the source files and place them in the binary subdirectory BIN.

The user can now go to the source subdirectory SRC, and use the makefile there to compile the main source code. Assuming the BLAS routines (if needed) have also been successfully compiled and can be linked in, the code is now ready to be run. All it requires are driver namelist input files.

We now give an overview and example of the driver namelist file; a more detailed view will be given in subsequent sections. The example has purposely been written so that kCARTA does a fairly comprehensive demonstration of its abilities—radiances for downlooking and uplooking instruments, solar and thermal background radiation alternately turned on or off, compute jacobians and so on. (Of course, this means the output file from this run will be quite large!). All the lines that begin with `!` are comment lines, and will explain to the reader what the different inputs are. A brief description of the purpose/effects of the file is as follows. The first section asks kCARTA to use water continuum version CKDv2.1, as well as to output all spectra as layer-to-space transmittances. The frequency region is from 2705 to 2780 cm^{-1}. We have chosen to explicitly specify which gasIDs are to be used from the compressed database, while the program is asked to include the required gases automatically from the crosssection database. kCARTA is asked to read in supplied regression profile #1, which happens to be the AFGL tropical (hot/humid) profile.

Note that in this example, we have turned on the water continuum (kCKD=21), and so we explicitly ask kCARTA to include gases 101,102 which are the self and foreign water continuums.

In this example the gas absorption coefficients will be weighted and combined in two different ways, one where all gases are equally weighted by unity, the second where water (and self and foreign continuums) and ozone weights differ slightly from unity. We can use these weights to define three separate atmospheres, for the program to do radiative transfer calculations. The first two atmospheres are defined so that the high-in-the-sky instrument is downward looking (in addition to the surface and layer emission terms, one atmosphere has both solar and background thermal turned on, while the other only has background thermal turned on). The third atmosphere is for an upward looking instrument that is close to the ground, with the sun filling the field of view. Jacobians will be computed and output for all the three atmospheres.

We have now added on scattering routines and the ability to import line by line spectra computed by an external code. In the example below, we use externally generated spectra for some CO2 regions and for some N2O regions. In addition, a two layer cloud is also specified.

The output section asks kCARTA to output radiances for all three atmospheres (at different pressure levels), output all layer-to-space transmittances, and output gas spectra for GasID=2.

Note that kCARTA will not really accept this namelist file, as it asks for Jacobians to be computed, in addition to scattering computations and importing external LBL spectra. In addition, it will complain as the output section asks for multi level radiance outputs. If you turn off the Jacobians, as ask for radiances to be output only at one level for each atmosphere, the namelist driver file will be successfully parsed in.

Here is the sample namelist river file (which we name example.nml) :

```
$nm_params
namecomment    =  ’******* PARAMS section *******’
!set this to output layer to space transmittances
kLayer2Sp = 2
!set this to v2.4
kCKD      = 24
!set kRTP = 0 so we read only the profile from the RTP file
kRTP = 0
$end

$nm_frqncy
namecomment    =  ’******* FRQNCY section *******’
rf1            = 905.000
rf2            = 1005.000
$end
$nm_molgas

namecomment    =  ’******* MOLGAS section *******’
!use  water, carbon dioxide, ozone, nitrous oxide,  methane, HCl and H2CO
! as well as the self and foreign continuums
iNGas     =            9,
iaGasesNL = 1, 2, 3, 4, 6, 15, 20, 101, 102
$end

$nm_xscgas
namecomment    =  ’******* XSCGAS section *******’
!use all minor cross section gases
iNxsec =            -1
$end

$nm_prfile
namecomment    =  ’******* PRFILE section *******’
iBinOrAsc    = 1
iMPSetForRadRTP = 1
iRTP            = 1

!this case assumes info for 2 clouds is given as slab info in caPfname
iNclouds_RTP  = 2
caCloudPFname = ’dne’
```

```
!this case assumes info for 2 clouds is given as profile info in caCLoudPfname
iNclouds_RTP  = 2
caCloudPFname = ’try_klayers_cloud.op.rtp’

iaCloudFile(1)   =  201
iaCloudFile(2)   =  202

iaCloudType(1)   =  100
iaCloudType(2)   =  200

raCloudDME(1)    =  10.0
raCloudDME(2)    =  30.0
caaCloudFile(1)  = ’MIEDATA/WATER250/water_405_2905_250’
caaCloudFile(2)  = ’MIEDATA/CIRRUS/cirrus_405_2905_220’

caPfname       =  ’testprof0.op.rtp’
$end

$nm_weight
namecomment    =  ’******* WEIGHT section *******’
!define 2 sets of mixed path sets
iNpmix =             3
caaMixFileLines        =
!first set has all gases weighted by 1.0
                         ’1   -1    1.0    -1’,
!second set has all gases weighted by 1.0, except water and CO2
                         ’2   -1    1.0    4’,
                            ’1 1.12 3 1.03 101 1.12 102 1.12’,
$end

$nm_radnce
namecomment    =  ’******* RADNCE section *******’
!define 3 atmospheres
iNatm          =  3

!use WEIGHT set number 1 (from above, this is all gases weighted by 1.0).
!The pressure endpts are from 971 to 0.0 mb (so radiation is travelling
!upwards, to the down looking instr at 0.0 mb)
!space temp=2.96K, surface temp=308.34k,
!satel view angle=20 degrees, do not include geometry effects in angle
iaMPSetForRad(1)   =  1
raPressStart(1)    =  971.7370000
raPressStop(1)     =  0.0
raTSpace(1)        =   2.960000
raTSurf(1)         =  308.3
raSatAngle(1)      =  0.0000000E+00
raSatHeight(1)     =   1.00
! turn off solar (-1), solar angle=0.0, use surface emiss(-1.0),
! turn on thermal diff approx, (0), thermal angle=53 degrees (-1),
!include thermal background if Jacobians are computed (1)
iakSolar(1)        =  -1
rakSolarAngle(1)   =  0.0000000E+00
cakSolarRefl(1)    =  ’NONESPECIFIED’
raKSolarRefl(1)    = 0.1234
iakThermal(1)      =   0
rakThermalAngle(1) =  -1.000000
```

```
iakThermalJacob(1) =   1
!use the following data file for spectral dependent surface emissivities
caEmissivity(1)    =   '../DATA/General/emissivity.dat'
raSetEmissivity(1) =  -1.000000

!use WEIGHT set number 2 (from above, this is all gases weighted by 1.0,
!except water and ozone). The pressure endpts are from 1200.0
!to 10.0 mb (so radiation is travelling upwards, to the down looking instr)
!space temp=2.96K, surface temp=303.34k,
!satel view angle=0 degrees, do not include geometry effects in angle
iaMPSetForRad(2)   =  2
raPressStart(2)    =  1200.0000
raPressStop(2)     =  10.0
raTSpace(2)        =   2.960000
raTSurf(2)         =  303.34
raSatAngle(2)      =  0.0000000E+00
raSatHeight(2)     =   -1.000000
!turn on solar (1), solar angle=0.0, use surface emiss (-1.0),
! turn on thermal accurate quadrature(1), thermal angle=53 degrees (-1 says
! use this default), include thermal background if Jacobians are computed (1)
iakSolar(2)        =  1
rakSolarAngle(2)   =  0.0000000E+00
cakSolarRefl(2)    =  '../DATA/General/sunrefl.dat'
raKSolarRefl(2)    = -1
iakThermal(2)      =   1
rakThermalAngle(2) =  -1.000000
iakThermalJacob(2) =   1
!use the following data file for spectral dependent surface emissivities
caEmissivity(2)    =   '../DATA/General/emissivity.dat'
raSetEmissivity(2) =  -1.000000

!use WEIGHT set number 1 (from above, this is 2 -1 1.0 -1, which means all
! gases weighted by 1.0). The pressure endpts are from 0.0
! to 985.0 mb (so radiation is travelling downwards, to the up looking instr)
!space temp=5600k (iakSolar = +1, sun fills FOV), dummy surface temp=303.34k,
!satel view angle=10 degrees,  include geometry effects in angle
iaMPSetForRad(3)   =  1
raPressStart(3)    =  0.0000
raPressStop(3)     =  985.0
raTSpace(3)        =  2.96.0000
raTSurf(3)         =  303.34
raSatAngle(3)      =  10.0000000E+00
raSatHeight(3)     =   705.0
! as we have upward looking instrument, just give six dummy values
iakSolar(3)        =  1
rakSolarAngle(3)   =  0.0000000E+00
cakSolarRefl(3)    =  'dummysun'
raKSolarRefl(3)    = -1
iakThermal(3)      =   0
rakThermalAngle(3) =  -1.000000
iakThermalJacob(3) =   1
! as we have upward looking instrument, just give a dummy emissivity
caEmissivity(3)    =   'dummy'
raSetEmissivity(3) =  0.9
$end
```

```
$nm_jacobn
namecomment    =  '******* JACOBN section *******'
!NOTE : THIS TELLS YOU HOW TO ENTER JACOBIAN SECTION; SINCE "SCATTER" AND
!"SPECTRA" BELOW ARE ON, THE CODE WILL NOT ALLOW YOU TO COMPUTE JACOBIANS!

!if you want to turn off NEWGASES , simply put iNumNewGases = -1 BELOW
!if you want to turn off SCATTERING, simply put kScatter = -1 BELOW

!compute two gas jacobians
iJacob =            2
!the gas IDs are 2,6  (co2 and n20)
iaJacob  = 2,6
$end

$nm_spectr
namecomment    =  '******* SPECTRA section ******'
!if you want to turn this off, simply put iNumNewGases = -1

!use externally generated LBL spectra for two gases
iNumNewGases   =            2

!the first gas has ID = 2 (CO2), with 6 new chunks of data
iaNewGasID(1)  =            2
iaNewData(1)   =            6
  iaaNewChunks(1,1)    =           2305
  iaaNewChunks(1,2)    =           2330
  iaaNewChunks(1,3)    =           2355
  iaaNewChunks(1,4)    =           2380
  iaaNewChunks(1,5)    =           2405
  iaaNewChunks(1,6)    =           2430
  caaaNewChunks(1,1)    =   '../DATANEW/co2_2305'
  caaaNewChunks(1,2)    =   '../DATANEW/co2_2330'
  caaaNewChunks(1,3)    =   '../DATANEW/co2_2355'
  caaaNewChunks(1,4)    =   '../DATANEW/co2_2380'
  caaaNewChunks(1,5)    =   '../DATANEW/co2_2405'
  caaaNewChunks(1,6)    =   '../DATANEW/co2_2430'

!the second gas has ID = 4 (N20), with 2 new chunks of data
iaNewGasID(2)  =            6
iaNewData(2)   =            1
  iaaNewChunks(2,1)    =           2255
  iaaNewChunks(2,1)    =           2280
  caaaNewChunks(2,1)     =   '../DATANEW/nitrousoxide2255'
  caaaNewChunks(2,1)     =   '../DATANEW/nitrousoxide2280'
$end

$nm_nonlte
namecomment    =  '******* NONLTE section ******'
!!!turn on nonlte
iNumNLTEGases    =            +1

!!use the slow accurate LBL model
iNLTE_SlowORFast =            +1

iDoUpperAtmNLTE    = +1
caaUpperMixRatio = '/home/sergio/KCARTADATA/General/NLTE/atm_md.ip'
```

```
iAllLayersLTE      = -1
iUseWeakBackGnd    = +1
iSetBloat          = +1

iaNLTEGasID(1)      =       2
raNLTEstrength(1)   =       1.000
raNLTEstart(1)      =       30.0
iaNLTEChunks(1)       =       10

!comment out this so kCARTA finds optimum profile!!!!!
!!!caaNLTETemp(1)     =
!!!  ’../SRC/NONLTE2/sergio/VT_48PROFILES/sergio_merge/vt5_s0.prf’
caaStrongLines(1) =
    ’/carrot/s1/sergio/AIRSCO2/CO2_BANDS_PARAM/co2_4um_bands.txt’

iaaNLTEChunks(1,1)  =        2230
iaaNLTEChunks(1,2)  =        2255
iaaNLTEChunks(1,3)  =        2280
iaaNLTEChunks(1,4)  =        2305
iaaNLTEChunks(1,5)  =        2330
iaaNLTEChunks(1,6)  =        2355
iaaNLTEChunks(1,7)  =        2380
iaaNLTEChunks(1,8)  =        2405
iaaNLTEChunks(1,9)  =        2430
iaaNLTEChunks(1,10) =        2455

iaNLTEBands(1)     = 19
!!! uses strongest  sigma-sigma, pi-pi, delta-delta
!!! 2350 .. 2354 = sigma-sigma
!!! 2320 .. 2322 = pi-pi
!!! 2310 .. 2312 = delta-delta
!!!                 GASID   GASIso  iLSGQ     iUSGQ   run7lblID
caaaNLTEBands(1,1) =’2       1       1         9       2350’
caaaNLTEBands(1,2) =’2       2       1         9       2351’
caaaNLTEBands(1,3) =’2       3       1         9       2352’
caaaNLTEBands(1,4) =’2       4       1         9       2355’
caaaNLTEBands(1,5) =’2       1       2         16      2320’
caaaNLTEBands(1,6) =’2       2       2         16      2321’
caaaNLTEBands(1,7) =’2       1       4         24      2310’
caaaNLTEBands(1,8) =’2       2       4         24      2311’
caaaNLTEBands(1,9) =’2       1       3         23      2353’
caaaNLTEBands(1,10)=’2       1       5         25      2354’
!!!these are the ones Manuel suggested adding on; some isotopes of above
caaaNLTEBands(1,11)=’2       2       3         23      2253’
caaaNLTEBands(1,12)=’2       2       5         25      2254’
!!!these are the others Manuel suggested adding on
caaaNLTEBands(1,13)=’2       1       2         15      2110’
caaaNLTEBands(1,14)=’2       1       3         25      2120’
caaaNLTEBands(1,15)=’2       1       5         23      2140’
caaaNLTEBands(1,16)=’2       1       6         36      2160’
caaaNLTEBands(1,17)=’2       1       7         37      2170’
caaaNLTEBands(1,18)=’2       1       8         38      2180’
caaaNLTEBands(1,19)=’2       3       2         16      2322’
!!!these one never seems to exist in the NLTE profiles
caaaNLTEBands(1,20)=’2       1       3         22      2150’
caaaNLTEBands(1,21)='2        1       4         22       2130'
$end

$nm_scattr
namecomment    =  '******* SCATTR section *******'
!if you want to turn this off, simply put iNClouds = -1

!use DISORT
kWhichScatterCode = 3

!instead of doing scattering at all 10000 pts, do it at 25 points and
!then scale the results according to ``k'' of lowest layer
kDis_Pts         =      25

!number of DISORT streams
kDis_Nstr         =      16

!use ``step over wavenumber points'' method
kScatter         =           1

!use Mie data in text file format
iScatBinaryFile  =             -1

!there are two clouds in the atmosphere
iNclouds         =             2

!first cloud, when expanded,  occupies three AIRS layers
iaCloudNumLayers(1)    =             2
raExp(1)               = 0.0
caaCloudName(1)   =  'HappyLittleCloud'
  !layer 1 is from 190 to 180 mb; IWP=5 g/m2   <dme>=50 um
  raaPCloudTop(1,1) =  1.8000000E+02
  raaPCloudBot(1,1) =  1.9000000E+02
  raaaCloudParams(1,1,1)       = 5.0
  raaaCloudParams(1,1,2)       = 50.0
  !associate table # 1 with it; data is in file cirrus.cloud.layer1
  iaaScatTable(1,1)    =  1
  caaaScatTable(1,1)   =  'cirrus.cloud.layer1'

  !layer 2 is from 220 to 190 mb; IWP=6 g/m2   <dme>=60 um
  !this wil occupy more than one AIRS layer ie will be expanded
  raaPCloudTop(1,2) =  1.9000000E+02
  raaPCloudBot(1,2) =  2.2000000E+02
  raaaCloudParams(1,2,1)       = 6.0
  raaaCloudParams(1,2,2)       = 70.0
  !associate table # 2 with it; data is in file cirrus.cloud.layer2
  iaaScatTable(1,2)    =  2
  caaaScatTable(1,2)   =  'cirrus.cloud.layer2'

!this cloud will be used in the rad transfer for TWO atmospheres defined
!in RADNCE above : atmospheres numbers 1 and 2
iaCloudNumAtm  =             2
iaaCloudWhichAtm(1,1)      =     1
iaaCloudWhichAtm(1,2)      =     2

!!!aerosol at boundary layer
```

```
iaCloudNumLayers(2) = 1
raExp(2) = +3.0
caaCloudName(2)='happy little boundary layer aerosol'
  !cloud is from 970 to 960 mb, and so will be expanded
  !also, as raExp(2) = +3, the IWP in the layers will be exponentially scaled
   raaPCloudTop(2,1)   = 970.0
   raaPCloudTop(2,1)   = 860.0
   raaPCloudBot(2,1)   = 970.0
   raaaCloudParams(2,1,1) =  2.5e-1
   raaaCloudParams(2,1,2) =  0.2e+0
   iaaScatTable(2,1)=2
  caaaScatTable(2,1)='aerosol.layer'
  iaCloudNumAtm(2)=1
  iaaCloudWhichAtm(2,1)=1

$end

$nm_output
namecomment    =  '******* OUTPUT section *******'
caComment      = 'output for sample file in documentation'

!notice how we first define paths to be output, then MPs, then radiances

!for GAS ID=2, output gas path spectra at all layers. Thus the kProfLayer
!paths of carbon dioxide (gasID=2) will be output
iaPrinter(1)   =    1
iaAtmPr(1)     =    2
iaNp(1)        =    -1

!output all mixed path spectra. Thus from *WEIGHT, there are 300
!mixed paths that will be output
iaPrinter(2)   =    2
iaAtmPr(2)     =    -1
iaNp(2)        =    -1

!for first atm, output radiance at TOA
iaPrinter(3)   =    3
iaAtmPr(3)     =    1
iaNp(3)        =    1
raaOp(3,1)     =    0.0

!NOTE THAT SINCE WE HAVE TURNED ON SCATTERING CODE, WE CAN ONLY OUTPUT
!RADIANCES AT ONE LEVEL. IF WE WANT TO OUTPUT RADIANCES AT MANY LEVELS, THEN
!WE CANNOT HAVE SCATTERING
!TO TURN OFF SCATTERING, SET kScatter = -1 ABOVE

! output two radiances for the second atmosphere, at pressures 10 and 20 mb.
! because of the atmosphere definition, these are at the satellite posn, and
! slightly below the satellite position, respectively
iaPrinter(4)   =    3
iaAtmPr(4)     =    2
iaNp(4)        =    1
raaOp(4,1)     =    10.0

! output four radiances for the third atmosphere, at pressures
! 960.0 mb, 970.0 mb, 980 mb and 985.0 mb
! because of the atmosphere definition, the first three are on top of the
! satellite, while the fourth is at the satellite.
iaPrinter(5)   =    3
iaAtmPr(5)     =    3
iaNp(5)        =    4
raaOp(5,1)     =    960.0
raaOp(5,2)     =    970.0
raaOp(5,3)     =    980.0
raaOp(5,4)     =    985.0

$end

$nm_endinp
namecomment    =  '******* ENDINP section *******'
$end
```

To run the program, the user should type " kcarta.x example.nml out.dat outjacob.dat" Assuming that the driver file "example.nml" is in the same directory as "kcarta.x," this will make kcarta.x to read in the driver file, and then output the radiances/transmittances to be output to "out.dat" and the jacobians to be stored in "outjacob.dat." A simple flow diagram that pulls together the different files required to run kCARTA, is shown below.

Our supplied programs readkcarta.x (a FORTRAN reader) and readkcstd.m (a MATLAB reader) can now be used to read the data produced by kCARTA. The FORTRAN reader can be used to read in the header and *all* the data, and save to a simpler binary file that only contains the data. Or the FORTRAN reader can be used to save *one* of the paths/mixed paths/radiances to a two column text file, that can easily be read in graphics packages. Using this option, Figure 3 shows the radiance computed from the first print option (i.e. the radiance at the top of atmosphere number 1). Here we have combined the results of three runs, one with no scattering, and the next two with DISORT and RTSPEC used for an atmosphere where there is aerosol near the ground, and a cirrus cloud at 11 km.

4 New Spectroscopy used by kCARTA

Most of the compressed optical path database that is used by kCARTA, is generated by our own custom line-by-line code. This code is based on $GENLN2$. The main differences between our code and $GENLN2$ are

- $GENLN2$ divides the lines into "near" and "far" bins, while our code also uses a "medium" bin

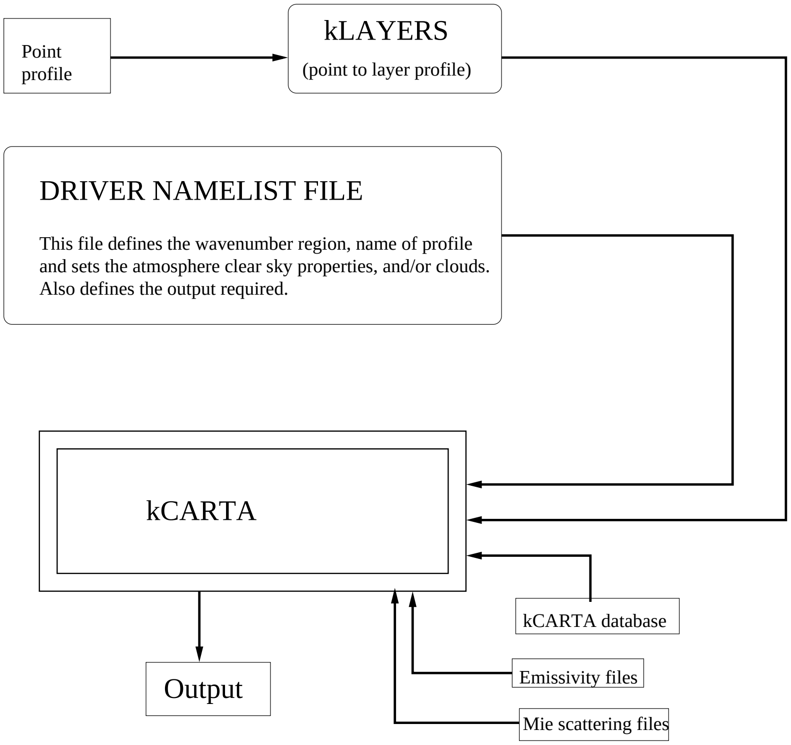

Point
profile
kLAYERS
(point to layer profile)
DRIVER NAMELIST FILE
This file defines the wavenumber region, name of profile
and sets the atmosphere clear sky properties, and/or clouds.
Also defines the output required.
kCARTA
kCARTA database
Emissivity files
Mie scattering files
Output

Figure 2: Files needed to run kCARTA.

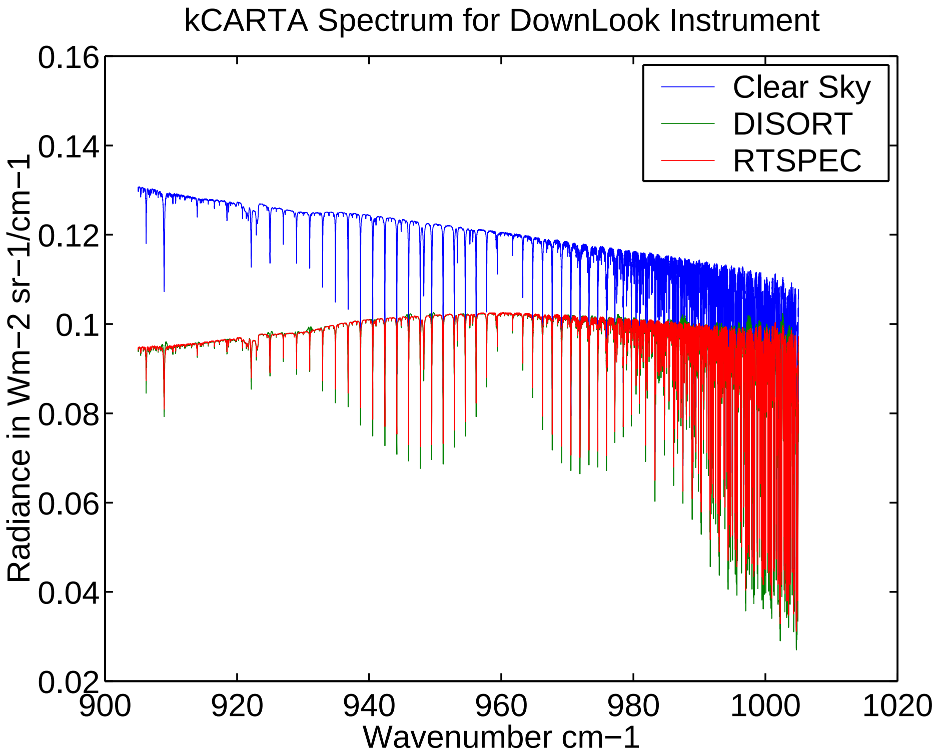


Figure 3: Sample output radiance spectrum.

- Our code uses Q branch line mixing, as well as P and R branch line mixing for the 4 and 15 micron CO2 optical depths

Some other features of the spectroscopy used by kCARTA also include our modifications to the water-self and -foreign continuums in the 1600 cm^{-1} bandhead. CKDv2.4 currently underestimates the continuum contribution in this region.

The spectroscopy used by kCARTA has been validated by comparing to data from campaigns such as CAMEX-1, WINTEX, ARIES and CLAMS. This variety means a number of different instruments, in differing atmospheric situations, all seem to imply that kCARTA is generally correct. In addition, we are now using data from the AIRS instrument, which is further helping us refine some of our spectroscopy. For example, we had previously noted that while the linemxing in the 15 micron region gave us very acceptable obs-calcs, there had always been a little trouble in the 4 micron bandhead. The AIRS data is helping us further refine our model, and we hope to come out with some updated corrections to the database very soon.

5 Units and Definitions

While reading this section, the user is reminded that kCARTA uses the 100 AIRS layers, spline interpolated onto MYNLAY layers. All angles set in the kCARTA files should be in degrees. Frequencies are in units of wavenumbers (cm^{-1}). In defining the atmospheres inside the *nm_radnce* section, pressure boundaries should be in millibars (1.0 atm=1013.25 mb). Surface and deep space temperatures are in Kelvin. If the user specifies pressures at which radiances are to be output, they should also be in millibars.

The gas profiles expected by kCARTA use the following units. The gas amounts are path averaged over the layers, and are in units of $kiloMoles\text{cm}^{-2}$. Temperatures should be specified in *kelvin*, while pressures and partial pressures should be expressed in *atmospheres*. Layer altitude (approximately at center of the layer) should be in *kilometers*. Each gas will have a kProfLayer layer profile. Note that if the user inputs a point profile in the appropriate format to our supplied kLayers program, the path averaged profile that is output by this program will be in the correct units.

Output gas and mixed path optical depths are dimensionless (absorption coefficient × gas amount); obviously so are transmittances. Output radiances are in blackbody radiance units ($milliwatts\ \ m^{-2}sr^{-1}/\text{cm}^{-1}$). Jacobians can be output

in one of three modes : an example is $d(rad)/ds_m$, where s_m is the temperature or gas amount in layer m. Fluxes are output in one of two modes : radiance * angle units, or $Kday^{-1}/cm^{-1}$
(note the smaller, second Jacobian file only dumps out radiances assuming the atmosphere was perturbed using a stemp/gas column perturbations).

The following terms will be found in various places throughout the document.

Table 1: Glossary of terms used in document

compressed coefficients	kCARTA uses a compressed database to quickly determine the absorption coefficients of most gases in the HITRAN database, for the required profile
xsec database	Some of the gases in the HITRAN database (notably the *CFCs*) are too complex to have their parameters determined experimentally. The absorption coefficients for these gases are computed by interpolating the measured coefficients at various temperatures and pressures
kMaxLayer	this is the number of AIRS layers that were used in producing the kCompressed Database (=100)
kProfLayer	our *kLayers* code allows the user to subdivide the lower AIRS layers or clump together the upper AIRS levels. This means the resulting number of layers could be lesser or greater than kMaxLayer, depending on how klayers.x is set up. This will be the actual maximum number of layers in each atmosphere for the particular kCARTA run

nm_params	some required parameters have default values. The user can change these values to other allowed values. Note these settings apply to the entire run. For example, if the user chooses to use water continuum version CKDv2.1, this will be used for all the mixed paths and atmospheres in the run.
nm_molgas	required section in kCARTA driver namelist file. This specifies which gases need to have their optical depths computed using the compressed database
nm_xscgas	optional section in kCARTA driver namelist file. This specifies which gases need to have their optical depths computed using the cross-sectional database (xsec)
path	the optical depth for a gas, in a particular layer. Each gas will have kProfLayer paths associated with it. If there are iNGas gases specified in *nm_molgas*, and iXsec gases specified in *nm_xscgas*, there will be a total of kProfLayer(iNgas+iNXsec) paths
nm_prfile	required section in kCARTA driver namelist file. This specifies which gas profile to use in the run. The profile would be in the *kLAYERS* output format. Each layer for each gas in this profile will constitute a gas path.
nm_weight	required section in kCARTA driver namelist file. This specifies the gas weightings (one weight per gas). This is a multiplier to the gas profile, applied equally to the gas in ALL *kProflayers*.

mixed path	When the paths for one gas are computed, as determined by the gas profile, the user can then combine different weights of this gas, with the optical depths of other appropriately weighted gases. In this way, the user can build up an atmosphere that consists of the gases specified in *nm_molgas* and *nm_xscgas*, weighted appropriately. As the paths are in sets of $kProfLayers$, so the mixed paths are also in sets of $kProfLayers$
mixing table	is a term used to describe how the weightings of various gases (from *nm_weight*) are added together cumulatively to obtain a set of mixed paths
atmosphere	Once the individual gas absorption coefficients have been combined to form sets of mixed paths (kProfLayers mixed paths per set), an atmosphere can be defined from any one of the sets. In addition, the boundary conditions of the atmosphere (start and stop pressures, surface temperature and emissivity) and the direction of radiation travel define individual atmospheres.
nm_radnce	optional section in kCARTA driver namelist file. Once the user has combined individual gas paths to form mixed paths, he/she can now define an atmosphere, as above. The radiance measured by an instrument anywhere within the boundaries of the atmosphere, can be calculated. The boundary conditions (eg surface temperature, upper and lower pressure boundaries) and direction of radiation travel are amongst other parameters the user specifies here

nm_jacobn	optional section in kCARTA driver namelist file. Once the user has defined an atmosphere, the sensitivity of the measured radiance to gas amounts and temperatures of the different layers can be studied by computing the Jacobians and weighting functions. Note these sensitivities are for clear skies only. At present, the code does allow the user to compute radiances using the $PCLASM$ scattering models, and then compute jacobians. In addition, since the code computes temperature jacobians while it is uncompressing the kCompressed Database, it cannot compute these jacobians if the spectra for some gases in a wavenumber interval are externally supplied.
nm_scattr	optional section in kCARTA driver namelist file. Once the user has defined an atmosphere, by default, radiance computations will assume a clear sky. However, the user can choose to include various clouds in each atmosphere, and use our interface to the scattering codes such as *rtspec.f*, *disort.f* or *twostream.f* or *pclsam.f* . If scattering is turned on, then jacobians will be be computed using the *pclsam* algorithm.
nm_nonlte	optional section in kCARTA driver namelist file. Once the user has defined an atmosphere, by default, radiance computations will assume a clear sky in LTE. However, the user can choose to include NONLTE computations for some of the upper layers of the AIRS atmosphere. Since this entails doing laborious line by line computations, the code slows down tremendously!

nm_spectr	This section allows the user to choose more than one gas, and have kCARTA read in LBL spectra produced by some other code for defined spectral regions. These regions have to overlap the kCARTA chunks eg 705-730, 1255-1280 etc If this option is used, then *nm_jacobn* cannot be used!! Also, the code will simply read in the spectra, and weight it appropriately ... it will not do checks . In other words, it is the users responsibility to make sure the spectra read in were computed with the correct layer amounts and temperatures.
nm_output	required section in kCARTA driver namelist file. The user can choose to output path or mixed path spectra, or radiances
nm_endinp	required section in kCARTA driver namelist file. Specifies end of driver namelist file
driver namelist file	this namelist file contains settings that are read in by kCARTA and then control the running of the code (namely, what output to produce)
iNatm	number of atmospheres defined in driver namelist input file

5.1 Note about computed fluxes

Here our definition of flux is integration of the upward radiance over incident angles minus integration of downward radiance over incident angles, at each boundary pressure level in the atmosphere. The difference between the upward and downward "fluxes" then give us the net heating of the layer, which is directly related to the temperature change per unit time. Refer to Liou, "An Introduction to Atmospheric Radiation", pg 107. If the atmosphere is defined between layers 3 and 65, then the flux differences (at 0.0025 cmresolution) between the top and bottom of each of layers 3-65 is output. In other words, the code does not sum over all the radiances across the infrared spectrum, but just outputs the monochromatic "flux". The user will read in this flux, and "add all the points, multiplied by 0.0025."

If the user supplies a name $filename$ for the usual radiance results to be output, then the flux results (kFlux = 1,2) are in $filename\_FLUX$ (or if kFLux = 3,4,5 $filename\_OLR$); otherwise they will stored in flux.dat. Note only if kFlux=1,2,3,4,5 and a radiance for an atmosphere is to be output, then the fluxes will be computed. This is independent of whether or not Jacobians are to be computed.

If kFlux=1, the fluxes are simply an integral of radiance over $2\pi\mu d\mu$, which means the units will be radiance units × steradians, or mW m-2/cm-1.

If kFlux=2, the fluxes are an integral of radiance over $2\pi\mu d\mu$, and then divided by $-c_p\ \ \rho\delta z$ where c_p is the specific heat at constant pressure, ρ is the density and δz is the layer thickness. This means the units will be $Kelvin$ per day percm^{-1}.

To get the same type of plots that are in Liou, "An Introduction to Atmospheric Radiation", pg 108, the user first has to read in the computed "fluxes"; then for each layer, sum across the monochromatic "fluxes", multiply the result by 0.0025 cm^{-1} (as this is the same as integrating $I(\nu)d\nu$) to get the correct flux units (the factor of 86400 has already been accounted for, so the units are "correct.").

If kFlux=3, only outgoing radiation at the top of each layer is computed; so the units are the same as for kFlux=1 case, namely mW m-2 per wavenumber point.

If the user is only interested in OLR at the top of the atmosphere, then kFlux=4 only outputs upgoing flux at TOA (thus making the binary output files approximately 100 times smaller than fluxes output at *all* layers).

If the user is only interested in OLR at TOA, tropopause and the ILR at GND, then kFlux=5 only outputs downgoing flux at GND and upgoing flux at tropopause and TOA (thus making the binary output files approximately 33 times smaller than fluxes output at *all* layers). This way the user can compute the greenhouse forcing parameter(s) [**?**, **?**]

$$g = \frac{\sigma T_{surf}^4 - OLR_{clearsky}}{\sigma T_{surf}^4}$$

$$G = \frac{\sigma T_{surf}^4}{OLR_{clearsky}}$$

For kFlux = 3,4,5 don't forget to multiply by d(nu) = 0.0025 cm-1 when doing the (Matlab) integral over frequency (and divide the result by 1000 to change from mW to W if kFlux = 1,3,4,5)

A sample flux plot is shown below, in Figure 4

6 Using KLAYERS : change point profile to layer averaged profile

See the KLAYERS documentation. It is supplied to default to the AIRS 100 layers but the user can rest the pressure levels as necessary.

6.1 Compiling kcarta

Once the user has successfully run klayers.x as described above, he/she should now go to the kCARTA/SRC directory.

The user now has to edit kcarta.param, and set kProfLayer to the same number of layers that $klayers.x$ can produce. Having done this, kCARTA is ready to be compiled (type "make") and then run.

7 Using RTSPEC : producing the Mie Scattering tables

If the user only wants to compute clear sky radiances, then this section can be skipped. However, this section must be read by someone interested in using kCARTA for scattering computations.

To do scattering computations, kCARTA has been interfaced with two extensively tested scattering routines : $RTSPEC$ and $DISORT$. While $RTSPEC$ [1] is very fast, it cannot be used when one has a beam source present (eg a solar beam). $DISORT$ [2] can handle this case; however it is much slower than $RTSPEC$. We have written our own $TWOSTREAM$ code that is fast and allows the user to use a solar beam. In addition we now have an interface to $PCLSAM$ which also

[1] M. N. Deeter and K. F. Evans, "A Hybrid Eddington Single Scattering Radiative Transfer Model for Computing Radiances from Thermally Emitting Atmospheres," *JSQRT* **60** 635-648 (1998)

[2] K. Stamnes, S-C. Tsay, W. Wiscombe and K. Jayaweera, "Numerically Stable Algorithm for discrete ordinate method Radiative Transfer in multiple scattering and emitting layered media," *Appl. Opt* **27**, No 12, 2502 (1988)

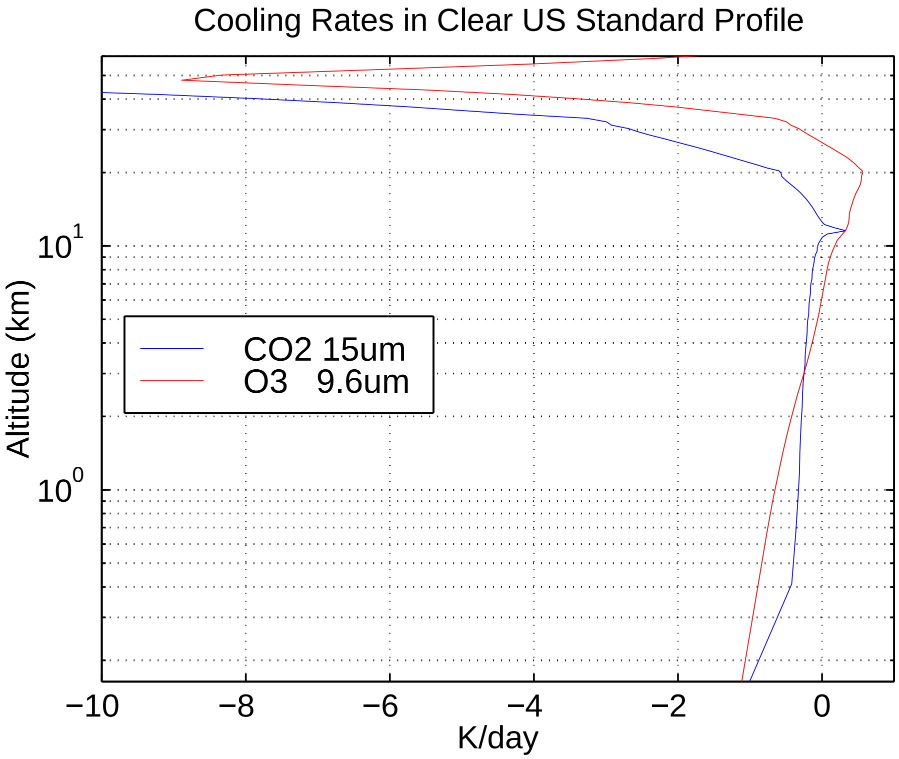


Figure 4: Sample flux computation.

allows the use of solar beam, and is very fast. Jacobians and fluxes can easily be computed with this algorithm.

To combine the advantages of both of these well documented and tested scattering codes, kCARTA has been interfaced to both these codes so that the same information is needed for either scattering routine. As *RTSPEC* needs some more specific information, we have chosen to include the scaled Mie properties computed by the sister code to *rtspec.f*, *sscatmie.f*. Given some general inputs, this code will compute extinction coefficients, single scattering albedos and asymmetry parameters that can be used by both codes, for ice or water clouds.

In addition, we have extended the capability of *sscatmie.f* by interfacing the aerosol package *OPAC* [3] to it. The addition of this database extends the capabilities of *sscatmie.f* to compute the same parameters for a variety of aerosol modes, at varying relative humidity values.

The original source code and documentation for some of the packages is easily obtained over the Internet :

```
RTSPEC : http://nit.colorado.edu/~evans/rtspec.html
DISORT : ftp://climate.gsfc.nasa.gov/pub/wiscombe/Multiple_Scatt/DISORT2.0beta/
OPAC : http://www.lrz-muenchen.de/~uh234an/www/radaer/opac.html
```

To produce the Mie scattering tables, go to the *../SCATTERCODE/RTSPEC* directory. Type “make” at the prompt to compile the code (at present the code has been tested on SGI Ultrix and Absoft F77 Linux). This should produce 5 executables :

- combine : this allows the user to “combine” a number of text output Mie files, into one single binary file
- translate : this allows the user to “translate” a text output Mie file, into a smaller binary file
- rtspec : the compiled “rtspec” code
- SSCATMIE : this is the executable the user will use to generate Mie scattering tables of ice, water clouds or aerosols

[3] M. Hess, P. Koepke and I. Schult, “Optical Properties of Aerosols and Clouds : The Software Package OPAC,” *Bull. Am. Met. Soc*, **79**, No 5, 831 (1998)

- refind : the user can use this to compute refractive indices of ice, water clouds or aerosols

The particle distribution assumed by *sscatmie* is a modified gamma distribution, where the user can vary parameter α (typically = 6):

$$n(r) = \gamma r^{\alpha} exp^{-br}$$

The peak of the distribution is at $x = \alpha/b$. Both b and r are in microns. The normalization term γ is related to the liquid water content ρ_{lwc} by

$$\rho_{lwc}(gm^{-3}) = \int (4/3)\pi r^3 \rho_{water} n(r) dr = \gamma \frac{4\pi}{3} \rho_{water} \frac{\Gamma(\alpha+4)}{b^{\alpha+4}}$$

while the average or effective radius of the distribution is given by

$$r_{avg} = \frac{\int n(r) r^3 dr}{\int n(r) r^2 dr} = \frac{\alpha+3}{b} = Dme$$

Depending on the grid of particle sizes chosen (see *Enter min and max Dme, and number of sizes* below), this are the values at which *sscatmie* reports results.

As *sscatmie* runs, it prompts the user to answer the follwoing questions :

```
Enter filename for output table
Fixed wavenos or evenly-spaced wavenos (F or E):
Enter min, max and number of wavenumbers (cm^-1)
Enter min and max Dme, and number of sizes
Enter alpha for particle distribution
Enter species ('I'=ice, 'W'=water, 'A'=aerosol):
If aerosol,
  Enter aerosol type and Relative Humidity
       0 : General NH4SO4, from Remote Sensing of Atm by G. Stephens
       1 : OPAC INSO : insoluble aerosol
       2 : OPAC WASO : water-soluble aerosol
       3 : OPAC SOOT : soot
       4 : OPAC SSAM : sea salt (accr mode)
       5 : OPAC SSCM : sea salt (coag mode)
       6 : OPAC MINM : mineral salt (nucleated mode)
       7 : OPAC MIAM : mineral salt (accr mmode)
       8 : OPAC MICM : mineral salt (coag mode)
```

```
        9 : OPAC MITR : mineral transported (desert dust)
       10 : OPAC SUSO : sulphate droplets
 Enter cloud temperature (Kelvin)
 If ice or aerosol
   Enter Volume fraction factor (a) and exponent (b) (vf=a*D^b, D in um)
 Enter E for evenly-spaced mu or Q for Lobatto-Quadrature mu
 Enter min and max mu, and number of angles
 Delta scale the scattering properties (Y, N, H, or G)
```

Typical answers for an atmosphere with sulphate droplets aerosol, are

```
/CLOUDS/aerosol.scatab805     !output file name
'E'                           !evenly spaced wavnos
805.0 1005.0 50               !start,stop waveno
0.1 100 10                    !particle sizes
6                             !alpha parameter, for the particle size distr
'A'                           !aerosol
 10  30.0                     !sulphates (OPAC file 10), at 30%RH
230.0                         !cloud temp (irrelevant)
1.0 0.0                       !volume fraction and exponent
'E'                           !evenly spaced mu (cos(x)) for quadrature
0.1 1.0 10                    !min(mu),max(mu) ... used by RTSPEC
'G'                           !delta scale to make things more accurate
```

Typical answers for a cirrus cloud in the troposphere (11 km), are

```
/CLOUDS/ice.scatab805         !output file name
'E'                           !evenly spaced wavnos
805.0 1005.0 50               !start,stop waveno
0.1 100 10                    !particle sizes
6                             !alpha parameter, for the particle size distr
'I'                           !aerosol
230.0                         !cloud temp (irrelevant)
1.0 0.0                       !volume fraction and exponent
'E'                           !evenly spaced mu (cos(x)) for quadrature
0.1 1.0 10                    !min(mu),max(mu) ... used by RTSPEC
'G'                           !delta scale to make things more accurate
```

The files produced by running the code can now be used by kCARTA

8 kCARTA source files

This section gives a brief description of the different files containing the source code for kCARTA.

- this set of continuum and cross section files will eventually be phased out, as we are moving towards using a kCompressed style format for the cross section gases, and a lookup table format for the CKD computations.
- calcon*.f : compute the water continuum (CKD v0,1,2,2.3,2.4 etc)
- calq.f : auxiliary file to compute cross-section absorption coefficients (gas IDs 51-63)
- calxsc.f : file to compute cross-section absorption coefficients (gas IDs 51-63)
- freqfile.f : kCARTA start/stop chunks, other misc stuff eg database freq checks for given gas/chunk combos and so on
- rtp_interface.f : interface with RTP libraries
- spline_and_sort.f : lots of functions; sorting, splines, mod, div,real2double etc
- h2oft0.f : water continuum data blocks, GENLN2 (lorentz) style
- h2ost0.f : water continuum data blocks, GENLN2 (lorentz) style
- h2ost1.f : water continuum data blocks, GENLN2 (lorentz) style
- h2oft0_wb.f : water continuum data blocks, correct (local) style
- h2ost0_wb.f : water continuum data blocks, correct (local) style
- h2ost1_wb.f : water continuum data blocks, correct (local) style
- rad_main.f : clear sky radiance calculations
- rad_diff.f : compute thermal background using diffusivity approx
- rad_quad.f : compute thermal background using quadrature
- rad_misc.f : solar and other misc radiance code

- rad_flux.f : clear sky flux code
- scatter_rtspec_main.f : interface to RTSPEC calculations
- scatter_rtspec_code.f : RTSPEC scattering code (from Frank Evans)
- scatter_rtspec_flux.f : flux code (totally incomplete!!!!)
- scatter_pclsam_main.f : interface to PCLSAM calculations
- scatter_pclsam_code.f : PCLSAM scattering code
- scatter_pclsam_flux.f : flux code
- scatter_disort_main.f : interface to DISORT calculations
- scatter_disort_code.f : DISORT scattering code (from K. Stamnes)
- scatter_disort_misc.f : couple of DISORT specific routines
- scatter_disort_flux.f : flux code (totally incomplete!!!!)
- scatter_twostream_main.f : interface to TWOSTREAM calculations
- scatter_twostream_code.f : twostream scattering code
- scatter_twostream_guts.f : twostream scattering code (contd)
- scatter_twostream_flux.f : flux code
- jac_main.f : main jacobian code
- jac_up.f : jacobian code for up looking instrument
- jac_down.f : jacobian code for down looking instrument
- kcartamain.f : main kCARTA source file
- kcartamisc.f : misc routines eg splines
- kcoeffMAIN.f : set up calls to compute LTE absorption coefficients
- kcont_xsec.f : set up calls to compute continuum, xsec

- kcoeffSPL.f : compute LTE absorption coefficients
- kcoeffSPLJAC.f : compute LTE absorption coefficients, do Jacobians
- knonlte.f : set up calls to compute NLTE absorption coefficients
- klineshapes.f : computes qfncs, lte line strengths etc for NLTE absorption coefficients; optimized for 4 um CO2 region, and also reads in HITRAN parameters
- kreadVTprofs.f : reads/writes GENLN2 styles vib temp files
- kpredictVT.f : tries to predict NLTE profs and writes out GENLN2 style vib temp files
- kvoigt_cousin.f : computes voigt and cousin lineshapes
- kcousin.f : computes co2 continuum and cousin lineshapes
- n_main.f : main input file parsing routines
- n_gas_spectra.f : MOLGAS,XSCGAS,SPECTRA,NLTE parsing routines
- n_pth_mix.f : PTHFIL,MIXING parsing routines
- n_rad_jac_scat.f : RADFIL,JACOBN, SCATTER parsing routines
- n_output.f : OUTPUT parsing routines
- s_writefile.f : output writing routines
- s_misc.f : misc parsing routines and readers

9 Required data files

kCARTA requires a number of data files to be present at compile time, and another set to be present at run time. At compile time, the required files contain definitions for the array sizes, as well as the paths to the database and required files.

At run time, the various required files drive the run-time session of the program, tell it which compressed data files exist for which gas, in different wavenumber regions, and so on. Thus, some of these files will have to be regenerated by the user each time he/she updates the compressed database. In particular, the program compdatabase.f, in subdirectory UTILITY (along with the script comp.sc that exists in subdirectory SCRIPTS) should be run each time the user updates the files that exist in the compressed database. A summary of the existing database files then exists in comp97.param.

The following parameter files need to be present at compile time, in the same subdirectory as the source code.

Table 2: Compile time parameter files.

kcarta.param	compile time file that allows the user to define the paths to compressed database, reference profiles etc In addition, the user can define the size of matrices, arrays and so on here
gauss.param	contains Gauss-Legendre wieghts for angular integrations
scatter.param	contains interface array dimensions kCARTA to scatter code
pre_defined.param	contains many interface array dimensions for kCARTA; DO NOT TOUCH!!!!
post_defined.param	contains many interface array dimensions for kCARTA; DO NOT TOUCH!!!!

The following data files need to be present at run time, in the relevant subdirectory specified in kcarta.param. (If no cross-section gases are to be used, then xsec.param is not necessary)

Table 3: Run time data files.

comp97.param	contains a summary of the compressed database, so the program knows which gas absorption coefficients can be uncompressed for a particular frequency range

xsec.param	contains a summary of the crosssection database, so the program knows which CFC absorption coefficients can be included for a particular frequency range. Note that currently there is a switch between using the H92 cross section database, or using the H98 kCompressed files for these gases; this will be phased out eventually.

Of the following data files, only the driver namelist input file and the profile need to be present at run time, in the relevant subdirectory chosen by the user. Depending on the set of instructions found in the driver namelist file, the last two files may or may not be needed.

Table 4: Other data files.

driver namelist file	contains user specified Gas ID's, frequency range gas weights, atmosphere definitions, output styles, etc. This is the main file that drives the running of kCARTA
gas profile	contains the path profile information (gas amounts, pressures, partial pressures, temperatures and so on. kProfLayers layers are required for each gas. User can opt to use a shorter version (water and ozone amount/temperature profiles), or just use read in a regression profile. NOTE the gas profile can either be in the old KLAYERS format, which is a text file, or in the new RTP format
continuum	we are using our own set of lookup tables for the water continuum. These tables are based on the CKDv0,21,23,24 tables, as well as CKD 51,55 which are our own modifications
emissivity	if radiances are to be computed, the user can vary the surface emissivity as a function of wavenumber. NOTE If a RTP file is used to read in atmospheric info as well, this file will also include the emissivity and solar reflectivity

xsecdata	if the absorption coefficients of the *CFC's* are to be included, and the user wants to use the old H92 cross sections data, the file containing this info must be present. the user needs to set kXsecFormat = -1 in the kcarta.param file We supply file xsecdata.dat in DATA/General. Note that this requirement has been phased out, as the user can use a kCompressed style format for the cross section gases, by setting kXsecFormat = +1 in the kcarta.param file

A partial listing of the parameter file comp97.param is

Table 5: Parameter file listing.

1	605.000000	2830.000000	2
2	605.000000	1105.000000	2
2	1205.000000	1405.000000	2
2	1830.000000	2555.000000	2
2	2580.000000	2655.000000	2
2	2730.000000	2780.000000	2
3	605.000000	880.000000	2

where the first column denotes the (HITRAN) GasID, and the second/third columns the start/stop frequencies of the compressed database coverage of that gas. The fourth column indicates these files are of type "2" which all have a wavenumber spacing of 0.0025 cm^{-1}. The only gases/frequency combinations stored in the database are those that have appreciable cross sections for an Earth atmosphere.

When the program starts to run, it checks to ensure that the parameters set in kcarta.param, pre_defined.param, post_defined.param make sense. It then parses in the driver namelist file, and if necessary the gas profile. After this, the program starts running in earnest, making use of files comp97.param, xsec.param and xsec.data when necessary.

10 kCARTA BANDS

kCARTA was originally developed for modelling IR instruments such as AIRS, IASI and CRiS. This means the original compressed database was only required to span 605 - 2830 cm^{-1}. The paths kWaterPath,kCompPath,kCKDPath tell kCARTA where to look for the compressed files for this wavelength span. The files are identified by having an "r" prefix, and the point spacing is 0.0025 cm^{-1}.

kCARTA now has databases (UNVALIDATED!!!) that span either end of this original region. Currently (as of 01/29/2010) the span is from 80 cm^{-1} (FIR) to 25000 cm^{-1} (UV). The point spacing changes from band to band, going from 0.00025 cm^{-1} at the large wavelength (low freq) to 0.05 cm^{-1} at the small wavelength (high freq) end. The bands, with the prefixes, are as follows :

```
    v = 12000-25000 cm-1, at 0.05000 cm-1 spacing (= 500 cm-1 chunk span)
    o = 8250-12250 cm-1,  at 0.02500 cm-1 spacing (= 250 cm-1 chunk span)
    n = 5550-8350 cm-1,   at 0.01500 cm-1 spacing (= 150 cm-1 chunk span)
    m = 3550-5700 cm-1,   at 0.01000 cm-1 spacing (= 100 cm-1 chunk span)
    s = 2830-3580 cm-1,   at 0.00250 cm-1 spacing (= 025 cm-1 chunk span)
*** r =  605-2830 cm-1,   at 0.00250 cm-1 spacing (= 025 cm-1 chunk span)
    q =  500- 605 cm-1,   at 0.00150 cm-1 spacing (= 015 cm-1 chunk span)
    p =  300- 510 cm-1,   at 0.00100 cm-1 spacing (= 010 cm-1 chunk span)
    k =  140- 310 cm-1,   at 0.00050 cm-1 spacing (= 005 cm-1 chunk span)
    j =  080- 150 cm-1,   at 0.00025 cm-1 spacing (= 2.5 cm-1 chunk span)
```

This means the paths to the databases need to be specified at compile time. For example kWaterPathj,kCompPathj,kCKDPathj need to be set to tell the code where to find the FIR database for 80-150 cm^{-1}.

It is best to divide kCARTA into runtime chunks. So for example if the user wishes to get spectra between 605-3330 cm^{-1}, do two separate runs, one from 605-2830cm^{-1} and the other between 2830-3330 cm^{-1}. This is because the resolution of the spectra changes as you move from band to band.

The default radiative transfer in the NIR/VIS/UV is clear sky .. not even accounting for Rayleigh. So beware! Also, in the UV the xsec databases (from HITRAN) for O3 mysteriously ended at around 40800 cm^{-1}, and so ozone absorption from the tail end of the band is missing. Bad, bad, bad!!!

11 Water isotopes

The HDO isotope is highly variable spatially. This is seen most clearly in the SW band of IASI (2600-2800 cm^{-1}). kCARTA defines a region bounded by [kWaterIsoBandStart,kWaterIsoBandStop] [2405,3330] cm^{-1}. This encompasses the "r" band (2405-2830 cm^{-1}) and the "s" band (2830 - 3530 cm^{-1}) regions of the database. Water is seprated into gasIDs 1,103 where 1 is almost the usual HITRAN gasID for water, and 103 signifies heavy water (seprated or "broken out" on;y within the above mentioned region).

To model water, kCARTA uses
I) gasID = 1 = water with ALL isotopes (HITRAN nomenclature 1,2,3,4,5,6) OUTSIDE this band region and
II) gasID = 103 = water for HDO (HITRAN nomenclature = 4)
gasID = 1 = water with ALL OTHER isotopes (HITRAN nomenclature 1,2,3,4,6) INSIDE this region.

Outside this region, when "water" is uncompressed, it contains all the isotopes in compressed file b1_wxyz.dat. Within this region, it uncompresses HDO as gasID = 103 and H2O(isotopes 1,2,3,4,6) as gasID = 1

To complete the optical depth due to water, the self and foreign (gasIDs 101,102) continuums must be added in as well. The CKD model needs to be chosen for this.

12 Compile time file kcarta.param

Before compiling the code, the user has to tell kCARTA what size arrays and matrices are to be declared, as well as the paths to required files, such as those in the compressed database and the reference and regression profiles. This is done by setting various parameters in file kcarta.param, which is divided into 2 sections. The first section has the set of user defined parameters. How to set up this section of the file has to be thoroughly understood, as the parameters set in the kcarta.param file determine where the programs searches for the compressed database, as well as allocating memory. Any errors here will likely cause the program to stop execution. The second section of the file contains other definitions and parameters and should NOT BE TOUCHED (if they are, insert usual hazard messages, use at own risk, etc., here).

Table 6 gives the list of user definable parameters for this file. Recommended settings are given in the table; the user should not set his/her parameters indiscriminately. Note that the datafiles that are pointed to by the paths, need to be for the correct endian versions.

Table 6: Setting up compile time file kcarta.param

kOrigRefPath	this tells kCARTA where to look for the original 100 layers AIRS reference profiles. One should think of this as the definition of the kCARTA database; it is essentially the 1962 US Standard Profile, tweaked slightly to reflect eg increased CO2 and CH4 levels, and interpolated onto the AIRS 100 layers (101 pressure levels) grid.
	IEEE_BE vs IEEE_LE
kWaterPath	Path to the compressed data files for Water. Typically ../DATA/WaterDataBase
kCO2Path	Path to the compressed data files for linemix CO2. Typically ../DATA/CompDataBase
kCousin_CO2Path	Path to the compressed data files for cousin CO2. If available, typically ../DATA/CompDataBaseOld/ If unavailable, set to ../DATA/CompDataBase/ and ignore
kCompPath	Path to the compressed data files for RestOfGases (excl water,Co2). Typically ../DATA/CompDataBase
kCKDPath	Path to the CKD continuum data files (Water Self and Foreign). Typically ../DATA/General/CKD
kSolarPath	Path to the solar spectral data files. Typically ../DATA/General/SOLAR
kXsecFile	path to xsecdata.dat, which stores the cross-sectional cross-sectional absorption coefficients. Typically, in DATA/Template subdirectory

kCompParamFile kXsecParamFile	path to comp97.param, which stores gas ID/frequency span in the compressed database. Typically, in DATA/Template subdirectory path to xsec.param, which stores gas ID/frequency combinations in the cross-section databases. Typically, in DATA/Template subdirectory
kProfLayer kMaxAtm	this is number of layers that kCARTA will do computations for. This parameter should be set to be the same as the number of layers produced from *klayers.x* max number of atmospheres allowed. Does not use too much memory, so 5 is a safe number
kGasStore kMixFilRows kMaxPrint	max number of gases for which kCARTA allocates storage (from MOLGAS + XSCGAS $\leq$ kMaxGas). Typically, set to 38 (25 from MOLGAS, 13 from XSCGAS) max number of mixfil rows that can be read in. Remember that each set of mixedpaths has kProfLayers mixed paths. Thus a safe number = kProfLayers*(number of sets you will define) So a forward model using four sets of mixed paths (F,FW,FWO and FO), will require this number set to 4*kProfLayers max number of printing options that can be read in. Does not use too much memory, so 5(atmospheres)+2(one for gas spectra, one for path spectra)=7 is a safe number. Note that you should account for gas paths, mixed paths and radiances from EACH atmosphere as separate options

kEmsRegions	max number of data points in any emissivity data file. This means the number of regions defined = kEmsRegions-1. This can typically be set to 20
kMaxDQ	max number of gases to compute d/dq Jacobians for. Note that this uses up a LOT of memory. All gases specified in *nm_jacobn* have their d/dq computed (ALL gases are used in computing d/dT). Typically, set to 2 or 3.
kProfLayerJac	this is set to kProfLayer if the user wants to compute Jacobians while running the code, else it is set to 1
kMaxPtsJac	this is set to 10000 (number of wavenumber points per kCompressed chunk - see kMaxPts defined above) if the user wants to compute Jacobians while running the code, else it is set to 1
kMaxClouds	tells the code how many clouds can be used in radiance computations
kCloudLayers	tells the code how many layers each cloud can have
kXsecFormat	tells the code if minor cross section gases data are in separate kCompressed files format (+1) or binary file (-1). Will eventually be phased out as we move towards supplying only the kCompressed files for these gases
kWaterIsoBandStart	tells the code where heavy water band starts
kWaterIsoBandStop	tells the code where heavy water band stops
kWaterIsotopePath	tells the codes where to find heavy water (gasID=103) and water w/o isotope 4 (still refereed to as gasID = 1) exist

kWaterPathj	Path to the compressed data files for Water for 80-150 cm^{-1}.
kCompPathj	Path to the compressed data files for RestOfGases for 80-150 cm^{-1}
kCKDPathj	Path to the CKD continuum data files (Water Self and Foreign) for 80-150 cm^{-1}
kWaterPathk	Path to the compressed data files for Water for 140-310 cm^{-1}.
kCompPathk	Path to the compressed data files for RestOfGases for 140-310 cm^{-1}
kCKDPathk	Path to the CKD continuum data files (Water Self and Foreign) for 140-310 cm^{-1}
...	...
kWaterPathv	Path to the compressed data files for Water for 12000-25000 cm^{-1}.
kCompPathv	Path to the compressed data files for RestOfGases for 12000-25000 cm^{-1}
kCKDPathv	Path to the CKD continuum data files (Water Self and Foreign) for 12000-25000 cm^{-1}
kBoxCarUse	tells the code how many points to BLOAT the NLTE calcs by.
kRegrProf	tells the code how many regression training profiles used
kMaxPtsBox	tells the code how much space to save for NLTE optical depths
kBloatPts	seems to be the same as kBoxCarUse, but not really; too lazy to remember the difference, but I am sure there is one; probably if kBloatPts == 1, then this is SPACESAVER mode and so cannot do the bloated computations

caWeakCO2Path	for each of the *N* gases in NLTE, we now know how many chunks are in NLTE. The gas molecules have many lines/bands; some are weak and in LTE, some are strong and are in LTE, while others are strong and in NLTE. caWeakCO2Path is a path that points to a compressed databse, that has the WEAK lines in LTE precomputed.
kUpperAtmRefPath	path to upper atm 100 layers reference profiles. These profiles were used in making upper atm kCARTA database.
caWeakUpperAtmCO2Path	same as above, except it is the database for the upper atm (70-120 km). Also has 100 layers x 10000 pts
caLineMixDir	tells the code where the CO2 4 um LineMix files are, for use in the LBL nonLTE computations.
caStrongLineParams	tells code where the CO2 4 um line parameters are, for strong NLTE lines used in the LBL nonLTE computations. Must agree with the information in caaStrongLines (see below)
kNLTEProfLayer	tells the code the maximum number of layers in the NLTE profiles supplied by Dave Edwards GENLN2 (¡= 120 right now)
kRegrFile	path to regression profiles that code needs if it guesses VT. In the nm_nonlte section, user can give kCARTA a vib temp NLTE profile in variable caaNLTETemp; if the user does not do so and puts the name 'nlteguess', kCARTA will use a polynomial guess of the NLTE temps (it could also read in 48 regression profiles to see which one most closely matches the current profile, and then interpolate this profile in solar angle, but this is a little too complicated)

kLOPEZPUERTAS	if kCARTA tries to put in its own vib temp profile based on those from the regression profiles, it needs to know where to pick up the NLTE files computed by Manuel Lopez-Puertas
kNLTEPolyfits	if kCARTA tries to put in its own vib temp profile based on polynomial fits from the regression profiles, it needs to know where to pick up the coefficients for the bands and QVs

To keep memory sizes manageable, the jacobian parameter declarations in this file might be set to “1”. To allow kCARTA to perform jacobian computations, turn off the “space save dimensions” that might be in the tar package by simply uncommenting the lines above which these space saver dimensions are defined.

13 Compile time file scatter.param

Here the user can prune or enlarge the arrays used by the scattering codes *DISORT*, *RTSPEC*, *TWOSTREAM PCLSAM*. This will have major consequences on the amount of memory used when running kCARTA. However, it it probably safer to use the param file as it is. We might supply the tar package with the “space save dimensions” turned on; variables maxcly, maxulv,mammon etc might be set to 1; if so simply uncomment the lines above which these space saver dimensions are defined, to turn on the parameter value that *DISORT* can work with.

14 The Driver Namelist File

At present, our kCompressed database spans the wavenumber range from 605 cm^{-1} to 2805 cm^{-1}, with a point spacing of 0.0025 cm^{-1}. kCARTAv0.97+ has been written so that planned future extensions to the database can be easily handled by this current version. (The planned extensions are from about 250 cm^{-1} to 605 cm^{-1}, and will need a higher resolution than 0.0025 cm^{-1}). Note that the program has been written so that the user cannot “mix” regions that have different wavenumber point spacings. In other words, if the starting frequency is such that the corresponding file has a wavenumber spacing of 0.0001 cm^{-1}, and the stop frequency is such that the corresponding file has a wavenumber spacing of 0.0025 cm^{-1}, the program will stop.

This section gives the structure of the namelist file that controls the running of kCARTA. Note that

- **IMPORTANT** Each run of kCARTA involves one driver namelist file. Within this file, the user can either read in his own profile, or use one of the supplied regression files. Thus one kCARTA run can only use one profile.

- At present the absorption spectra for the minor cross section gases can be computed by either using a binary file containig all HITRAN92 data, or by reading in kCompressed files for these gases. This is controlled by parameter kXsecFormat above (set to -1 in kcarta.param distribution)

- While the code does allow one to compute the radiation measured by an upward looking instrument, at present even the bottommost layers are probably too coarse for an accurate estimate. Using our new kLAYERS code, the bottom AIRS layers can be subdivided finely, or the topmost layers can be clumped together. In addition, if the user wants the sun to be included in the FOV of the instrument, the program does not compute the variation in the local solar angle; at present, it simply uses the variation in the local path angle. We will be working on fixing these issues in the future.

- While reading this section, it might be helpful to look at the sample template files in ../DATA/TemplateNML subdirectory. In addition, the following points should be remembered :

  - the namelist file is divided into about 12 separate sections. For example, nm_prfile contains the profile name, while nm_frqncy contains the start and stop frequencies.
  - if at most the profile is driven by an RTP file (kRTP = -2,-1,0) then keywords MOLGAS, FRQNCY, PRFILE, OUTPUT are required
  - if the RTP file drives almost everything (kRTP = 1) then keywords MOLGAS, PRFILE, OUTPUT are required
  - keywords PARAMS, XSCFIL, WEIGHT, RADNCE, JACOBN, SCATTR, NONLTE, SPECTRA are optional
  - comment lines begin with !, character strings enclosed in quotes

- The namelist file is divided into separate sections, using the convention $*nm_section*$*end*. Within each section, the user will specify the variables required to be changed; any variables that are not changed will be initialized to dummy values that should not affect the running of kCARTA. The namelist sections should be defined in the order they appear below.
- Looking at the example files, you will notice that each of the separate namelist sections have a character string *namecomment* (of 33 characters) associated with them. This is not required, but it is good practice, as it clearly visually separates the sections from each other.

The following sections describe the formats and requirements for each of the *nm_keywords*.

14.1 *nm_params* (optional)

Here the user can reset values of parameters specific to a particular run. All lines beginning with "!" are considered comment lines and are ignored. If the value of a parameter is not set here, it defaults to the defined value below. These parameters affect the entire run of the program (including parsing in the driver namelist input file).

The following line is repeated as many times as necessary

caP = NewValue

where caP is an string that specifies which of the following 8 parameters is to be reset, and NewValue is the new setting of that parameter. The user should ensure that NewValue is real or integer as required.

Using the list of variable names below, the user can choose to reset the values corresponding to that particular variable. All of these parameters are considered one-time settings, and so will affect an entire kCARTA run.

Following is the table which gives a listing of the parameter name, allowed list of values for the parameter, and a description of what that setting of the parameter does. The boldface indicates the default setting of the parameter. If the user wants the program to use this default, he/she does not have to refer to the parameter in *nm_params*. Thus, if the user wants to use *all* the default settings, this section need not appear in the driver namelist file, or just be blank. All parameter values are integers. At present there is space for 12 parameters, but only 9 are used.

Table 7: Setting up parameters in *nm_params*

PARAM NUMBER	PARAM NAME	ALLOWED VALUES	DESCRIPTION
1	kLayer2Sp	-2,-1,1,2	(integer) Controls output for paths/ mixed paths, i.e., specifies whether to output transmittances or optical depths.
		-2	Output Layer transmittance $t(i) = exp(-k(i))$
		-1	**Output Layer Optical depth** $k(i)$
		1	Output Layer-to-Space Optical Depth $k2s(i) = \sum_{j=i}^{n}(k(j))$
		2	Output Layer-to-Space transmittance $t2s(i) = \exp(-\sum_{j=i}^{n} k(j))$
2	kCKD	-1,N	(integer) turn water continuum versions on/off. If turned on, specifies which of the following models to use
		0,21,23,24	Official CKD releases prior to January 2003
		1	MT_CKD releases (January 2003)
		2	our modified MT_CKD 1 (July 2003)
		51,55,60	derived by Machado, Strow, Hannon from RAL + AIRS data)
		12,13,50,52,56	will be gotten rid of eventually
		-1	no continuum
		0	use water continuum CKDv0
		21	use water continuum CKDv2.1
		23	use water continuum CKDv2.3
		24	use water continuum CKDv2.4
		60	use water continuum CKD-MSH v60
		1	**use water continuum MT_CKDv1**

		2	use water continuum MT_CKDv1, modified
3	kGasTemp	-1,1	(integer) how to set mixed path vertical temperatures for use in radiative transfer algorithm and Jacobian algorithm. The program can either use the CO2 temperature, or do a weighted average over all gases.
		-1	**do weighted gas average**
		1	use CO2 layer temperatures (if present)
4	kLongOrShort	-1,0,1	(integer) whether or not to output all driver namelist file information (including path profiles and weighting table lines) into output binary file header, or to output only essential information into the header
		1	**output complete header info to output binary file**
		-1	output summary header info to output binary file (does not repeat mixing table/gas profile info)
		0	output bare minimum header info, and bare data
5	kJacobOutput	-1,0,1,2	(integer) one of 3 output styles for Jacobians. They can be in raw dR/ds_m, $dR/ds_m\Delta s_m$ or $d(BT)/ds_m\Delta s_m$ where R is radiance, s_m is layer gas amount or temperature for layer m and BT is the brightness temperature. Δs_m is the amount of gas in layer m, or a 1 K change in temperature
		-1	output dR/dq,dR/dT
		0	output dR/dq * q, dR/dT
		1	**output d(BT)/dq * q, d(BT)/dT where q is the gas profile being perturbed.** Note d(BT)/d(ln(q)) = q d(BT)/dq

		2	output d(BT)/dq, d(BT)/dT Independent of the value of kJacobOutput, the second smaller Jacobian file always contains *radiances* computed at *surftemp*,*column gas* perturbation values of 1.0K and 0.1 respectively.
6	kFlux	-1,1,2,3,4,5	(integer) to calculate flux (+1,+2), outgoing radiation at all levels(+3), OLR at TOA (+4), OLR at TOA and ILR at GND(+5) or nothing (-1). There are 2 different output styles for the computed "fluxes." Note that the user has to read in the computed "flux" for each layer, and then sum over the computed fluxes, at the output resolution, to get the correct units. Also note that at present, only clearsky fluxes, or single cloud layer twostream fluxes, or multiple cloud layer pclsam fluxes, are possible
		-1	**no flux computations**
		1	flux at all levels : radiance * angle
		2	flux at all levels : Kelvin/day
		3	OLR at all levels : radiance * angle
		4	OLR at TOA : radiance * angle
		5	ILR at gnd; OLR at tropopause and TOA : radiance*angle
6a	kFlux	0,-1,1,2,3	(integer) to output Planck modifiers if NLTE computations are being done (+1), or nothing output (-1).
		-1	**no output planck modifiers**
		0	output planck modifers
		1,2,3	no output planck modifiers

7	kSurfTemp	-1,1	(integer) use this to either use the surface temperature found in *nm_radnce*, or to use the surface temperature found in *nm_radnce* as an offset to the temperature found by interpolating the surface pressure with respect to the AIRS pressure levels/profile layer temperatures. Note that if the atmosphere is being defined using the information in an RTP file (kRTP = 1), then the only possible setting for kSurfTemp is -1
		-1	**use surface temps in** *nm_radnce*
		+1	interpolate profile, and add on to surface temps in *nm_radnce*
8	kTempJac	-2,-1,0	(integer) how to compute temperature jacobians. This jacobian can be computed as a sum over d/dT(Planck $\times\tau$), or as d/dT(planck) or as d/dT(τ). Note that whether you use kTempJac = 0,-1,-2 the gas amount jacobians should not get messed up. This parameter only works for a downlooking instrument (ie has no effect on calculating the Jacobians for an uplook instrument)
		0	**use temp dependence in both Planck and** τ
		-1	use temp dependence only in τ. Note this should not mess up gas amount jacobians
		-2	use temp dependence only in Planck. Note this should not mess up gas amount jacobians

9	kRTP	-2,-1,0,1	(integer) whether or not to use an RTP file. Can have arbitrary number of layers per gas in each profile, as long as N ¡= kProfLayer (except fpr the kRPT=-1 case, where number of layers per gas must equal kProflayer
		+1	**RTP style profile, RTP atmosphere, scatter defn**
		0	**RTP style profile, user atmosphere, scatter defn**
		-1	**old style "klayers" profile, user atmosphere, scatter defn. This case NEEDS number of layers in profile == kProfLayer**
		-2	**GENLN4 style "layers" profile, user atmosphere, scatter defn**
10	kActualJacs	-1,20,30,40,50,100	(integer) how many jacs to actually do (dump zeros for the rest). Default (-1) is to compute all profile jacobians; others (20:50) dump out zeros for stuff not computed option 100 dumps column gas, stemp jacobians
		-1	**do ALL profile jacobians (gases(z),temp(z), wgt(z),surface)**
		20	only do Q(z) jacs
		30	only do T(z) jacs
		40	only do W(z) jacs
		50	only do S jacs
		100	only do column and stemp jacs
		100ABCXYZ	only do column and stemp jacs, with radiating atmosphere perturbed between layers ABC to XYZ read with "readkcbasic.m()

Parameter 3 (kGasTemp) is relevant **only** for a radiance (and jacobian) calculation. If set to +1, and CO2 is in the gas profile, the layers of the atmospheres built from the mixed paths, are set to the temperatures of the CO2 If CO2 is not present in the profile, or if the parameter is set to -1, the layer temperature that is used is a weighted average over the gases.

Options -1,20,30,40,50 for Parameter 10 (kActualJacs) produces (large) jacobian profile files, to be read in by "readkcjac.m". If the user wishes to only dump out column gas jacobians, and stemp jacobian, a much smaller file that can be read in using "readkcbasic.m" is produced when option 100 is used. The relationship between the column and profile jacobians is

$$\sum_{l=GND}^{TOA} J_l^q(n) = Jcol^q(n)$$

where q is the gas ID, l is the layer and n is the frequency index

If you use kActualJacs = 100; then it perturbs the entire atmosphere. If you use kActualJacs = 100ABCXYZ; then it perturbs the radiating atmosphere between ABC and XYZ wrt surface. So eg if the atmosphere was defined from (surface) 1013 mb to (TOA) 0 mb, the radiating layers correspond to KCARTA/KLAYER layers = 4:100. So if kActualJacs = 100001004, the first four of these radiating layers are used in the jacobian, which are 4,5,6,7. Code also does this for UPLOOK instrument : if the atmosphere was defined from (TOA) 0 mb to (surface) 1013 mb, the radiating layers correspond to KCARTA/KLAYER layers = 100:4. So if kActualJacs = 100001004, the first four of these radiating layers WRT surface are used in the jacobian, which again are 4,5,6,7 (instead of first four in the defn of the atmosphere, which would be 100,99,98,97). WARNING : 100ABCXYZ = column gas/temp from *radiating* layers wrt surface ABC to XYZ, and stemp. Note d/dT(i) in this case accounts for layer temp changes, and includes the OD changes due to temp changes; so it maybe a very good estimate, as it does radiative transfer with correct background thermal

14.2 *nm_frqncy* (mandatory)

This section specifies the frequency start/stop endpoints. If kRTP = -2,-1,0 the user has to speciy the start/stop wavenumbers that kCARTA has to process. The format here is

rf1 =
rf2 =

Note that the start/stop values could be altered by the program at run time, so that entire 10000 point chunks are output each time. For instance if the start,stop frequencies are specified by the user as 720.0 780.0, the program will reset these values to 705.0 and 779.9975 so that entire 10000 point chunks are output (705-729.9975, 730-754.9975, 755-779.9975).

If kRTP = +1, the RTP file must contain the channel minimum and maximum wavenumbers (head.vcmin and head.vcmax) that kCARTA will chunk through. kCARTA will automatically select start and stop endpoints that include the extremeties of these numbers.

kCARTA can currently compute spectra between 605.0 to 2830 cm-1, and so head.vcmin and head.vcmax need to be set within this interval. 605.0 = kaMinFr(2), 2830.0 = kaMaxFr(2) ... are the extremeties of our current database; in the future, when we go to variable wavenumber spacing, this will be changed.

All database files starting with “r” have a point spacing of 0.0025 cm^{-1}. These files currently span 605 to 2805 cm^{-1}.

In the future we will extend the frequency region that kCARTA can process. All database files starting with “q” will have a point spacing of 0.001 cm^{-1}. These files will probably span 205 to 605 cm^{-1}.
All database files starting with “s” will have a point spacing of 0.005 cm^{-1}. These files will probably span 2405 to 2805 cm^{-1}.

The start and stop frequencies specified in this section should be set with the above wavenumber restrictions in mind, as kCARTA will not allow a run that involves “mixing” of these database files. For example, if the start/stop frequencies are 605.0 and 2805.0 respectively, this means only files from the “r” database are required, and kCARTA will allow this run to proceed. However, if the start/stop frequencies are 405.0 and 805.0 respectively, this means that files from the “q” and “r” databases are required, and kCARTA will not allow this run to proceed.

14.3 *nm_molgas* (mandatory)

This section specifies molecular gas ID’s. The format is

iNgas =
iaGasesNL = Lgases(1), Lgases(2), ... , Lgases(iG)

or

iNgas = -1

iNgas specifies how many molecular gas ID's to read in. If iNgas = -1, then the program automatically includes all gases that exist in the compressed database. If iNgas > 0 then the program requires a list of iaGasesNL valid GasID's (between 1and 28)

Note there is a a subtle point here, the water continuum. If $kCKD \geq 0$ then the user wants water vapor plus continuum. Hence gasIDs 101 and 102 will also be included as two separate "gases, " if $iNgas = -1$. If the user specifies $kCKD \geq 0$ and lists the $molgas$es to be used, but neglects to use gasIDs 101 and 102, the program will halt. Similarly if the user specifies $kCKD < 0$ (no continuum) but includes gasIDs 101,102 in the list of $molgas$es to be used, the program will halt. However if the user specifies $kCKD < 0$ (no continuum) and uses $iNgas = -1$, the program will ingest this, knowing that gasIDs 101,102 are NOT to be used.

14.4 nm_xscgas (optional)

This section specifies cross-section gas ID's. The format is similar to above,

iNxsec =
iaNxsecNL = Lxsec(1), Lxsec(2), ..., Lxsec(iX)

or

iNxsec = -1

or

iNxsec = 0

where iNXsec specifies how many cross-section gasID's to read in. If iNXsec = -1, then the program automatically includes all gases that exist in the cross-section database. If iNXsec$>$ 0 then the program requires a list of iaNxsecNL valid GasID's (between 51 and 63). If iNXsec = 0 then no cross section gases will be used.

The total number of gases to be used in building up the atmosphere is

iN = iNumGases = iNgas + iNxsec.

14.5 nm_prfile (mandatory)

If kRTP = -2,-1, this means the user wishes to use an old style (text) GELN4/ kLAYERS file for the profile. So in addition the user simply has to specify the name of the $kLayers$ profile file. If kRTP = -1, the number of layers N for each gas MUST equal kPRofLayers; else we need $N \leq$ kProfLayer caPfname =

Inside the profile file, the following data is required:

- an integer iNpath that gives the total number of paths in the datafile. For each of the iNumGasesInProfile gases, there should be kProfLayer paths, giving a total of iNpath=iNumGasesInProfile*kProfLayer paths. This number should correspond to all the iNumGases*kProfLayer required paths corresponding to the gases found in MOLGAS and XSCGAS above. If it is less than the required number, the program will stop
- The rest of the strings contain the actual gas path information, repeated $kProflayer$ times for each of the iNumGasesInProfile gases. Note that the program only saves in memory the gas profiles for the gas ID's actually found in MOLGAS,XSCGAS. An example file is in the DATA/TemplateNML subdirectory. This is file testprof0, which contains a 100 layer water and a 100 layer ozone profile, for the US Standard Profile.

The datafile produced by $kLayers$ is therefore in the following format :

iNpath (where iNpath=iNumGases * kProfLayer)

```
idgas amt t dt p dp partp height      repeat × kProfLayer for gas 1
idgas amt t dt p dp partp height
...
idgas amt t dt p dp partp height      repeat × kProfLayer for gas 2
idgas amt t dt p dp partp height
...
idgas amt t dt p dp partp height      repeat × kProfLayer for gas iN
idgas amt t dt p dp partp height
```

idgas gives the Gas ID (by the end of the file, all gases found in GASFIL and XSCGAS must have been found in file caPfname). amt,t,p,partp are required during the uncompressions, and stand for gas integrated amount (in the GENLN2 units of kilomoles/square centimeter), temperature in Kelvin, layer pressure and gas partial pressure in atmospheres. height is the layer height, in kilometers (could be used during the radiative transfer, if user wants to account for the local radiation angle changing due to curvature of the earth)

The two other variables, dt and dp, at present are not needed, and can be set to 0.0.

If kRTP = 0,+1, this means the user wishes to use an new style netcdf RTP file for the profile. So in addition the user has to specify the name of the $kLayers$ profile file. The number of layers N for each gas must be less that or equal to kProfLayer : $N \leq$ kProfLayer

caPfname =

as well as state $which$ of the profiles should be used iRTP =

The RTP profile file will have the gasID, layer gas amounts, pressures and temperatures. The partial pressures (used only by water, for the continuum as well as for broadening effects) are computed by kCARTA

In this section, if the RTP file drives kCARTA to include an atmophere, computation, then one more variable must be set at this point

iMPSetForRadRTP =

This variable gives the MP set that is used to construct the atmosphere. For further information, the reader is referred to the namelist section nm_radnce (see below).

Additionally, in this section, if the RTP file drives kCARTA to include a cloud scattering computation, then at least three more variables must be set at this point. Case 1 allows for the main RTP file to set the cloud parameters for two cloud slabs, each between $ptop, pbot$ for iwp, dme combinations, and only requires the following three variables (in addition to the above info in the RTP file)

iNclouds_RTP =
caaCloudFile =
iBinORAsc=

The first (integer) variable tells kCARTA how many clouds to expect. The second (character*80) variable gives the name of the file that contains the cloud scattering information (such as asymmetry, single scattering albedo and extinction). The third (integer) parameter tells the code whether the files are in ascii format or binary format (-1/+1). These files need to be in the same format (producde by running Frank Evans' Mie scattering code) as that used to drive the scattering computations in the namelist section nm_scattr (see below), to which the user is referred to.

However, the user can also set arbitrary cloud profiles using KLAYERS. The cloud layer info must come in via a RTP file, which has gases 201,202,203 for water, ice, aerosol respectively. The name of this RTP file is

caCloudPFname=

and its cloud info (for profile $iRTP$ must correspond to that read in for the gas profiles (ie have same pressure levels, layers, number of layers etc). Integer variable $iaCloudFile$ associate the cloud scattering files with each cloud in the header (remeber, these are gases 201, 202, 203 in the .rtp file), while integer variables $iaCloudType$ associates these with water clouds(100-199), ice clouds (200-299), dust clouds(300-399). Real variable $raCloudDME$ associates the effective particle diameter with each cloud. An example is

```
$nm_prfile
namecomment =  '******* PRFILE section *******'
iBinOrAsc    = 1
iMPSetForRadRTP = 1
iRTP            = 1
caPFname = 'junk_pin_feb2002_sea_airsnadir_op.1cld.sun.rtp'

iNclouds_RTP  = 2
caCloudPFname = 'dne'

iNclouds_RTP  = 2
caCloudPFname = 'try_klayers_cloud.op.rtp'

iaCloudFile(1)   =  201
iaCloudFile(2)   =  202

iaCloudType(1)   =  100
iaCloudType(2)   =  200

raCloudDME(1)    =  10.0
raCloudDME(2)    =  30.0

caaCloudFile(1)  = 'MIEDATA/WATER250/water_405_2905_250'
caaCloudFile(2)  = 'MIEDATA/CIRRUS/cirrus_405_2905_220'

$end
```

If caCloudPFname is set to dne or omitted then the program assumes the cloud info is coming in slab format, as set in RTP file caPFname. Else it assumes a cloud profile info is coming in via RTP file caCloudPFname.

14.6 *nm_weight* (optional)

If this section is not included, then no radiance calculations will be attempted, even if *nm_radnce* exists. In this section, the following are required. Other than the first line, which contains an integer, the input information in this section is inscribed using strings (with opening and closing single quotes).

The first line contains the integer number of sets of kProfLayer mixed paths in the table, iNpmix (which gives a total of kProfLayer*iNpMix mixed paths defined).

The next few lines each contain, for each set, the weights assigned to the gases in one of three formats. Thus this section would look like

iNpmix
1 set of weights
2 set of weights
3 set of weights
...

where the set of weights is in one of the following possible formats. Note that each line you write out, should be preceded by 'caaMixFileLines(x) = ' , where "x" is a counter that specifies which line you have written out. Also, the set of weights needs to be enclosed in single quotes.

caaMixFileLines(x) =
'iN list of weights'

or

caaMixFileLines(x) =
'iN -1 rW -1'

or

caaMixFileLines(x) =
'iN -1 rW iG'
caaMixFileLines(x+1) =
'i1 r1 i2 r2 ... iG rG'

Each set of weights defines kProfLayer mixed paths. Thus if the user specifies that there are 5 sets of weights in the *nm_weight* section (iNpmixtemp=5), then 500 mixed paths are defined by kCARTA. We now explain the three formats, giving examples.

14.6.1 format iN list of weights

For mixed path set iN, the weights for the gases are specified by the list of weights (one for each of the gases in MOLGAS/XSCGAS). Thus for example, if there are a total of 5 gases (from MOLGAS/XSCGAS), then

'1 1.0 1.1 1.01 1.0 1.0'

is valid. The leftmost 1 identifies this as the first set of weights in the section. The next five numbers are the weights of the 5 gases, and says that the first, fourth and fifth gases have a weight of 1.0, while gases 2,3 have weights of 1.1,1.01 respectively.

14.6.2 format iN -1 rW -1

For mixed path set iN, the weights for ALL the gases are specified by rW Thus again, if there are five gases, then

'2 -1 1.0 -1'

says that all five gases are weighted by 1.0 (the leftmost number 2 indicates this is the second set of weights in our hypothetical nm_weight section)

14.6.3 format iN -1 rW iG

This format specifies that for mixed path set iN, all the gases have a weight of rW, except for iG of the gases. These iG gases then have their weights specified by the additional lines(s), where the gas ID is followed by the weight of that gas.

'i1 r1 i2 r2 ... iG rG'

Then, for example, a set of gas weights that are explicitly specified in subsection 14.6.1 above, 1 1.0 1.1 1.01 1.0 1.0, could be specified equally by the following two lines

'1 -1 1.0 2'
'2 1.1 3 1.01'

On the first line, the leftmost 1 specifies this is the first set of mixed paths being defined. All gases have a weight of 1.0, except for 2 gases. The next line then lists the gasIDs and the weights associated with these 2 exceptions.

14.6.4 Detailed example

Assuming 6 gases have been found in MOLGAS/XSCGAS, the following WEIGHT table defined in a file, then

```
        3
caaMixFileLines(1) =
        '1 -1 1.0 -1'
caaMixFileLines(2) =
        '2 -1 1.0 2'
caaMixFileLines(3) =
            '101 1.8 102 1.1'
caaMixFileLines(4) =
        '3 1.1 1.2 1.3 1.4 1.5 1.6'
```

defines 3*kProfLayer mixed paths.

- The first kProfLayer have all 6 gases equally weighted by 1.0, in all layers (ie the weight table is 1.0 1.0 1.0 1.0 1.0 1.0)
- The next kProfLayer have all 6 gases equally weighted by 1.0, in all layers. The exception are 2 gases—gasID 101 (water self) has weight 1.8 in all layers, gasID 102 (Water Foreign) has a weight of 1.1 in all layers (ie the weight table is 1.0 1.0 1.0 1.0 1.8 1.1)
- The last kProfLayer have weight 1.1 for the first gas (all layers),weight 1.2 for the second gas (all layers), ..., weight 1.6 for the sixth gas (all layers) (i.e., the weight table is 1.1 1.2 1.3 1.4 1.5 1.6)

14.7 nm_radnce (optional)

If this section is ignored, no radiance calculation will be done (only absorption coeffs or mixed path spectra can be output). If this section is present, without nm_weight, no radiance calculations can be performed. Also note : if this section is present, without any radiance from a chosen atmosphere being specified in the nm_output section, the program will not bother to do a radiative transfer for that defined atmosphere. This also means that no Jacobian or flux calculations will be done for that atmosphere. Satellite angles are with respect to the vertical.

If $kRTP = -2, -1, 0$ this means the user will be defining the atmosphere, using the specifications stated below. If $kRTP = +1$ then kCARTA will use the atmosphere definitions specified in the RTP file that contains the profile; the documentation can be found in KLAYERS/Doc. Only ONE atmosphere can be read in from the RTP file.

If kRTP = -1, the number of layers N for each gas MUST equal kPRofLayers; else we need $N \leq$ kProfLayer
Note that if a radiance computation is needed, and the lowest pressure (highest altitude) level in the RTP file is greater than 10.0 mb, the code will halt, as there will be inaccurate radiance computations performed.

14.7.1 kRTP = -2,-1,0

We now assume that $kRTP = -2, -1, 0$ and so the user needs to define the atmosphere(s) in the namelist file, as follows. The first line has an integer iNatm that specifies how many atmospheres are going to be defined in this section. For each of those atmospheres, the user has to specify the start and stop pressures, satellite view angle, information specifying whether solar and background thermal are on or off, and the emissivity. Thus this section would look like

iNatm

iaMPSetForRad(1) =
raPressStart(1) =
raPressStop(1) =
raTspace(1) =
raTsurf(1) =
raSatAngle(1) =
raSatHeight(1) =
iakSolar(1) =
rakSolarAngle(1) =
cakSolarRefl(1) =
rakSolarRefl(1) =
iakThermal(1) =
rakThermalAngle(1) =
rakThermalJacob(1) =
caEmissivity(1) =
raSetEmissivity(1) =

iaMPSetForRad(2) =
raPressStart(2) =
raPressStop(2) =
raTspace(2) =

raTsurf(2) =
raSatAngle(2) =
raSatHeight(2) =
iakSolar(2) =
rakSolarAngle(2) =
cakSolarRefl(2) =
rakSolarRefl(2) =
iakThermal(2) =
rakThermalAngle(2) =
rakThermalJacob(2) =
caEmissivity(2) =
raSetEmissivity(2) =

...

OR iNatm
iaMPSetForRad = i1,i2, ...
raPressStart = r1,r2, ...
raPressStop = r1,r2, ...
raTspace = r1,r2, ...
raTsurf = r1,r2, ...
raSatAngle = r1,r2, ...
raSatHeight = r1,r2, ...
iakSolar = i1,i2, ...
rakSolarAngle = r1,r2, ...
cakSolarRefl = ca1,ca2, ...
rakSolarRefl = r1,r2, ...
iakThermal = i1,i2, ...
rakThermalAngle = r1,r2, ...
rakThermalJacob = r1,r2, ...
caEmissivity = ca1,ca2, ...
raSetEmissivity = r1,r2, ...

Note that if the instrument is upward looking, the last 8 parameters ($iakSolar$,$rakSolarAngle$,...,$raSetEmissivity$) are read in and ignored.

iaMPSetForRad(iI) is an integer specifying the program which set of mixed paths to use (1 corresponds to the first set of mixed paths, numbered 1 to kProfLayer, 2 corresponds to the second set of mixed paths, numbered kProfLayer+1 to 2*kProfLayer etc. Refer to the section on *nm_weight* above).

The program internally defines the atmospheres sequentially from 1 to iNatm, so that the user can easily specify the output he/she desires for any of the atmospheres. Thus the user should remember that iaMPSetForRad(iI)... refer to the *mixed path weight sets* defined in *nm_weight* above, in any order, and *NOT* to the atmosphere number.

raPressStart(iI),raPressStop(iI) are real numbers (units of millibars) that define the direction of radiation travel thru the atmosphere by specifying the pressures that constitute the upper/ lower boundaries.

- if the pressures are less than 0.005 mb, they are reset to 0.005 mb (top of 100th AIRS layer 100km)
- if the pressures are greater than 1100.0 mb, they are reset to 1100.0 mb (bottom of 1st AIRS layer)
- if rStartP is greater than rStopP then the radiation is travelling from high pressure (low in the atmosphere) to low pressure (higher up in the atmosphere), and so the instrument is downward looking. In this case, the user can further fine tune the radiative transfer model, by specifying surface temperatures, emissivities and so on (see below)
- if rStartP is less than rStopP then the radiation is travelling from low pressure (high in the atmosphere) to high pressure (lower down in the atmosphere), and so the instrument is upward looking. Parameters such as surface temperature and emissivities are now meaningless (but they must be set to dummy values for the program to read in.
- the start and stop pressures are flexible enough to allow the user to be able to define fractional top and bottom layers. Please refer to Section [?] below

raTSpace(iI) is a real number, stating the blackbody temperature of the background space($\simeq$ 2.6K).

raTSurf(iI) is a real number, stating the temperature of the earth's surface. Note for an upward looking instrument, this value is read in and ignored. Also note that depending on the value of *kSurfTemp* set in *nm_params*, this value should

be set with care!!!
If $kSurfTemp$ is less than 0.0, then for a downlooking instrument, the program will read in raTSurf(il) and use this for the actual surface temperature.
If $kSurfTemp$ is greater than 0.0, then for a downlooking instrument, the program will read in the surface temperatures raTSurf(il) in nm_radnce, and add these values to a pressure interpolated boundary temperature it computes, for the actual surface temperature.

raSatAngle(il) is a real number that specifies the satellite view angle (wrt vertical) in degrees. For a downward looking instrument, this is the angle between the satellite looking down to the center of the earth, and the satellite looking away at a point on the surface of the earth. For an upward looking instrument, this is the angle between where the satellite is looking at and the local vertical.

raSatHeight(i) is a real number that specifies the satellite height (in km). This accounts for the slight change of angle wrt vertical, as radiation travels between the earth's surface and satellite, due to the intrinsic curvature of the earth's surface.

- If this value is set to a negative amount, then the angles used in all layers are the same (specified by Sat Angle above).
- if this value is set to a positive amount, then the angles used in all layers are a modification to Sat Angle.
- if the satellite is upward looking, and if parameter Sat Height is positive, the code automatically sets the height of the instrument to 705 km. (In other words it tries to reverse trace the path that a ray from a downward looking instrument at 705 km would take). In addition, at present, if the sun fills the FOV, the local sun angle that is used is simply the local mixed path angle. As mentioned elsewhere, since the present layering is probably too coarse for an upward looking instrument, more work will be done on the upward looking algorithm in the future.

iakSolar(il) is an integer **(limited to -1, 0 or 1)** that tells the program whether or not to include solar radiation for up or downlooking instruments

- (+1) : include solar radiation, using actual solar spectral data
- (0) : include solar radiation, using T=5800K

- (-1) : do not include solar radiation
- Obviously, if the sun angle and the satellite view angle are different for an uplook instrument, then the sun being on will have no effect if there is no cloud scattering.

rakSolarAngle(iI) is a real number specifying the sun angle, in degrees. It has to be between 0.0 and +90.0, or iaKSolar will be reset to -1

cakSolarRefl(iI) is a character allowing the user to assign a file to read sun reflectance values from. If this is not specified, then

- if raKSolarRefl $\leq$ 0, $(1-\epsilon)/\pi$ is used across the spectrum.
- if raKSolarRefl $\geq$ 0, this value is used across the spectrum.

- If DISORT scattering code is used, then the solar reflectance cannot be set by the user ie it is always $(1-\epsilon)/\pi$
- If the filename specified is DISORT scattering code is used, then the solar reflectance can be set by the user, and so can violate $refl = (1-\epsilon)/\pi$

iakThermal(iI) is an integer (**limited to -1, 0 or 1**) that tells the program whether or not to include the effects of reflected background thermal radiation.

- If set to -1, no background thermal effects are included
- If set to 0, background thermal effects included are computed using our fast diffusive approximation at lower layers, where it matters, while the upper layers use an $acos(3/5)$ diffusive approximation. Also, parameter rThermalAngle is meaningful.
- If set to $+1$, background thermal effects included are computed using a Gaussian quadrature over the zenith angles (slow)
- If DISORT scattering code is used, then background thermal is always ON

rakThermalAngle(iI) is a real number specifying the diffusivity angle, in degrees (whose magnitude must be less than 90.0). This parameter has meaning only if iThermal = 0, else the background thermal is either not computed or computed accurately using Gaussian quadrature.

- If the user specifies a positive value between 0.0 and +90.0, this angle will be used at all layers.

- If the user specifies a negative value, a value of acos(3/5) will be used at the upper layers, while our optimum diffusive angle will be used at lower layers, the combination of which is a better approximation than using a single diffusive angle at *all* layers.
- If JACOBN is set, then this could lead to some discrepancies. The clear sky jacobian assumes all angles are $acos(3/5)$, while the radiative transfer will use whichever angles it needs, as set by rakThermalAngle
- If DISORT scattering code is used, then this angle is irrelevant

Thus if iakThermal(iI) = 0, then rakThermalAngle(iI) should be used with care. If it is set at a negative value x, then for the upper layers the diffusive angle $acos(3/5)$ is used for the reflected thermal, while for the lower layers, a parameterized optimum diffusivity angle is used. If it is set at a positive value, then for all layers, the diffusive angle $acos(x)$ is used for the reflected thermal. acos(3/5) = 53.1301 degrees

Table 8: Diffusivity Options.

kThermalAngle	-x	use parameterized optimum diffusivity angle at lower layers, acos(3/5) at higher layers
	+x	user specified value (in degrees) at ALL layers

iakThermalJacob(iI) is an integer (**limited to -1 or 1**) that tells the program whether or not to include the effects of background thermal

- (-1) : do not include background effects in radiance Jacobian
- (+1) : include background effects in radiance Jacobian
- if effects of background thermal are to be included in the radiative transfer algorithm, but not in the Jacobian, there could be large errors (as much as 30 %) in the Jacobians of the lower levels.
- to speed up the Jacobian code, an angle of $acos(3/5)$ is used for all layers, instead of the optimum diffusivity angle being used at the lower layers. Thus the setting of parameter kThermal is ignored here.

Either the emissivity data file caEmissivity(iI) is a text file containing the (wavenumber dependent) surface emissivity, or emissivity value raSetEmissivity(iI) is a real number for the surface emissivity (constant at all wavenumbers). If a emissivity file is to be read in then raSetEmissivity(iI) $MUST$ be set to a negative value; if it is set to a positive value, this is what will be used across the entire wavenumber range. For an upward looking instrument, the surface emissivity file/values are read in and ignored.

To summarize,

for a downward looking instrument (rStartP $>$ rStopP)

iaMPSetForRad tells which set of mixed paths to use
raPressStart is the pressure of the starting layer
rPressStop is the pressure of the stop layer
raTSpace is the space blackbody temperature (2.6k)
raTsurf is the surface temperature
raSatAngle is the satellite angle (in degrees)
raSatHeight (in km) indicates whether to modify SatAngle
iakSolar (-1, 0 or 1) sets solar radiation on/off
rakSolarAngle sets the solar zenith angle
cakSolarRefl is a file for the solar reflectance
rakSolarRefl sets the solar reflectance (-1 means use (1-e)/pi or cakSolarRefl)
iakThermal (-1, 0 or 1) sets the reflected thermal on/off
rakThermalAngle sets the diffusivity angle
iakThermalJacob sets thermal background inclusion in Jacobian on/off
caEmissivity is a file containing the emissivity parameters OR
raSetEmissivity specify a constant emissivity value

For an upward looking instrument (rStartP $<$ rStopP)

14.7.2 The Emissivity and Solar Reflectance Data Files

This is a text file containing the (wavenumber dependent) surface emissivities or solar reflectances are in the following format

iE
rS(1) emiss(1)
rS(2) emiss(2)
... ...
rS(iE) emiss(iE)

where iE is an integer that tells how many emissivity/reflectance points are going to be set (minimum of 2). The frequencies should be input in ascending order. Each of the iE sets of data contain

start-freq emissivity

Thus the total number of regions defined by the file is iE-1. A linear interpolation of emissivities is done between the *i,i+1*th data points set in the file, rS(i) and rS(i+1). For frequencies less than the minimum frequency defined in the file, the emissivity is set to the corresponding emissivity value; for frequencies more than the maximum frequency defined in the file, the emissivity is set to the corresponding emissivity value.

For example, if the start/stop frequencies from nm_frqncy were 705.0 755.0 and the following was found in the emissivity data file

2	
730.0	0.9
750.0	0.95

then the following emissivities are used

705.0-730.0	0.9
730.0-750.0	0.9+(0.05)*(freq-730.0)/(750.0-730.0)
750.0-755.0	0.95

14.7.3 kRTP = +1

As mentioned above, the documentation for this will be found in the KLAYERS/Src/Doc directory. If $kRTP = 1$ then only ONE radiance can be computed for ONE single atmosphere. Parameter $kSurfTemp$ has to be set to -1, as the RTP file specifies the surface temperature. From the $WEIGHT$ section, one still needs to set IAMPSetForRad, to tell kCARTA which set of weights to use for the atmosphere. The rest of the variables needed to define the radiating/emitting/absorbing atmosphere are set as follows.

```
The atmosphere start/stop pressures depend on prof.spres,prof.pobs and
prof.upwell!! In addition, the observing pressure needs to be set as well.
see more explanations below !!!!
               a) rPressStop  = 0.0 (TOA)
                   (can be reset depending on pobs and upwell)
               b) rPressStart = prof.spres
                   (can be reset depending on pobs and upwell)

The surface temperature and satellite view angle are straightforward
               c) rTBdy       = 2.96
               d) rTSurf      = prof.stemp
               e) rAngle      = abs(prof.satang)
```

```
              f) rHeight     = prof.zobs (in meters!!!! if prof.zobs > 0)
                             = -1 o/w

Background thermal is alsways turned on (iaKThermal(iC) = 0)
At least one data point is required in prof.efreq, prof.emis (corresponding
to a constant emissivity); two or more points correspond to a spectrally
varying emissivity
              g) iaKThermal(iC)   = 0
                 raKThermalAngle(iC) = -1.0
                 iakThermalJacob(iC) = 1
              h) iaSetEms(iC) = prof.nemis
                 DO i=1,iaSetEms(iC)
                   r1   = prof.efreq(i)
                   rEms = prof.emis(i)
                   END DO

The sun is turned ON or OFF depending on the value of prof.solzen
At least one data point is required in prof.rfreq, prof.rho (corresponding
to a constant emissivity); two or more points correspond to a spectrally
varying emissivity. O/W a sun reflectivity of (1-emiss)/pi is used

              i) rakSolarAngle(iC) = prof.sunang
                 IF (prof.sunang >=  0.0) & (prof.sunang <= 90.0)) THEN
                   iakSolar(iC) = +1
                 ELSE
                   iakSolar(iC) = -1
                  END IF

              j) caKSolarRefl(iC) = 'Dummy' which gives
                 iaSetSolarRefl(iC) = prof.nrho
                 DO i=1,iaSetSolarRefl(iC)
                   r1   = prof.rfreq(i)
                   rRefl = prof.rho(i)
                   END DO
         OR
              j) caKSolarRefl(iC) = 'NONESPECIFIED' which gives
                 iaSetSolarRefl(iC) = prof.nemis
```

```
DO i=1,iaSetSolarRefl(iC)
  r1   = prof.efreq(i)
  rRefl = (1-prof.emis(i))/pi
  END DO
```

As seen above, more explanation of the setting of the atmosphere start/stop pressures is needed. Recall the RTP file has the lowest and highest profile pressures (corresponding to highest and lowest altitudes), $pLow$ and $pHigh$

- A first cut is to assume a downlooking instrument at TOA (upwelling radiation), and so the code sets upwell = +1, rPressStop = 0.0, rPressStart = prof.spres, observation pressure pobs = 0.0
- Then the code checks the value of prof.pobs :
  If $prof.obs \geq 0$, then the observation pressure is checked to ensure that it lies between $plow \leq pobs \leq phigh$
  If $prof.obs \leq 0$, then the observation pressure is still 0.0
- Then the code checks the value of prof.upwell :
  If $prof.upwell = 1$, then $upwell = prof.upwell$ (downlook instr)
  If $prof.upwell = 2$, then $upwell = prof.upwell$ (uplook instr)
  If $prof.upwellO/W$, then $upwell = 1$ (downlook instr)
- Now that the code has set the radiation direction ($upwell$) and pbservation pressure ($pobs$), it can set the start/stop pressures correctly
  If $upwell = 1$, it sets $rPressStart = prof.spres, rPressStop = pobs$ and the code requires $rPressStart \geq rPressStop$, or it halts
  If $upwell = 2$, it sets $rPressStart = pobs, rPressStop = prof.spres$ and the code requires $rPressStart \leq rPressStop$, or it halts

14.7.4 Fractional Layers

Depending on the user requirements, such as surface pressure or position of a downward looking instrument, the start and stop pressures can define layers that are fractional. Thus for example, we could have the following namelist initializations for this section :

```
iaMPSetForRad(1) = 1
raPressStart(1) = 1000.0
raPressStop(1) = 40.0
raTspace(1) = 2.7
raTsurf(1) = 303.34
raSatAngle(1) = 0.0
raSatHeight(1) = -1.0
iakSolar(1) = -1
rakSolarAngle(1) = -1.0
cakSolarRefl(1) = 'NONEDEFINED'
rakSolarRefl(1) = -1.0
iakThermal(1) = 0
rakThermalAngle(1) = -1.0
rakThermalJacob(1) = 1
caEmissivity(1) = 'emiss.dat'
raSetEmissivity(1) = -1

iaMPSetForRad(2) = 2
raPressStart(2) = 0.0
raPressStop(2) = 1000.0
raTspace(2) = 2.7.0
raTsurf(2) = 303.34
raSatAngle(2) = 10.0
raSatHeight(2) = 705.0
iakSolar(2) = 1
rakSolarAngle(2) = 0.0
cakSolarRefl(2) = 'dummysun'
rakSolarRefl(1) = -1.0
iakThermal(2) = -1
rakThermalAngle(2) = 35.0
rakThermalJacob(2) = 1
caEmissivity(2) = 'dummy'
raSetEmissivity(2) = 0.85
```

Assume we are using the 100 original AIRS layers in this discussion. The above defines instruments in two atmospheres. The first atmosphere has a downward looking instrument, with the radiating atmosphere defined between pressures 1000.0 and 40.0, with the first kProfLayer mixed paths being used. This corresponds to roughly layers 4 to 68. However, only the top one third of layer 4 is included (due to the position of the surface). Similarly layer 68 has a fractional weight of 0.2 (only the bottom 2 tenths of the layer is included, due to the position of the instrument flying within the layer). Note that if the program is asked to include the solar and background thermal contributions, then it will use layers kProfLayer downto 4 in the computation of solar/thermal incident at the surface, and then revert back to atmosphere from layers 4 to 68.

From the second line, solar radiation is turned off. However, the fast diffusive approximation is turned on, and if Jacobians are to be computed, the contribution of background thermal is included. The last -1.000 indicates that the effects of curvature of the earth on the local satellite viewing angle, are not to be included (in any case, the satellite view angle is 0.0). From the third line, file ../OUTPUT/emissivity.dat contains the the data for the emissivity.

The second atmosphere has a upward looking instrument, with the radiating atmosphere defined between layers 200 and 110. (the integer "2" indicates that the mixed paths 101-200 should be used—the pressures correspond to the 10th and 100 th in this set). However, layer 110 has a fractional weight of 0.6 (due to the position of the surface). As the start pressure was input as 0.0 mb, it is reset to 0.005 mb (top of 100th layer). The highest layer (200) thus has a fractional weight of 1.0 as we are looking all the way into space. The sun is filling the FOV. The instrument is pointed in a direction of 10 degrees with respect to the vertical, and the program has been asked to include the effects of curvature on the viewing angle. The next two lines contain settings that are read in and ignored (the emissivity is set to 0.85 across the entire frequency range, even though it is ignored in the forward model calculations).

14.8 *nm_output* (mandatory)

This section requires the user to specify the output filename. Note the program will not overwrite an existing file. Additionally, this section requires the user to specify which paths/mixed paths/radiances are to be output (for ALL wavenumbers). Note that if the user specifies the same output option more than once, the program will try to merge the information together.

The information required in this section is of the form output file name followed by output specification part, as follows

caComment =
caLogFile =

```
    iaPrinter(1) =
        iaGPMPAtm(1) =
        iaNp(1) =
        iaaOp(1,:) =
        raaOp(1,:) =

    iaPrinter(2) =
        iaGPMPAtm(2) =
        iaNp(2) =
        iaaOp(2,:) =
        raaOp(2,:) =
    ...
```

OR

```
    iaPrinter = i1,i2, ...
        iaGPMPAtm = i1,i2, ...
        iaNp = i1,i2, ...
        iaaOp =
        raaOp =
    ...
```

caComment is a 80 character string that the user can use to enter a short one line summary of the kCARTA run eg 'Cirrus cloud run' or 'Gas breakouts for Fast Models' or 'Uplook instrument' etc. If this variable is not set, it is defaulted to 'KCARTA run'

caLogFile is a 80 character string that the user can use to specify the name of the log file that kCARTA produces. This log file details any warnings, and pretty much tells the user hoiw kCARTA interpreted the directives of the namelist file. If this variable is not set, it is defaulted to 'warning.msg'
One of three possibilities (iaPrinter) can be output by the program :

Table 9: Output options.

1	specify which gas path spectra to output
2	specify which mixed path spectra to output
3	specify atmospheres/pressures to output radiances at

If iaPrinter(iI) is not one of 1,2 or 3, the program aborts

Because of the structure of the algorithm, the output options have to be listed in order ie Gas paths followed by Mixed paths followed by radiances. If the user specifies eg radiances to be output, followed by path spectra, the code will halt immediately after parsing in the driver namelist file. WARNING : At present, both $DISORT$ and $RTSPEC$ can only output radiances at pressure level boundaries. If arbitrary pressures are specified, they are rounded up or down to the nearest boundary.
Depending on what output option was chosen (iaPrinter(iI) = 1,2 or 3), the user then has to to specify

iaGPMPAtm and iaNp

where iaGPMPAtm(iI) = gas id/mixedpath set/atmosphere number
and iaNp(iI) = number of paths/mixed paths/radiances to output

As kCARTA outputs the lists of paths and MPs to be output, it checks to make sure that the same path/MP layer does not occur more than once. Similarly, it checks the number of atmospheres read in from RADNCE against the highest atmosphere number read in from OUTPUT to make sure that information from an an invalid atmosphere will not be expected to be output. It also makes sure that the radiances output per atmosphere is less than or equal to *minimum(kProfLayer,actual number of radiating layers)* output per atmosphere. If these criteria are violated, the program informs the user and halts.

Note that when the user specifies the pressures at which radiances are to be output, he/she must ensure the pressures lie within those that define the atmosphere, else the program will halt. Again, as in the nm_radnce section, if the user inputs pressures greater than 1100.0 mb or less than 0.005 mb, they are reset to 1100.0 and 0.005 mb respectively. In addition, depending on the specifics of the atmosphere, the highest/lowest pressures at which radiances are output are set according to those specified by the boundaries.

The next three subsection explains the requirements for a path, mixed path and radiance outputs.

14.8.1 Output Gas Paths (Option 1 : iaPrinter(iI)= 1)

For this iaPrinter(iI) (= 1), the user needs to specify iG (gas IDs) and list of gas paths to output. The program expects one of the following four combinations of iaGPMPAtm(iI) iaNp(iI) :

Table 10: Output options for iaPrinter(iI)=1

iaGPMPAtm	iaNP	DESCRIPTION
-1	-1	all paths/mixed paths output
iG	-1	for gasID=iG output all kProfLayers layers
-1	iN	for all gases iG=1,iNumgases, output spectra in the iN layers specified in list on next line
iG	iN	for gas iG, output spectra in the iN layers specified in list on the next line

If iaNP(iI) is not equal to -1, then the next line should contain a list of the iaNP paths to be output

Note that iPrinter = 1 can be repeated more than once—the instructions found each time it finds iPrinter = 1 are simply merged together with the previous instructions. Thus for instance

iaPrinter(1) = 1
iaGPMPAtm(1) = 1
iaNp(1) = -1
iaPrinter(2) = 1
iaGPMPAtm(2) = 101
iaNp(2) = -1
iaPrinter(3) = 1
iaGPMPAtm(3) = 102
iaNp(3) = -1
iaPrinter(4) = 1
iaGPMPAtm(4) = 2
iaNp(4) = -1

are perfectly valid, as the program will output the spectra for all kProfLayers layers for Gas ID's 1,101, 102 (water basement, water self, water foreign) and 2 (CO2)

WARNING : kCARTA has been structured so that the total water optical depth is broken into that of three components, according to the CKD definitions : gasID = 1 being the without basement lorentz term, gasID = 101 being the self continuum and gasID = 102 being the foreign continuum. An unsuspecting user might only turn gasID = 1 on, and be puzzled why the water optical depth seen in the output files seems small (eg in the 10 um window region, the main contribution is due to the self continuum). Thus the user should ask for gases 1,101 and 102 to be output, as has been done in the example above!!! See Figure 5, which illustrates this for the tropical (wet) profile. One can see that in between the lines (eg at 894 cm, the water without basement (blue) and water foreign (red) have a negligible contribution; the water self (green) contributes the most to the overall optical depth (cyan). Conversely, Figure ??watersaw) for the sub artic winter (dry) profile will demonstrate that only about 1/2 to 2/3 of the optical depth in the same region is due to water(self); other gases will have an appreciable contribution to the total optical depth

Example : Suppose 3 gases have been found in nm_molgas or nm_xscgas and 400 mixed paths defined using nm_weight and 2 atmospheres defined in nm_radnce. The following are examples of what output paths the user can specify

```
iaPrinter(1) = 1
iaGPMPAtm(1) = 1
iaNp(1) = -1
```

will list ALL spectra for ALL gases. For each 10000 point chunk, the kProfLayers layers of gas 1, kProfLayers layers of gas 2 and kProfLayers layers of gas 3 will be output (10000 pts per layer).

```
iaPrinter(1) = 1
iaGPMPAtm(1) = 1
iaNp(1) = 2
iaaOp(1,1)=10
iaaOp(1,2)=20

iaPrinter(2) = 1
iaGPMPAtm(2) = 2
iaNp(1) = 1
iaaOp(2,1)=30

iaPrinter(3) = 1
iaGPMPAtm(3) = 3
iaNp(3) = 2
```

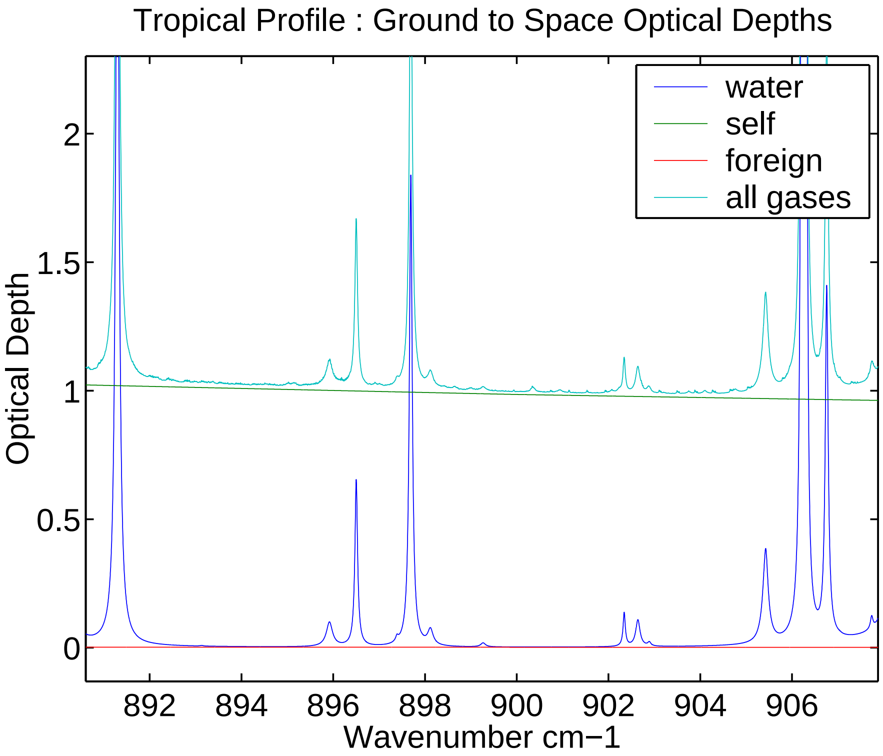


Figure 5: Water and total optical depths in 10 um window.

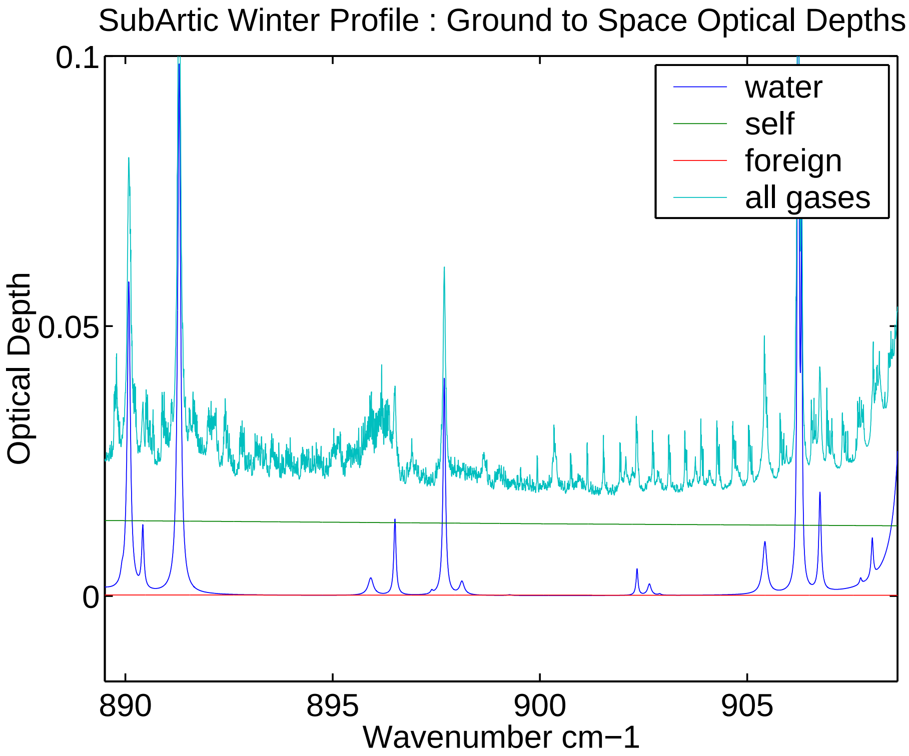


Figure 6: Water and total optical depths in 10 um window.

```
iaaOp(3,1)=60
iaaOp(3,2)=70
```

- will list layers 10,20 of gas ID 1
- will list layer 30 of gas ID 2
- will list layers 60,70 of gas ID 3

(if for instance, GAS ID 2 were not defined in MOLGAS/XSCGAS, the program will halt)

```
iaPrinter(1) = 1
iaGPMPAtm(1) = -1
iaNp(1) = 5
iaaOp(1,1)=1
iaaOp(1,2)=2
iaaOp(1,3)=3
iaaOp(1,4)=4
iaaOp(1,5)=5
```

will output the spectra for the 5 lowest layers for ALL gases.

```
iaPrinter(1) = 1
iaGPMPAtm(1) = 2
iaNp(1) = -1
```

will output the spectra for ALL kProfLayers layers for GAS ID = 2

Note that for options 1 these outputs can be layer to space cumulative, layer to space transmittances, layer or layer transmittances, depending on the relevant setting of parameter 1 kLayer2Sp in nm_params

14.8.2 Output mixed paths (Option 2 : IaPrinter(iI)=2)

For this iOutputOption (= 2), the user needs to specify iSet (mixed path set) and list of mixed paths to output. The program expects one of the following four combinations of iWhich iNp :

Table 11: Output options for iaPrinter(iI)=2

iaGPMPAtm	iaNP	DESCRIPTION
-1	-1	all mixed paths output
iSet	-1	for mixed path set = iSet, output all kProfLayers layers
-1	iN	for all mixed path sets iMP=1,iSetNum, output spectra in the iN layers specified in list on next line
iSet	iN	for mixed path set iSet, output spectra in the iN layers specified in list on the next line

If iaNP(iI) is not equal to -1, then the next line should contain a list of the iaNP(iI) mixedpaths to be output

Note that iaPrinter(iI) = 2 can be repeated more than once—the instructions found each time it finds iaPrinter(iI) = 2 are simply merged together with the previous instructions. Thus for instance

iaPrinter(1) = 2
iaGPMPAtm(1) = 1
iaNp(1) = -1
iaPrinter(1) = 2
iaGPMPAtm(1) = 2
iaNp(1) = -1

are perfectly valid, as the program will output the spectra for all kProfLayers layers for mixed path sets 1 and 2

Example : Suppose 3 gases have been found in nm_molgas and nm_xscgas and 4 mixed path sets (or 400 mixed paths) defined using nm_weight and 2 atmospheres defined in nm_radnce. The following are examples of what output mixed paths the user can specify.

iaPrinter(1) = 2
iaGPMPAtm(1) = -1
iaNp(1) = -1

will list ALL spectra for ALL mixed path sets. For each 10000 point chunk, the kProfLayers layers of set 1, kProfLayers layers of set 2, kProfLayers layers of set 3 and kProfLayers layers of set 4 will be output (10000 pts per layer).

iaPrinter(1) = 2
iaGPMPAtm(1) = 1
iaNp(1) = 2
iaaOp(1,1) = 10
iaaOp(1,2) = 20

iaPrinter(2) = 2
iaGPMPAtm(2) = 2
iaNp(2) = 1
iaaOp(2,1) = 30

iaPrinter(3) = 2
iaGPMPAtm(3) = 3
iaNp(3) = 2
iaaOp(3,1) = 60
iaaOp(3,2) = 70

- will list layers 10,20 of MP set 1
- will list layer 30 of MP set 2
- will list layers 60,70 of MP set 3

(if for instance, the user tried to output something from MP set 5, the program will halt as this set was undefined)

iaPrinter(1) = 2
iaGPMPAtm(1) = -1
iaNp(1) = 5
iaaOp(1,1) = 1
iaaOp(1,2) = 2
iaaOp(1,3) = 3
iaaOp(1,4) = 4
iaaOp(1,5) = 5

will output the spectra for the 5 lowest layers for ALL four mixed path sets.

iaPrinter(1) = 2
iaGPMPAtm(1) = 2
iaNp(1) = -1

will output the spectra for ALL kProfLayers layers for mixed path set number 2

iaPrinter(1) = 2
iaGPMPAtm(1) = -1
iaNp(1) = -1

will list all 400 mixed path spectra

Note that for option 2, these outputs can be layer to space cumulative, layer to space transmittances, layer or layer transmittances, depending on the relevant setting of parameter 1 kLayer2Sp in nm_params

14.8.3 Output radiances (Option 3 : iaPrinter(iI) = 3)

For iOutputOption = 3, the user needs to specify iAtm (atmosphere number). For this option, the program expects one of the following four combinations of iaGPMPAtm iaNp :

Table 12: Output options for iOutputOption=3

iaGPMPAtm	iaNp	DESCRIPTION
-1	-1	radiances for all layers in all atmospheres output
iAtm	-1	for atmosphere iAtm, output radiances at all layers (bottom or top, depending on up or down looking instrument)
-1	iN	for all atmospheres, output radiances at the iN pressures specified in the next line
iAtm	iN	for atmosphere iAtm, output radiances at the iN pressures specified in the next line

If iaNP(iI) is not equal to -1, then the next line should contain a list of the iNP pressures at which to output radiances.

As in the case of concatenating paths/mixed paths output, for a specific radiating atmosphere, if iOutputOption=3 is repeated more than once, the program tries to merge the pressures together, as long as no more than kProfLayers different pressures found for any one of the individual atmospheres. In addition , if iAtm < 0, as the program keeps on adding each atmosphere as a new print option, the program will halt if the total number of print options exceeds kMaxPrint. Thus

iaPrinter(1) = 3
iaGPMPAtm(1) = 1
iaNp(1) = 1
raaOp(1,1) = 0.01
iaPrinter(1) = 3
iaGPMPAtm(1) = 1
iaNp(1) = 1
raaOp(1,1) = 0.10

is allowed. However if kMaxPrint=5 and there are 6 atmospheres defined in nm_radnce, then

iaPrinter(1) = 3
iaGPMPAtm(1) = -1
iaNp(1) = -1

will cause the program to stop.

Example : Suppose as in the above example, 3 gases have been found in nm_molgas and nm_xscgas, 400 mixed paths defined using nm_weight and 2 atmospheres defined in nm_radnce (with the start/stop pressures being such that the first one has 90 layers and the other has 100 layers). The following are examples of what output radiances the user can specify

iaPrinter(1) = 3
iaGPMPAtm(1) = -1
iaNp(1) = -1

will list the radiances at the top of each layer (or bottom if the radiation traveling down to instrument on ground), for each atmosphere. Thus there will be 90 radiances from atmosphere 1, and 100 from atmosphere 2

iaPrinter(1) = 3
iaGPMPAtm(1) = 2

```
iaNp(1) = -1
```

will list the radiances at the top of each layer (or bottom if the radiation traveling down to instrument on ground), for atmosphere number 2. Thus there 100 radiances from atmosphere 2

```
iaPrinter(1) = 3
iaGPMPAtm(1) = -1
iaNp(1) = 2
raaOp(1,1) = 10.0
raaOp(1,2) = 0.01
```

will list two radiances for each atmosphere. These radiances are computed at pressures 10.0 and 0.01 mb (if one or both the atmospheres does not contain these pressures, the program will halt)

```
iaPrinter(1) = 3
iaGPMPAtm(1) = 2
iaNp(1) = 3
raaOp(1,1) = 700.0
raaOp(1,2) = 10.0
raaOp(1,3) = 0.01
```

will list three radiances for atmosphere 2. These radiances are computed at pressures 700.0, 10.0 and 0.01 mb (if the atmosphere does not contain these pressures, the program will reset the pressures so that they are within the max/min specified by the user in *nm_radnce*)

Note that the program takes into account the pressure layering, and the pressures where the radiance is to be output. As an example, suppose that the top pressure layer is from 10 mb to 0.005 mb , and the aircraft carrying the downlooking instrument flies at 5.0 mb. For downlooking instruments, the program uses the *lower* portion of the layer when doing its fractional layering computations, except for the bottommost layer (see below). So in this example, only fraction rF=(10-5)/(10-0.005)   1/2 = rFracTop of the top layer is used.

```
--------------

//////////////          use this LOWER portion
```

```
--------------
```

If a print option for this atmosphere is

iaPrinter(1) = 3

iaGPMPAtm(1) = 1
iaNp(1) = 1
raaOp(1,1) = 00.0

implying that one radiance, at aircraft height, is to be output, then the program will compute the output radiance as follows :
the radiance immediately below this layer is the incident radiation. A fraction 0.5 (=rFracTop) of the layer is to be used in the radiative transfer. The program interpolates the vertical temperatures, to find the temperature of this fractional layer, and then uses rFracTop of the computed absorption coefficients, to find the absorption in this fractional layer. These are then used, together with the radiation at the bottom of the layer, to compute the radiance at the instrument.

If a print option for this atmosphere is

iaPrinter(1) = 3

iaGPMPAtm(1) = 1
iaNp(1) = 1
raaOp(1,1) = 7.5

(ie the user wants a radiance output at pressure 7.5 mb), the program will compute the output radiance as follows :
the radiance immediately below this layer is the incident radiation. A fraction 0.25 (= (10-7)/(10-0.005)) of the layer is to be used in the radiative transfer. The program interpolates the vertical temperatures, to find the temperature of this fractional layer, and then uses rFrac of the computed absorption coefficients, to find the absorption in this fractional layer. These are then used, together with the radiation at the bottom of the layer, to compute the radiance at the instrument.

As mentioned immediately above, the program is very careful about the bottommost layer, and will use the *top* fraction of this layer (between the ground and specified pressure level). For example, if the bottom layer is from 1000 mb to 900 mb, and the surface pressure is 950 mb, then only fraction rF=(950-900)/(1000-900) 1/2 = rFracBot of the bottom layer is used

```
--------------
//////////////         use this UPPER portion
//////////////
```

If a print option for this atmosphere is
iaPrinter(1) = 3

iaGPMPAtm(1) = 1
iaNp(1) = 1
raaOp(1,1) = 1000.0
this means that the radiation at the surface is output :
raRad(vu) = raEms(vu) * ttorad(Tsurf,vu) + thermal(vu) + solar(vu)

If a print option for this atmosphere is iaPrinter(1) = 3

iaGPMPAtm(1) = 1
iaNp(1) = 1
raaOp(1,1) = 925

implying that one radiance, at 925 mb, is to be output, then the fraction of the bottom layer that will be used is rFracBot - (925-900)/(1000-9000) = 0.5 - 0.25 = 0.25. The program will compute the output radiance as follows :
the radiance from the surface is used as the incident radiation. A fraction 0.25 of the layer is to be used in the radiative transfer. The program interpolates the vertical temperatures, to find the temperature of this fractional layer, and then uses rFrac of the computed absorption coefficients, to find the absorption in this fractional layer. These are then used, together with the surface radiation, to compute the radiance at the instrument.

If the user specifies radiances to be output at EACH layer

iaPrinter(1) = 3
iaGPMPAtm(1) = 1
iaNp(1) = -1

then for each layer, the radiation at uppermost part of the layer is output
ie bottom layer : surface to top of layer –¿ output rad
next layer : bottom of layer to top of layer –¿ output rad
....
top layer : bottom of layer to aircraft posn –¿ output rad

The same considerations would be applied for a uplooking instrument.

Note that if you use the clear sky radiative transfer code, you can output radiances at as many levels as you desire; however if you turn on the scattering code, at present you can only output the radiance at the TOA (for downlook instrument) or at surface (for uplook instrument); if some other option is found in the namelist, the code will stop.

Also note how if iaPrinter(iI) = 1,2 we had to specify the gas paths or mixed paths in iaaOp(i,j), but if iaPrinter(iI) = 3 we had to specify the pressure levels in raaOp(i,j)

14.9 *nm_jacobn* (optional)

This keyword turns on/off Jacobian calculations. The user can either ask to output the first kMaxDQ gases, or give a list containing which gas IDs he wants to output gas amount Jacobians for, or turn off jacobian computations :

iJacob = -1

OR

iJacob = n

g1 g2 ... gn

OR

iJacob = 0

If$iJacob = 0$, then no Jacobian computations are done.

Depending on the setting of parameter $kJacobOutput$ in the PARAMS section, the jacobians that are output could be the raw jacobians (dr/ds), the brightness temperature jacobians (d(BT)/ds) or the scaled brightness temperature jacobians (d(BT)/ds $\times\delta$ s) where s is the variable with respect to which the jacobian is being computed (gas amount, temperature or surface parameter) ($kJacobOutput = -1, 0, 1$).

In addition, depending on the setting of parameter $kTempJac$ in the PARAMS section, the temperature Jacobian could be computed using the temperature dependence of only the planck terms, the transmission terms or both ($kTempJac = -2, -1, 0$)

For a downlooking instrument, the code will output gas jacobians for all gases found in the list in the JACOBN section, temperature jacobians, weighting functions, and four surface jacobians : the radiance change with respect to surface temperature, surface emissivity, thermal background and solar background.

For an uplooking instrument, the code will output gas jacobians for all gases found in the list in the JACOBN section, temperature jacobians, weighting functions. Since surface terms are meaningless here, four sets of zeros are also output.

If the user wants kCARTA to do scattering computations, then jacobian computations can be done, by assuming that the cloud (or scattering volume) only has an abosprptive component. If the user imports spectra for some gases from some line by line code, no jacobian computations can be done as kCARTA cannot "perturb" the gases in temperature.

WARNING : kCARTA has been structured so that the total water optical depth is broken into that of three components, according to the CKD definitions : gasID $=$ 1 being the without basement lorentz term, gasID $=$ 101 being the self continuum and gasID $=$ 102 being the foreign continuum. An unsuspecting user might only turn gasID $=$ 1 on, and be puzzled why the water jacobians seen in the output files seems small (eg in the 10 um window region, the main contribution is due to the self continuum). Thus the user should ask for gases 1,101 and 102 to be output!!!!

14.10 *nm_nonlte* (optional)

The default behaviour of kCARTA is to assume the entire atmosphere is in local thermodynamic equilibrium. This assumption is used in compiling the kCompressed Database using our UMBC-LBL code. However, AIRS observations, and indeed observations by many limb viewers, indicate that some molecules are not in LTE in the upper layers of the atmosphere. In particular, the 4 um band of CO2 is significantly in NonLTE above 60 km, during daytime.

To account for this, we have implemented a NonLTE capability into kCARTA, optimized and tested for the 4 um CO2 bands. However, since the line shapes and planck function modifiers need to be computed on the fly, for the many CO2 lines in this region of the spectrum, the code is significantly slowed down!!!

In addition, to simplify the computations, we mainly use Cousin chi functions instead of line mixing in this region. However, for the R branch of the strong $\Sigma - \Sigma$ band, in the important temperature sounding region (2380 to 2430 cm^{-1}), the code does a first order linemixing times an appropriate "chi" function. This should not really affect the results too much, as NLTE is in the higher parts of the atmosphere (stratopause), mininising the effects of linemixing and thereby rendering the Cousin lineshape as "good enough." In the lower part of the atmosphere (tropopause), pressures are high enough to warrant a linemixing lineshape; in this region the atmosphere is in LTE, and kCARTA essentially uses its default (linemix) database here.

The code works as follows. It first uncompresses the optical depths for all layers, for the gas in question (ie it assumes all layers are in LTE). It then does a LBL computation for the upper layers of the atmosphere, using parameters and data supplied below.

The information required in this section is of the following form

iNumNLTEGases =

iNLTE_SlowORFast =
iaNLTEGasID(1..N) =
raLTEStrength(1..N) =
raNLTEstart(1..N) =
iaNLTEChunks(1..M) =
iaaNLTEChunks(1..N,1..M) =
caaStrongLines(1..N) =
caaUpperMixRatio(1..N) =
caWeakCO2Path =
iaNLTEBands(1..N) =
caaNLTETemp(1..N) =
caaNLTEBands(1..N,1..P) =

$iNLTE_SlowORFast$ tells the code whether to use the slow accurate LBL model (+1) or regression coefficients used by the fast model (-1). If you use the Fast Version (-1), then the only namelist parameters you need to set are iNumNLTEGases,iNLTE_SlowORFast,iaNLTEGasID(),iaNLTEChunks(a), iaaNLTEChunks(a,b)
$iNumNLTEGases$ tells the code the number of gases that are in NLTE. If NLTEGases = -1, then all gases are in LTE (no gas in NLTE). The code uses the defualt kCompressed Database for *all* gases.

If NLTEGases ¿ 0, then one or more gases are in NLTE; all other gases are in LTE, and the code uses the default kCompressed Database for these gases; for the N gases that are in NLTE, the code will perform line by line computations based on information specified further in this section.
$iaNLTEGasID(1..N)$ gives the HITRAN gas IDs of the N gases in NLTE
$iSetBloat$ tells code whether to use default (-1) 0.0025 cm^{-1} spacing or to use bloted (+1) 0.0005 cm^{-1} resolution.
If iNLTE_SlowORFast = +1, this is automatically (re)set to -1
$iDoUpperAtmNLTE$ tells code whether to do NLTE calcs upto 80 km (default (-1) or upto 120 km (+1)
If iNLTE_SlowORFast = +1, this is automatically (re)set to -1
$raNLTEstart(1..N)$ tells for each gas in NLTE, at which height (in km) to start the NLTE computations. This height information is turned into the relevant layer; for all layers below this, the code uses the optical depths from the kCompressed Database.
$raLTEStrength(1..N)$ is akin to the "weight" section; this tells the code the multi-

plier for the spectral computations performed by the code. Obviously this parameter will usually be 1.0 for almost all cases.
You can be very mischievous and use a different database for CO2 here!!!!!
If gasID == 2, raLTEStrength(X) ¡ 0 then kCARTA will go ahead and say, aha, you do not want NLTE but you simply want to substitute the LINEMIX compressed database optical depths for eg COUSIN compressed database optical depths. So kCARTA will simply substitute the optical depths of the upper layers with data from *kCousin_CO2Path*. So if you cleverly set the value of raNonLTEStart(X) = -1400, then *all* layers will be substituted, not just the upper layers!!!!! Devilish, eh?. A nice parameter to use for raLTEStrength(X) in this case is -1.1212, which is 370/330 ppmv used in the newer linemix database vs that used in the older Cousin database
$iaNLTEChunks(1..N)$ tells for which chunks the above gases are in NLTE
$iaaNLTEChunks(1..N, 1..M)$ for each of the N gases in NLTE, we now know how many chunks are in NLTE. iaaNLTEChunks specifies these chunks.
$caaUpperMixRatio(1..N)$ tells the code where the stratosphere/ mesosphere mixing ratios can be found. Remember the default kCARTA database extends from 0 to 80 km, while NLTE is important between about 40-120 km; so kCARTA can compute optical depths and Planck modifiers above the standard database, as long as it knows the ppmvs! Right now it scales the MR read in from this file, to 370 ppmv at bottom
$caaStrongLines(1..N)$ for each of the N gases in NLTE, we now know how many chunks are in NLTE. The gas molecules have many lines/bands; from $caWeakCO2Path$ there are the weak background lines, and also there are strong lines in bands. Some bands are in LTE while others are in NLTE. caaStrongLines is a file that specifies all the bands for the "strong" lines that could be in LTE or NLTE. Depending on what bands are specified to be in NLTE in $caaaNLTEBands(1..N, 1..P)$, an XOR is done with the bands in $caaStrongLines$ to compute the optical depth contribution due to strong bands in LTE (the NLTE lines will have their optical depths computed separately). The files listed in this MUST have the FULL path and name
The files listed must agree with those listed by appending $caStrongLineParams$ in the "NLTEBandMapper" subroutine, else the code gets very perplexed
$iaNLTEBands(1..N)$ tells for each gas, how many bands are in NLTE
$caaaNLTEBands(1..N, 1..P)$ for each of the N gases in NLTE, give the igasID,iISO,iUSGQ numbers so that kCARTA maps these partams to a file that has the weak HITRAN line parameters (such as line centers, broadening coefficients etc). The lineshapes for these strong lines will be Voigt * Cousin if the gas is CO2; else they will just be Voigt
$caaNLTETemp(1..N)$ for each of the N gases in NLTE, give the file that has the NONLTE temperatures and vibrational partition function terms, in GENLN2 style. See

the f90 programs supplied by M. Lopez-Puertas and B. Funke in KCARTA/SRCv1.12/NONLTE2/sergio/VIB
as well as
/KCARTA/SRCv1.12/NONLTE2/sergio/AIRSDATA_NLTE_VIBTEMPS_Puertas/do_makeVT.m
for a script that takes in an input kinetic temp profile and dumps out the NLTE temperature profiles

If the user puts the name 'nlteguess', kCARTA will use a polynomial guess of the NLTE temps, and output the guessed profile into
caOutName_VTX : this is the GENLN2 style file
caOutName_VTX.sss : this is a summary of the GENLN2 style file
where "X" is the closet regression profile (of 48) to the user profile, between 800 and 25n mb (2 - 25 km). The .sss file will have 15 columns :
ii,P,Tk,[QV1,QV2,QV3,QV4],[S1,S2,S3,S4],[P1,P2],[D1,D2]
where the $QVi, i = 1,4$ are the vib partition functions, the SSi,PPi,DDi are the isotopic sigmasigma, pipi, deltadelta NLTEs.

The example below is primed for the 4 um CO2 band.

- iNumNLTEGases = +1 : says that one gas is in NLTE
- iNLTE_SlowORFast = +1 : says that slow accurate LBL model
- iaNLTEGasID(1) = 2 : says the gasID of the gas is "2" (CO2)
- raNLTEstrength(1) = +1.0 : says use a weight of 1.0 for all the upper layer optical depths computed for this gas.
  Together with parameter kCousin_CO2path, this lets kCARTA do cousin lineshape computations via the following fudge : if raNLTEstrength(X) = -1.1212 and iaNLTEGasID(X) = 2, kCARTA not do NLTE but instead swaps the linemix uncompressed spectra for cousin uncompressed spectra in the layers above that specified by raNONLTEstart(X)
  Otherwise raNLTEstrength is really not used
- raNLTEstart(1) = 45.0 : tells the code to start the NLTE computations above 45.0 km (which is about layer 90 for the standard AIRS layers)
- iaNLTEChunks(1) = 10 : says that 10 chunks are in NLTE
- iaaNLTEChunks(1,1..10) gives the 10 chunks that are in NLTE

- caaLTEWeakLines(1) gives the file that has the parameters for the weak lines that are in LTE.
- iaNLTEBands(1) = 8 : says that for this gas, there are 8 bands in NLTE
- caaaNLTEBands(1,1..8) are strings that give the (iGasID iISO iLSGQ iUSGQ) numbers so kCARTA finds the names of the files that contain the HITRAN lineparameters of the vibrational bands in NLTE. Any other info in the string is ignored; the info is then mapped onto UMBC-LBL identifiers so that kCARTA can open the correct files containing the line paameters for the band in question. At present only CO2 4um bands can be included.
- caaNLTETemp(1) gives the names of the file that contains the NLTE profiles, as well as the vibrational partition functions, in GENLN2 style.

Following is a typical NLTE identifier section for the 4.3 um CO2 region. Note how we identify the sigma-sigma, pi-pi and delta-delta lines.

```
iNumNLTEGases    =             +1
iNLTE_SlowORFast =             +1

iaNLTEGasID(1)      =       2
raNLTEstrength(1)   =       +1.0
raNLTEstart(1)      =       45.0
iaNLTEChunks(1)        =       10
caaNLTETemp(1)   = '/home/sergio/AIRSCO2/NONLTE/hit2350_day_profile3A'
caaLTEWeakLines(1) ='/home/sergio/AIRSCO2/NONLTE/CO2/co2_2230.dat'

iaaNLTEChunks(1,1)  =        2230
iaaNLTEChunks(1,2)  =        2255
iaaNLTEChunks(1,3)  =        2280
iaaNLTEChunks(1,4)  =        2305
iaaNLTEChunks(1,5)  =        2330
iaaNLTEChunks(1,6)  =        2355
iaaNLTEChunks(1,7)  =        2380
iaaNLTEChunks(1,8)  =        2405
iaaNLTEChunks(1,9)  =        2430
iaaNLTEChunks(1,10) =        2455
```

```
iaNLTEBands(1)     = 19
!!! uses strongest  sigma-sigma, pi-pi, delta-delta
!!! 2350 .. 2354 = sigma-sigma
!!! 2320 .. 2322 = pi-pi
!!! 2310 .. 2312 = delta-delta
!!!                 GASID   GASIso  iLSGQ     iUSGQ   run7lblID
caaaNLTEBands(1,1) ='2        1       1         9        2350'
caaaNLTEBands(1,2) ='2        2       1         9        2351'
caaaNLTEBands(1,3) ='2        3       1         9        2352'
caaaNLTEBands(1,4) ='2        4       1         9        2355'
caaaNLTEBands(1,5) ='2        1       2         16       2320'
caaaNLTEBands(1,6) ='2        2       2         16       2321'
caaaNLTEBands(1,7) ='2        1       4         24       2310'
caaaNLTEBands(1,8) ='2        2       4         24       2311'
caaaNLTEBands(1,9) ='2        1       3         23       2353'
caaaNLTEBands(1,10)='2        1       5         25       2354'
!!!these are the ones Manuel suggested adding on; some isotopes of above
caaaNLTEBands(1,11)='2        2       3         23       2253'
caaaNLTEBands(1,12)='2        2       5         25       2254'
!!!these are the others Manuel suggested adding on
caaaNLTEBands(1,13)='2        1       2         15       2110'
caaaNLTEBands(1,14)='2        1       3         25       2120'
caaaNLTEBands(1,15)='2        1       5         23       2140'
caaaNLTEBands(1,16)='2        1       6         36       2160'
caaaNLTEBands(1,17)='2        1       7         37       2170'
caaaNLTEBands(1,18)='2        1       8         38       2180'
caaaNLTEBands(1,19)='2        3       2         16       2322'
!!!these one never seems to exist in the NLTE profiles
caaaNLTEBands(1,20)='2        1       3         22       2150'
caaaNLTEBands(1,21)='2        1       4         22       2130'
$end
```

In addition, kCARTA can output a file containing the Planck modifiers. This file is in the same output format as the flux file. Control for turning this on/off is thru parameter kFlux in *nm_param*; set to -1,1,2,3 (off) or 0 (on)

Figure 7 shows a comparison plot of kCARTA vs GENLN, compared to some actual NLTE data seen in daylight viewing conditions on the AIRS instrument. What is plotted for AIRS, is actual NLTE dayime observations minus LTE Fast Model computations. For KCARTA and GENLN2, we plot (NLTE - LTE) calculations, with the TOA at about 85 km. The main CO2 Sigma-Sigma, Delta-Delta, Pi-Pi bands are in NLTE while the weak background lines are in LTE.

14.11 *nm_scattr* (optional)

If $kRTP = -2, -1, 0$ this means the user will be defining the atmosphere and any scattering parameters, using the specifications stated below. If $kRTP = +1$ then kCARTA will use the atmosphere and cloud definitions specified in the RTP file that contains the profile; the documentation can be found in KLAYERS/Src/Doc.

This keyword turns on/off scattering calculations. kCARTA can interface with one of five scattering codes, depending on the users needs.

The first is $rtspec.f$ written by K. F. Evans of the University of Colorado at Boulder. This code computes the scattering using one of three models : Single Scattering(1), Eddington(2) or a Hybrid(3) combination of the first two. This code is very fast, but it does not allow the user to include scattering by a beam.

The second is $disort.f$ written by K. Stamnes $et.al.$ To speed up the running of this part of the code, kCARTA allows the user to compute the scattering using one of three ways. These ways essentially allow the user to compute radiative transfer on N out of the 10000 points per chunk and then interpolate onto the rest of the wavenumber grid. The three ways differ in the way they choose the wavenumber points to do the radiative transfer on. This code is slow, but it does allow the user to include scattering by a beam.

The third is a twostream code that also allows for the inclusion of a solar beam. This code has been developed here at UMBC, and is very fast.

The fourth is a languishing code (perturbative solution to Schwatzchild) and will be implemented once Mathematica can figure out the required integrals. (it WILL work, bloody hell!!!)

The fifth code also allows for the inclusion of a solar beam. This code is based on the longwave parametrizations and is also very fast, and transperant enough to allow for jacobians to be computed.

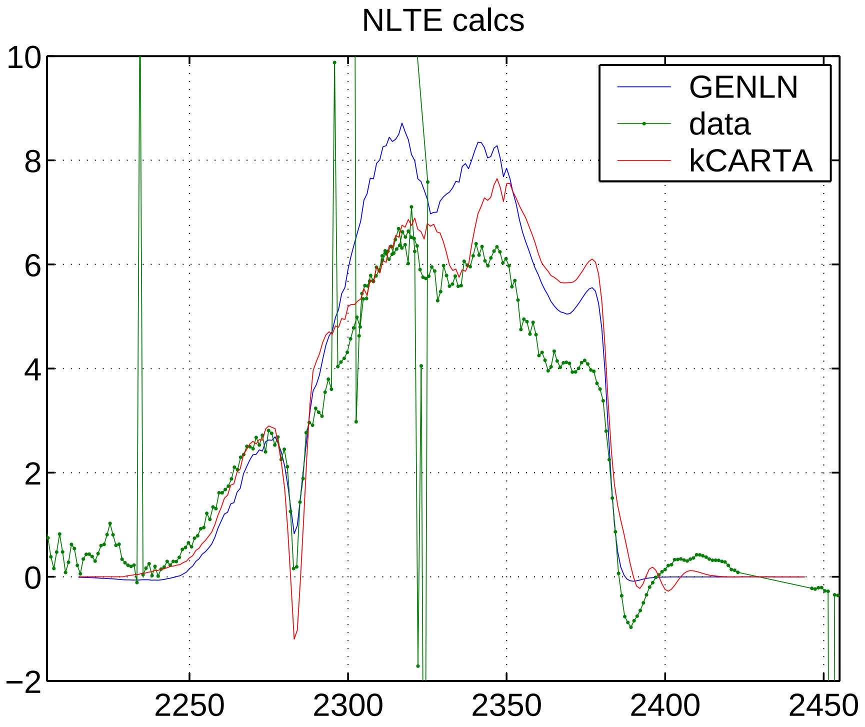


Figure 7: NLTE plots : Comparison of kCARTA, GENLN to actual AIRS data.

14.11.1 kRTP = -2,-1,0

The user has to tell the code which of the above five scattering models $kWhichScatterCode$ to use (1 $= TWOSTREAM$, 2 $= RTSPEC$, 3 $= DISORT$, 4 $= SIMPLE$, 5 $= PCLSAM$). If the user chooses $RTSPEC$ or $DISORT$, then he/she has to state which specific submodel $kScatter$ to use (eg for $RTSPEC$, use Hybrid or Eddington, for $DISORT$, use frequency or optical depth interpolation). The user also has to state how many clouds $iNumClouds$ are being defined, and then has to define the clouds. In addition, the user has to indicate in which atmosphere a cloud should be used. If the user expects Jacobians to be computed, the radiative transfer code uses the specified model $(TWOSTREAM, RTSPEC, DISORT, SIMPLE, PCLSAM)$, but always uses the $PCLSAM$ model to compute jacobians.

For the code to interface to any of the scattewring codes, the user has to set the number of clouds in this section to a value greater than zero. If $iNclouds$ is less than 1, the code will not do a scattering computation.

Note that a cloud has to be built up so that it occupies adjacent pressure layers eg it can occupy layers 12,13,14 but not 12,14,15. Also, the cloud must occupy full layers.

Clouds that occupy completely different heights can be processed. For example if cloud 1 is an aerosol cloud layer from KCARTA layers 4-5, and cloud 2 is a cirrus cloud from kCARTA layers 43-46, $TWOSTREAM$ and $RTSPEC$ handle this by setting a “third” cloud from layers 6-42, with IWP=0.0. $DISORT$ would simply just use the original two clouds, as widely separated as they are.

The information required in this section is of the following form

kWhichScatterCode =

kScatter =
kDis_Pts =
kDis_nstr =
iScatBinaryFile =
iNClouds =

Cloud 1 definitions
Cloud 2 definitions
...
Cloud N definitions

$kWhichScatterCode$ = +5 for PCLSAM, +4 for SIMPLE, +3 for DISORT, +2 for

RTSPEC, +1 for $TWOSTREAM$. Note that all but DISORT run very fast, but RTSPEC does NOT allow a solar beam.

- If $kWhichScatterCode = +1$, (use $TWOSTREAM$) then $kSCatter$ tells the code how many times to run the radiance thru the cloud, to get better values for the incident radiation at cloud top and bottom, and hence better guesses at radiation exiting cloud top and bottom. This parameter should be set between 1 and 3. After doing quite a few tests, the best setting of parameter $kScatter$, when $TWOSTREAM$ is used, is 1
- If $kWhichScatterCode = +2$, (use $RTSPEC$) then $kScatter$ = -1 if the Single Scattering model is to be used, 2 if the Eddington model is to be used, or 3 if the hybrid model should be used. *It is best to use the Hybrid model when using $RTSPEC$* After doing quite a few tests, the best setting of parameter $kScatter$, when $RTSPEC$ is used, is 3
- If $kWhichScatterCode = +3$, (use $DISORT$) then everything defaults to $kScatter = +1$ for now, as this seems the best option for both uplooking and downlooking instruments After doing quite a few tests, the best setting of parameter $kScatter$, when $DISORT$ is used, is 1

If $kWhichScatterCode = +3$, (use $DISORT$) then

- Everything defaults to $kScatter = +1$ for now, as this seems the best option for both uplooking and downlooking instruments After doing quite a few tests, the best setting of parameter $kScatter$, when $DISORT$ is used, is 1
- $kScatter = +1$ means use only kDis_Pts pts (equally spaced over the kMaxPts) to do radiative transfer, and then interpolate in wavenumber to estimate the intensities for the rest of the wavenumber grid. *It is best to use this model for either an up or down look instrument.*

  If the instrument is uplooking, then a simple straightforward interpolation of the radiance at the chosen wavenumber points, to the entire grid, is done.

  If the instrument is downlooking, then a more elaborate computation is done. First the nonscattering radiative transfer is performed at all points on the wavenumber grid. The output radiance is then the nonscattered radiance, plus an interpolation of scattered radiance onto the required point. Let raW,$raNS$ be the entire $kMaxpts = 10000$ wavenumber grid, and associated clear sky radiances. Similarly, let raX, raY be the wavenumber points at which DISORT was called, along with the computed scattered radiance. Then the correct radiance raI(iF) at wavenumber point raW(i) is estimated to be

$$raI(i) = raNS(i) + \delta I$$

where δI is an interpolation in Temperature space, then converted back to a radiance. In other words, for point raW(iF), find which wavenumber points in raX bisect this $W0 \leq raW(iF) \leq W1$. Find the DISORT and clear sky radiances at these bisecting points, and compute the temperature differences between scattered and non scattered radiances, at these two points

$$T0 = ttorad(W0, ScatterRad(W0)) - ttorad(W0, NonScatterRad(W0))$$

$$T1 = ttorad(W1, ScatterRad(W1)) - ttorad(W1, NonScatterRad(W1))$$

Then interpolate to find the estimated correction to the non scattering temperature

$$\delta T = \frac{T1 - T0}{W1 - W0} \times (raW(iF) - W0) + T0$$

Add on the non scattering temperature, and change this to a radiance

$$T_{scatter} = ttorad(raW(iF), NonScatterRad(raW(iF))) + \delta T$$

$$I_{estimate} = radtot(raW(iF), T_{scatter})$$

By doing the interpolations in temperature, we can account for the variation of Planck radiance with wavenumber (which should then have the spectral line dependancies in it).

- $kDis\_Pts$ is the number of points tha kCARTA will compute the radiances at, out of the 10000 points per chunk. So if this variable is set at eg 25, then kCARTA will ask $DISORT$ to compute the radiance only at 25 points, and then interpolate the results onto the remaining 9975. This obvioulsy sppeds up the code, but the user has to be careful! A low setting, such as 25, for a downlook instrument is fine. But things get quite bad for the same setting for an uplook instrument; a setting of 500 might be needed (but this of course means that the code runs much more slowly!)

- $kScatter = +2$ means step thru kDis_Pts pts that are chosen in the following way. Find the optical depths for the layer closest to ground, and sort them from smallest to largest. Choose the kDis_Pts points that have the smallest optical depths. Do radiative transfer on these points, and then interpolate in optical depth to estimate the intensities for the rest of the wavenumber grid.

- $kScatter = +3$ means step thru kDis_Pts pts that are chosen in the following way. Find the optical depths for the layer closest to ground, and sort them from smallest to largest. Choose the kDis_Pts points that span this, equally spaced from smallest to largest optical depth. (This should not be confused with a $correlated\ k$ method, since we do not do radiative transfer on multiple correlated k distributions as the vertical structure of the atmosphere changes). Do radiative transfer on these points, and then interpolate in optical depth to estimate the intensities for the rest of the wavenumber grid.

- If $kScatter = 2, 3$ then for an uplook instrument, there is simply an interpolation of scattered radiance in wavenumber space. In other words, the scattered radiance is computed at discrete wavenumbers, *that are chosen because of their ordering in lowest layer optical depths*. For the rest of the points in the grid, just do a simple interpolation of the scattered computations at these chosen optical depths, onto all the optical depths (and hence all wavenumbers) found at the lowest layer. This method works very well for an uplook instrument, as most of the radiance measured by the instrument depends on the layer closest to the instrument.

- For a downlook instrument, there is a more elaborate computation done, exactly along the lines of that for a wavenumber interpolation for the downlooking case described earlier. However, since the wavenumber dependence of the Planck function is quite crucial, the code proceeds as follows. First it chooses the wavenumber points according to the sorted optical depths of the lowest layer, as described above. It then does the $DISORT$ radiative transfer on the chosen points, and then non scattering radiative transfer on all points. To interpolate the chosen points onto the wavenumber grid, it resorts the chosen points according to wavenumber, and interpolates these few points onto the entire wavenumber grid, as described above for $kScatter = 1$. However, since the wavenumbers chosen for the downlooking $kScatter = 2, 3$ cases do not necessarily sample the $25cm^{-1}$ chunk adequately in wavenumber, there are noticeable errors when compared to $RTSPEC$ or to $kScatter = 1$

If $kWhichScatterCode = +3$, then $kDis\_nstr$ tells how many streams should be used by DISORT (2,4,8,16 ...)

If $kWhichScatterCode = +3$, then $kDis\_Pts$ is used as described above.

$iScatBinaryFile = 1$ if the scattering tables produced by $sscatmie.f$ have been translated to binary format, else it should be set to -1 if they are still in the original text file format.

$iNClouds$ is the number of clouds which are defined in this section, to be be used with the $rtspec$ code. If this number is less than one, than only clear sky computations will be done.

Each of the the clouds are defined and used as follows

raExp(iI) = rE
iaCloudNumLayers(iI) = iLayers
raaPCloudTop(iI,iJ) = rPTop
raaPCloudBot(iI,iJ) = rPBot
raaaCloudParams(iI,iJ,1) = IWP
raaaCloudParams(iI,iJ,2) = dme = mean particle size
iaaScatTable(iI,iJ) = iTable
caaaScatTable(iI,iJ) = Mie_FileName
caaCloudName(iI)
iaCloudNumAtm(iI) = iNatms
  iaaCloudWhichAtm(iI,iJ)

The first line indicates how the IWPs will be scaled. If the cloud is "expanded" to occupy pressure layers spanning $rPTop$ to $rPBot$, then the IWPs can either be the same in each of the layers (rE $\leq$ 0), or they can exponentially decrease from bottom to top as

$$rIWP = rIWP0 \times exp^{(-rE*(rPBot-rP)/(rPBot-rPTop))}$$

where $rIWP0$ is the IWP specified by the user, and rP is the pressure of the current layer.

The second line gives the (integer) number of pressure layers $iLayers$ the cloud will occupy. These pressure layers *must* correspond to the layers defined by the pressure levels from *klayers.x*.

In the next few lines, the user then has to define, for each of the layers, the cloud parameters; starting from the lowest pressure (highest layer), the user has to enter :
rPTop, rPBot : real variables defining the cloud layer top, bottom pressure
IWP : real parameter IWP/LWP in g/m^2
dme : real parameter mean particle diameter (um)
iTable : integer giving scattering table number (integer)
Mie_FileName : character array gives the Mie Scattering table file name

The user can then baptise the cloud with a name (in a character array). Finally the user has to tell the code in which atmosphere(s) to use the defined cloud. This is done in the next two lines :
iaNatms : integer that tells how many atmospheres the cloud has to be used
iaaCloudWhichAtm : integers listing these atmospheres

Note that the code automatically expands a cloud layers into multilayers spreading across the specified pressure levels eg if a user specifies a "one layer cloud" from 250 to 200 mb, then kCARTA will expand this so that the start (top) layer is at 200 mb, and the stop (bottom) layer is 250 mb.

So for instance the following lines define a cloud that occupies three pressure layers, and will be used in two atmospheres

```
kScatter = 2
iScatBinaryFile = 1
iNclouds = 1
iaCloudNumLayers(1) = 3

    raExp(1) = 0.0
    raaPCloudTop(1,1) = 2.0000000E+02
    raaPCloudBot(1,1) = 2.5000000E+02
    raaaCloudParams(1,1,1) = 5.0
    raaaCloudParams(1,1,2) = 20.0
    iaaScatTable(1,1) = 1
    caaaScatTable(1,1) = 'cir1'

    raExp(2) = 0.0
```

```
        raaPCloudTop(1,2) = 2.5000000E+02
        raaPCloudBot(1,2) = 3.0000000E+02
        raaaCloudParams(1,2,1) = 8.0
        raaaCloudParams(1,2,2) = 30.0
        iaaScatTable(1,2) = 2
        caaaScatTable(1,2) = 'cir2'

        raExp(3) = 0.0
        raaPCloudTop(1,3) = 3.0000000E+02
        raaPCloudBot(1,3) = 3.5000000E+02
        raaaCloudParams(1,3,1) = 10.0
        raaaCloudParams(1,3,2) = 40.0
        iaaScatTable(1,3) = 3
        caaaScatTable(1,3) = 'cir3'

    caaCloudName = 'HappyLittleCloud'
    iaCloudNumAtm = 2
    iaaCloudWhichAtm = 1, 2
```

The code automatically compares the cloud top and bottom pressures against the start/stop pressures of the atmospheres in RADNCE; if discrepancies are found, the code will stop running.

WARNING 1 : If one uses DISORT for a down looking instrument, with solar on, then DISORT automatically uses a solar reflectance = albedo/π (which is the same as that used for the reflected thermal).
However, with nonscattering kCARTA, one can specify the solar reflectance, using parameter cakSolarRefl(i). If this is set to -1, then the results for nonscattering kCARTA and cloudless DISORT should be the same. If this is set to a non zero positive number, then the results for nonscattering kCARTA and cloudless DISORT will be different.

WARNING 2 : At present, both DISORT and RTSPEC can only output a radiance at TOA or aircraft posn (for a down look instr), or at ground level (for a uplook instrument).

WARNING 3 : DISORT is extensively tested by its authors (Stamnes et al). However, it runs much slower than non scatter kCARTA or RTSPEC. Instead of computing radiative transfer at all points, kCARTA skips thru its wavenumber points, and then interpolates the results back onto the 0.0025 cm^{-1} grid.

For a 10000 point chunk, typical timings are 30 sec for kCARTA nonscatter (both up and downlook)m 35-40 sec for RTSPEC (up and downlook). For DISORT, with kDis_Pts = 200, for a downlook instrument, sun on, takes about 240 secs, while for an uplook instrument, sun on, takes about 1200 secs).
To speed up the code, the user can do one or both of the following. In the input namelist file, one can reduce the number of streams used in the computation (kDis_nstr, preset to 16), or decrease the number of points at which radiances are computed (kDis_Pts, preset at 400).

WARNING 4: For a downlook instrument, iakThermal(i) is only relevant for RTSPEC (ie this parameter turns it off or on). DISORT will always compute the background thermal.

WARNING 5: For a downlook instrument, iakThermal(i) should be turned off when TWOSTREAM is used, for better comparisons to DISORT

Figure 8 is a plot of the brightness temperatures for 4 cases, using the RTSPEC code (sun off). The instrument is downlooking. Case 1 is clear sky, case 2 is cirrus cloud at 10 km, third is water cloud at 1 km and fourth is aersol dust at ground. Mean particle sizes of 10 um were used for all three scattering cases, while the particle concentrations were 1.0,1.0,10.0 g/m2 respectively.

Figure 9 is a plot of the brightness temperatures for the same 4 cases, using the TWOSTREAM code (sun on). The instrument is uplooking.

14.11.2 kRTP = +1

Since the $AIRS$ Fast Forward Model uses a one layer version of our twostream code, this model is automatically selected when the RTP file contains information to include a cloud in the atmosphere. Many of the variables needed above are then automatically set by kCARTA; only the very needed information is read in from the RTP file. Only ONE cloud can be read in and used from the RTP file.

```
a) kWhichScatterCode      = +1        !use TwoStream
b) kScatter               = 1         !use one run of TwoStream
c) raExp(1)               = 0.0
d) iScatBinaryFile        = cbinORasc     --> set in nm\_prfile
e) iNClouds               = 1
f) iaCloudNumLayers(1)    = 1
g) iaaScatTable(1,1)      = 1
h) caaCloudName(1)        = 'RTP cloud'
i) caaaScatTable(1,1)     = cfile         --> set in nm\_prfile
```

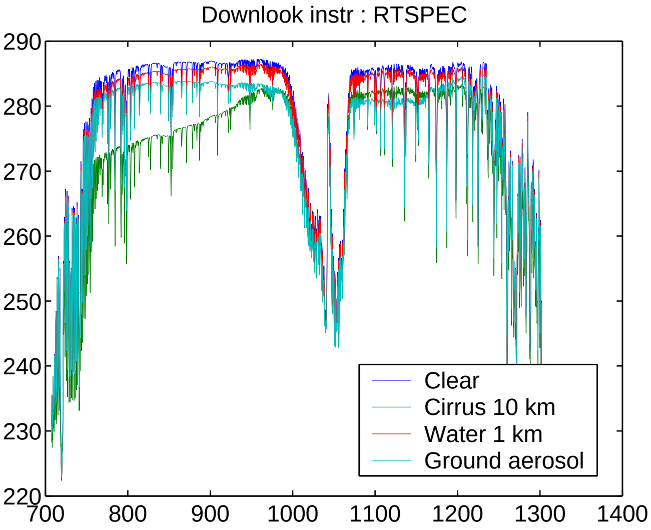


Figure 8: Scatter plots : Downlook instrument using RTSPEC

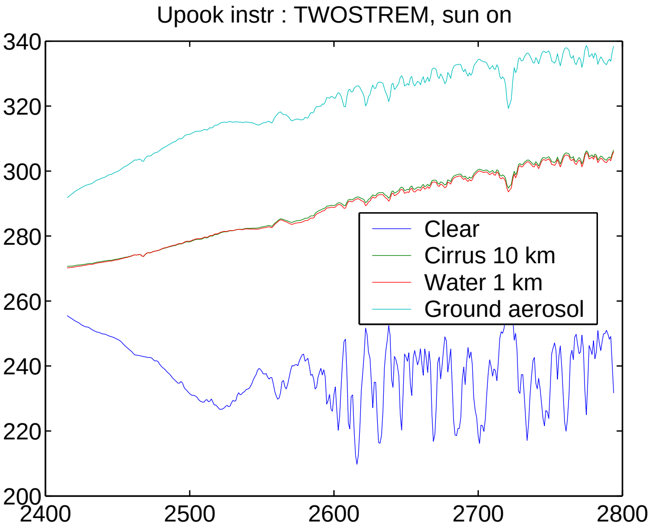


Figure 9: Scatter plots : Uplook instrument using TWOSTREAM

```
    j) raaaCloudParams(1,1,1) = cngwat
    k) raaaCloudParams(1,1,2) = cpsize
    l) raaPCloudTop(1,1)      = cprtop
    m) raaPCloudBot(1,1)      = cprbot
    n) iaCloudNumAtm(1)       = 1
    o) iaaCloudWhichAtm(1,1)  = 1

Note that if $kcarta.x$ is being used when RTP file is being read in, and
clouds specified, then everything is fine and the code proceeds. However if the
RTP file specifies a cloud when the basic $bkcarta.x$ is being used, then the
code will obviously halt.
```

14.12 *nm_spectra* (optional)

This keyword allows the code to read in externally computed spectra, for more than one gas. If $iNumNewGases$ is greater than 0, Jacobians cannot be computed, as the code has no way of perturbing the supplied spectra. The files should all be binary unformatted files, and the information in them is specified in one of the following paragraphs.

The information required in this section is of the following form

iNumNewGases =

iaNewGasID(iI) =
iaNewData(iI) =
    iaaNewChunks(iI,1) =
    iaaNewChunks(iI,2) =
    iaaNewChunks(iI,3) =
    caaaNewChunks(iI,1) =
    caaaNewChunks(iI,2) =
    caaaNewChunks(iI,3) =
...

$iaNewGasIDIiI)$ is the HITRAN gasID of the gas that the user is supplying externally computed absorption spectra for. $iaNewData(iI)$ is an integer value denoting the number of kCompressed chunks that the data will be supplied for. For each of these new chunks, $iaaNewChunks(iI, iJ)$ is an integer value telling the code which kCARTA chunk the data corresponds to, while $caaaNewChunks(iI, iJ)$ is a character array telling the code in which file the absorption spectra resides. So for example

```
    iNumNewGases = 2
    iaNewGasID(1) = 1
    iaNewData(1) = 5
        iaaNewChunks(1,1) = 405
        iaaNewChunks(1,2) = 415
        iaaNewChunks(1,3) = 425
        iaaNewChunks(1,4) = 435
        iaaNewChunks(1,5) = 445
        caaaNewChunks(1,1) = 'TABLES/scatter405'
        caaaNewChunks(1,2) = 'TABLES/scatter415'
        caaaNewChunks(1,3) = 'TABLES/scatter425'
        caaaNewChunks(1,4) = 'TABLES/scatter435'
        caaaNewChunks(1,5) = 'TABLES/scatter445'
iaNewGasID(2) = 3
iaNewData(2) = 2
        iaaNewChunks(2,1) = 1055
        iaaNewChunks(2,2) = 1080
        caaaNewChunks(2,1) = '../DATA/scatter1055'
        caaaNewChunks(2,2) = '../DATA/scatter1080'
```

tells the code not to use the kCompressed data files for 2 gases. The gases are GasID = 1, which has 5 new chunks of data (starting with wavenumbers 405, 415, 425, 435 and 445 cm), and GasID 3, which has 3 new chunks of data (starting with wavenumbers 1055 and 1080 cm).

The data in each of the files has to be in the following format

```
header

data layer 1
data layer 2
...
data layer kProfLayer
```

The header info contains the following integers on one line : $idgas, npts, nlay$. These are the gasID, number of wavenumber points (should equal kMaxPts=10000) and number of layers (which should equal kProfLayer). The next line in the header contains two reals : $sfreq, fstep$ which are the start frequency and wavenumber step respectively. These numbers should correspond to the corresponding kCompressed file the data is replacing :

idgas npts nlay

sfreq fstep

After this, the actual data should be stored in layer form, as double precision variable :

daAbsLayer1(J),J=1,kMaxPts)

daAbsLayer2(J),J=1,kMaxPts)
daAbsLayer3(J),J=1,kMaxPts)
...
daAbsLayerN(J),J=1,kMaxPts)

where as usual, layer 1 is the ground (bottommost) layer and layer kProfLayer is the highest layer. Matlab programs umbclbl_2_kcarta.m and umbclbl_2_kcarta2.m in the $UTILITY$ subdirectory allows one to easily translate spectra generated (by our MATLAB line-by-line code, $run6/7.m$) from a MATLAB to a f77 file format.

14.13 *nm_endinp* (mandatory)

Once this keyword is found, the input file is closed and a check of the keywords found is done. If one of the above 5 mandatory keywords has not been found, the program halts. If EOF is found before this keyword is read in, the program halts.

14.14 Sample template files

As mentioned at the beginning of this section, four sample template files exist in the subdirectory ../DATA/TemplateNML test1.nml could be used for fast forward model development, as it only sets up and outputs mixed paths. The rest of the files set up mixed paths, and then build atmospheres from which radiances can be computed.

15 Driver Namelist File : Important Points to Remember

Having read the previous section, we now make a list of the some requirements the user has to adhere to whilst writing the driver namelist file, in order to avoid some pitfalls. One thing noticed from above is the naming convention for variables found in the namelist file :

- variables starting with “i” are integers
- variables starting with “ia” are arrays of integers, and so need one index
- variables starting with “iaa” are arrays of arrays of integers and so need two indices
- variables starting with “r” are reals
- variables starting with “ra” are arrays of reals, and so need one index
- variables starting with “raa” are arrays of arrays of reals and so need two indices
- variables starting with “c” are characters (not used)
- variables starting with “ca” are arrays of characters (strings) index
- variables starting with “caa” are arrays of strings and so need one index
- variables starting with “caaa” are arrays of arrays of strings and so need two indices

15.1 General

- kcarta.x is command line driven. Driver namelist input file and binary output file names, if not present on the command line, are automatically defaulted to stdin and stdout. Similarly, if the program runs to completion successfully, it exits(0) to the operating system; if it catches a proble, it exits(1) to the operating system. See section on running the program.
- The gas ID's that the user should use are those specified by the HITRAN database, and are summarized in the file “gasids” in the DOC subdirectory. Note that gasID 101, 102 are the water self and foreign continuums respectively. If the kCKD parameter is set so that there is to be no continuum (kCKD = -1) then gasIDs 101,102 should NOT be included. If the kCKD parameter is set so that there is to be continuum (kCKD ¿= 0) then gasIDs 101,102 should be included.
- All angles specified by the user (eg in nm_radnce) are in degrees

- The namelist sections found in the user input file should follow the following optimum order, to prevent the program from either complaining too much or grinding to a halt, because it is too confused: PARAMS, MOLGAS, XSCGAS, PRFILE, WEIGHT, RADNCE, JACOBN, SPECTRA, SCATTR, OUTPUT, ENDINP. As mentioned above, not all the above keywords have to be used in the input namelist file

- Parameter kGasTemp (see table in nm_params) is used to control the radiating layer temperatures only for radiance (and jacobian) calculations. When uncompressing and computing the gas optical depths, the individual gas path temperatures are used.

- After reading in the driver namelist file, the program "processes" the frequencies set from nm_frqncy. If the start and stop frequencies are such that they fall in different sets of database files, as described above, the program stops.

- When reading in sections nm_molgas and, nm_xscgas the program expects to find the specified number of Gas IDs. For example, if section nm_xscgas specifies that $iX = 2$, then two Gas IDs should be specified. If the program finds less than two, it will stop. However, if it finds *more* than two, it will ignore the extra GasIDs, not complain and go on its way.

- The program can either assume a plane parallel atmosphere, or include effects on the satellite viewing angle due to the curvature of the earth. At present, no refractive effects are included in the code.

15.2 RTP file

- If kRTP = -2,-1, then the driver namelist file needs to specify, define all the needed information, such as start and stop frequencies, profile name, atmosphere, cloud etc

- If kRTP = 0, then the driver namelist file needs to specify and define all the needed information, such as start and stop frequencies, profile name, atmosphere, cloud etc. The profile is read in from an RTP file, but all the other information in the RTP file is ignored.

- If kRTP = +1, then the driver namelist file only needs to specify and define $MOLGAS, (XSCGAS) PROFLE, WEIGHT, OUTPUT$. The profile is read in from an RTP file, as are the start/stop frequencies, and the atmosphere and cloud information. Only one atmosphere, and only one cloud in this atmosphere, can be used. The $TWOSTREAM$ scattering code is automatically used if a cloud is specified.

15.3 Profiles and Weights

- The reference profiles, and the PTHFIL profiles, must have kProfLayers layers for each gas. We supply a separate package, $KLAYERS$. This can take in (almost any) point profile and change it to a path averaged kProfLayers layer profile.

- Any gas IDs read in from nm_molgas and nm_xscgas should be in ascending order.

- When reading in MOLGAS/XSCGAS, the gasID's are stored in the order they are read in. When WEIGHT is read in, the user has control over the weightings of the individual gases. In addition the mixed paths are numbered sequentially in blocks of kProfLayer mixed paths, in the order they are read in.

  So if 4 blocks are defined in WEIGHT, this means that after the program has read in the weights for all four blocks, a total of 4*kProfLayer=400 mixed paths will have been defined by the program.

  Similarly, in RADFIL, the atmospheres are ordered sequentially, 1-iNatm, in the order they are read in. However, the user can refer to any set of the declared mixed paths (from nm_weight), when building up the atmosphere

- When reading in the file specified in PRFILE, the subroutine only scans for the GAS ID's stored previously from MOLGAS/XSCGAS. The water continuum is defined by parameter kCKD in nm_params —all other relevant gases will automatically have the continuum included in their abs coeffs (eg O2,N2). If kCKD is turned on and gas IDs 101,102 are included in $MOLGAS$ then the water vapor profile will be used for these two gases

- One should **not** use the weights as a method to change the satellite viewing angle. This is now a parameter set by the user in nm_radnce . However, if the user is not going to compute radiances with kCARTA, but only use it to output transmittances at various angles, for example, the user could use the nm_weight section to achieve this.

15.4 Radiances and Jacobians

- the temperatures of the various layers in the atmospheres, built up using the mixed paths, are controlled by parameter kGasTemp (in nm_params). If kGasTemp = 1 **and** CO2 is present, then the layer temperatures are set using the CO2 path temperatures. If one of the above conditions is false, the temperatures of the layers are set using a weighted average over the gases.

- When defining output pressures, you cannot define more than kProfLayers output pressure levels per atmosphere. If necessary, all these kProfLayers pressure levels could be within the same layer.

- Suppose the user puts in a Jacobian file name on the command line, but does not have a nm_jacobn section in the input driver namelist file. The program handles this by ignoring the jacobian file name.

- The temperature Jacobian involves the cumulative weighted contributions from ALL gases. Because of memory limitations, the individual gas d/dT contributions are not stored. This means that the d/dT Jacobian is completely correct only if one atmosphere has been defined, since the d/dT contributions of the individual gases are then accurately weighted by the information found in nm_weight. If more than one atmosphere has been defined, with different weightings for the gases, then the code stores a d/dT matrix where the individual gas contributions have been weighted by 1.0, and uses this matrix for ALL atmospheres.

  As the program allocates enough memory to store the individual d/dq matrices that the user specifies, this is not a problem for the gas amount d/dq Jacobians if more than one atmosphere has been defined. The program stores a d/dq matrix weighted by 1.0; when it loops thru the atmospheres, it uses the correct gas weightings (from the mixing table) for the relevant atmosphere.

- Parameter 8 (ktempJac) tells the code whether it should compute the temperature Jacobian using only the Planck temp dependence (-2), only the optical depth/transmission dependence (option -1) or both (option 0 = default). If kTempJac $<$ 0, then the gas amount Jacobians should not be messed up, as the code sets various Planck terms to 0. Also, the setting of kTempJac has no effect on the temperature Jacobians of a upward looking instrument.

15.5 Output

kMaxPrint is a parameter in kcarta.param, that sets the maximum number of different "printing jobs" that the program can handle. There are three primary print options. Options 1 and 2 allow the user to output gas optical depths and mixed path transmittances respectively, while Option 3 allows the user to output radiances for specified atmospheres. Note the following important points

- if option 1 is found more than once, then the specific gas paths to be output are merged together to form just one list; the program considers all this as just one printing option
- if option 2 is found more than once, then the specific mixed paths to be output are merged together to form just one list; the program considers all this as just one printing option

Thus if the following were specified

iaPrinter(1) = 1
iaGPMPAtm(1) = 1
iaNp(1) = -1

iaPrinter(2) = 1
iaGPMPAtm(2) = 4
iaNp(2) = -1

iaPrinter(3) = 1
iaGPMPAtm(3) = -1
iaNp(3) = -1

first the program would decide it had to print all path spectra for gasID 1. The program would then then decide it had to print all path spectra for gas IDs 1 and 4. Finally the program would decide it had to print the path spectra for ALL gases. All of this would be considered as ONE print option.

Option 3 (radiances) is special. If a specific atmosphere is found more than once, then the output pressures are merged together to form just one list. Note that the length of the list should always be less than kProfLayer, else the program halts. If iNp < 0 (output radiance at the top of each pressure layer), then the program does not allow any more radiances to be output for that atmosphere. Thus for example, if iAtm=1 used kProfLayers pressure layers

```
iaPrinter(1) = 3
iaGPMPAtm(1) = 1
iaNp(1) = -1

iaPrinter(2) = 3
iaGPMPAtm(2) = 1
iaNp(2) = 3
raaOp(2,1) = 10.0
raaOp(2,2) = 20.0
raaOp(2,3) = 30.0
```

first the program would decide it had to output radiances at kProfLayers pressures then the program would try to add on pressures 10,20,30, but find it cannot.

16 kCARTA run-time architecture

This program computes the radiances associated with a specified atmosphere, by using the k-compressed data files. The program is run in a mode where the user supplies a driver namelist file. In addition, a gas profile file must be present (unless the user has specified that one of the regression profiles are to be used). Note that, as supplied, it is expected that the executable file is run from the RUN subdirectory; this can be changed by modifying paths in kcarta.param.

The kcarta namelist driver and output data files are specified with command line arguments. Valid invocations of kcarta have one of four following forms:

- kcarta driver outfile jacfile
- kcarta driver outfile
- kcarta driver
- kcarta

If the driver namelist file is not specified, or is "–", then the driver namelist file is read from standard input. If the output or jacobian files are not specified, or are "–", then the respective outputs or jacobians are written to standard output. (Note that it is possible, but probably not a good idea, to mix regular and jacobian output data.)

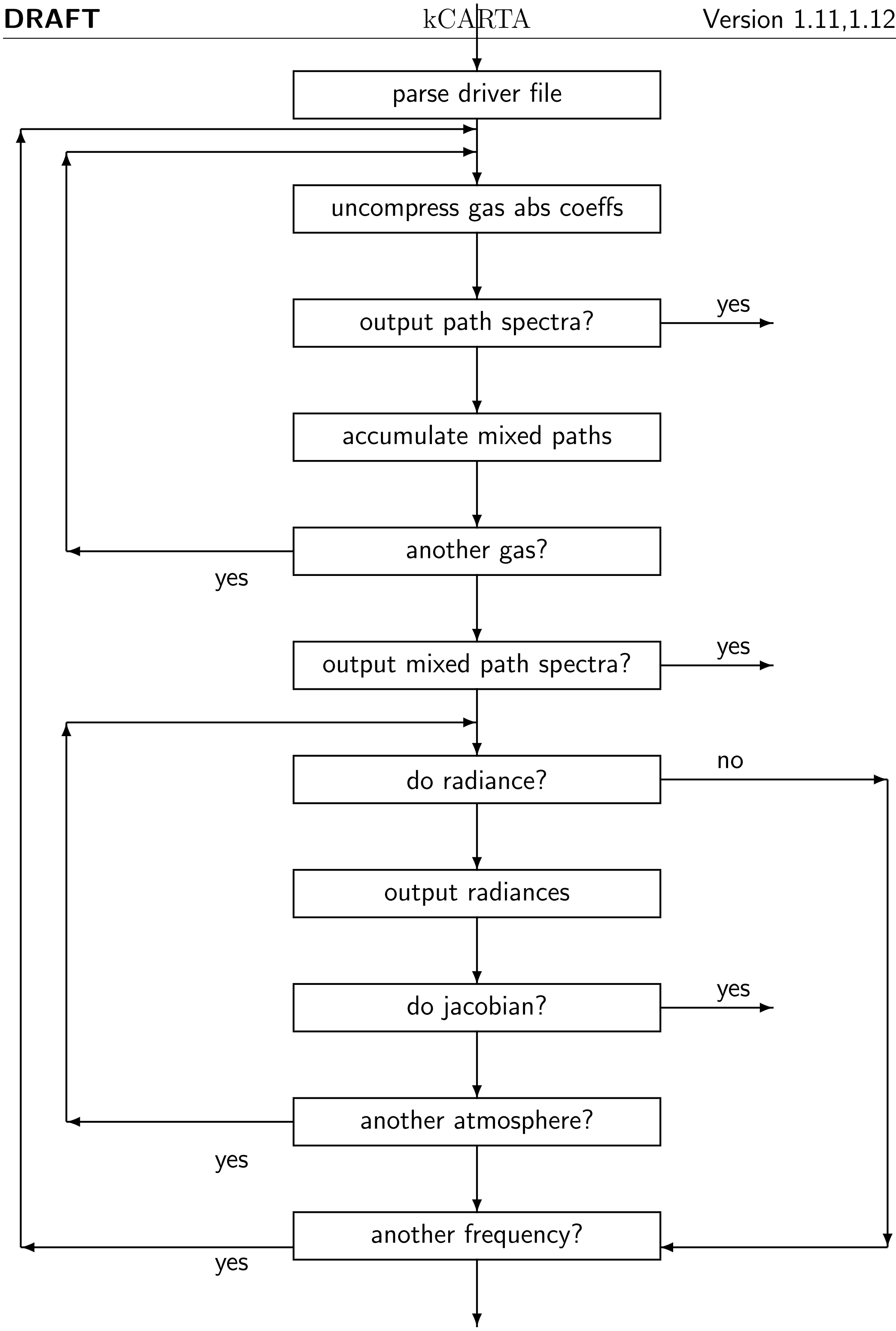
Fatal errors are written to "standard error", while a list of informative messages are saved in the file specified by caLogFile (from the nm_output section. This message filename can be set to /dev/null, if messages are not wanted.

After the driver namelist file is parsed, the program then proceeds with the uncompressions, radiance calculations, outputting results as necessary. Again, if the program finds an error while doing these computations, it politely prints out a message and halts.

WARNING!!!! All the character strings in the driver namelist file (e.g., the comment in nm_output) MUST be enclosed in quotes

The first section of code executed is to ensure that the user set parameters in kcarta.param are correct and consistent—if not, the program halts. After the start/stop frequencies are read in, the program verifies that they lie between the Min and Max allowed frequencies. They are set to the min and max of the associated 10000 point blocks. In addition, 0.0025 (or 0.001 or whatever necessary wavenumber spacing) is subtracted from the stop frequency, so that an extra set of calculations does not have to be done for the endpoint.

If the input file was successfully read in, the binary output file is opened (if this file exists, the program halts). As these summaries are saved, the program performs some consistency checks, and halts if it finds weird stuff (e.g. asking information from atmosphere 2 to be output, even if information for only one atmosphere has been read in). The main portion of the program then runs, where the loop structure is as follows :


parse driver file
uncompress gas abs coeffs
output path spectra?
yes
accumulate mixed paths
another gas?
yes
output mixed path spectra?
yes
do radiance?
no
output radiances
do jacobian?
yes
another atmosphere?
yes
another frequency?
yes

The contributions to the absorption coefficient is computed gas by gas

if $1 \leq$ iGasID ≤ 28 we have compressed database
if $1 =$ iGasID can include water continuum
if $51 \leq$ iGasID ≤ 63 we need XSEC

As the gas profiles are read in, they are checked to see if the same temperature profile is found for each gas. If not, a warning message is flashed. If Jacobians need to be computed for the atmosphere, they are done so after the radiance for the atmosphere is calculated.

All through the running of kCARTA, warning messages and information messages are output by kCARTA, so that if the program does have to stop, the user will know where and why (this is not a politically correct way of avoiding saying the nasty word "crash," as in our experience kCARTA never crashes; instead, it could get perplexed by some of the users directions, missing files and so on and then politely halts).

17 Binary Output Files from a **kCARTA** run

This section describes the FORTRAN output binary file that results from a kCARTA run. This will allow a user to write his/her own reader. It might behoove the user to refer to our supplied $FORTRAN$ readers readkcarta.f, readjacob.f, readfkux.f and/or the $MATLAB$ readers readkcstd.m, readkcjac.m, readkcflux.m.

As a further hint, if the user still has problems writing reader code after perusing this section, the user should refer to file s_writefile.f in our SRC distribution. The latter half of subroutine prepareoutput creates the header information, while subroutine wrtout outputs the actual binary data.

One file is always output - a binary file whose name is defined in the command line options when starting kcarta.x. The program always makes sure the status of this file is "new" i.e. it does not overwrite an existing data file.

Depending on the value of $kLongOrShort$, one of three types of output files can be written by kCARTA.

17.1 Binary Output Files :$kLongORShort = 0$

The beginning of this data file contains a very short summary header of the driver namelist file that was read in, followed immediately by data output by kCARTA, in blocks of 10000 points. A text version of the input namelist file is saved to file specified by $caLogFile$, so the user can easily view what kCARTA read in and how it did some set-ups. At present, this option cannot be used if fluxes and/or jacobians are to be output.

The output binary file is in the following binary format

MAIN HEADER
DATA

17.1.1 The MAIN HEADER

This part of the file summarizes kCARTA parameters, as well as how many outputs to expect in the data file.

char*80	comment	set by program, giving version number
integer	kProfLayer	
integer	kMaxUserSe	max number of params in nm_params
real array	(raParams(il),il=1,kMaxUserSet)	
char*80	comment	entered in nm_output section
2 reals	FrMin,FrMax	set in nm_frqncy
2 integers	iSetMin,iSetMax	kcomp block numbers corresponding to above freqs
integer	kLongOrShort	whether long or short header style used
real	frequency step size = dv	
real	chunk wavenumber spread	10000*dv
integer	iTotal = total num outputs per 1000 pt chunk	

17.1.2 The DATA

The most relevant information from the above header is contained in the five parameters $iSetMin, iSetMax, iTotal, FrMin, dv$

The total number of 10000 point kCARTA chunks processed is

$$iChunk = iSetMax - iSetMin + 1$$

For each of these chunks, there are $iTotal$ outputs of 10000 points each, corresponding to each of the optical depths, mixed paths and radiances output by kCARTA. The corresponding wavenumbers are found by

$$ii = (1 : 10000 * iChunks) - 1$$

$$wnums(ii) = FrMin + dv * ii;$$

The data can be sequentially read in as

```
ii=(1:10000*iChunks) - 1;
wnums = fmin + dv*ii;
data = zeros(10000*iChunks,iTotal);
for ii = 1 : iChunks
  fprintf(1,'  reading in chunk number %4i .... \n',ii);
  index = (1:10000) + (ii-1)*10000;
  for jj = 1 : iTotal
    flen  = fread(fin, 1, 'integer*4');
    ra    = fread(fin, 10000, 'real*4');
    flen  = fread(fin, 1, 'integer*4');
    data(index,jj) = ra;
    end
  end
```

17.2 Binary Output Files :$kLongORShort = \pm 1$

The beginning of this data file contains a summary main header of the driver namelist file that was read in, as well as any path spectra/mixed path spectra/radiances output by kCARTA. If the user has set kLongShort to +1 (in nm_params), a text version of this is saved to the $caLogFile$ file, so the user can easily view what kCARTA read in and how it did some set-ups. If kCARTA has been asked to compute Jacobians, another binary file is also opened for output (the program checks to ensure that this file is also "new"). This file also contains a short header section, where the gases whose gas amount Jacobians are to be output, have their profiles summarized. After this header section, comes the actual Jacobian output.

Depending on the value of iOutputOption set in the nm_output section, the path/ mixed path abs spectra are output for the specified paths, or the radiance spectra, are output for the relevant pressures. The output is in blocks of 10000 points.

The output binary file is in the following binary format

MAIN HEADER
DATA

17.2.1 The MAIN HEADER

This part of the file essentially summarizes the input driver namelist file that was parsed in, so that the user can easily find out how the program interpreted the input file (especially as regards the possible merging of the path/mixed path/radiance outputs).

The first section contains general information for the entire run : a (kCARTA) comment, basically giving program version number, followed by kProfLayer. Next is the integer value of kMaxSet (which tells the program how many parameters the user can default or set in *nm_params*), followed by the (real) values of these *nm_params* parameters—parameter 1 to parameter kMaxSet.

raParams(1)=real(kLayer2Sp)
raParams(2)=real(kCKD)
raParams(3)=real(kGasTemp)
raParams(4)=real(kLongOrShort)
raParams(5)=real(kJacobOutput)
raParams(6)=real(kFlux)
raParams(7)=kSurfTemp
raParams(8)=kTempJac

The user defined comment for the specific run then follows, after which are the frequency endpts, the kCompressed strart/stop file numbers. Finally the parameter kLongOrShort (whether or not the complete driver namelist file info is summarized in the header - gas profiles etc) is explicitly given (even though it is buried inside raParams)

The next three sections then consists of the path info, mixed path info and atmosphere info (NOTE : if kLongOrShort = -1, then the short header option is used, and so information that is repeated from the profile file, and the mixtable file, is not included in the header). All the loops shown are FORTRAN implied do loops. Finally, there is a summary section that is output. This tells the reader how many sets of kCompressed files were processed, and also gives a succinct summary of how many paths/mixed paths/radiances to expect for each kCompressed set that is processed.

————-GENERAL INFO————-

char*80	comment	set by program, giving version number
integer	kProfLayer	
integer	kMaxUserSe	max number of params in *nm_params*
real array	(raParams(iI),iI=1,kMaxUserSet)	
char*80	comment	entered in *nm_output* section
reals	Frequency min, max	set in *nm_frqncy*
integers	iSetMin,iSetMax	kcomp block numbers corresponding to above freqs

integer	kLongOrShort	whether long or short header style used
integer array	(iaParams(iI),iI=1,6 =M100mb,...,MthickLayer)	
real array	(raPressLevels(iI),iI=1,kProfLayer+1)	

————-SINGLE PATH INFO————-

	if iLongOrShort > 0	for each gas, loop over kProfLayers layers,
integer	iNumPaths	iNumGases*kProfLayer
i i r r	do loop over gas	for each of kProfLayers layers, give
	iPath iGasID rTemp rAmt	path num, GASID, temp, amt
	end	
integer	iNumPathOut	number of paths to be output
	if iNumPathOut > 0	
integer array	(iaPathsOut(iI),iI=1,iNumPathsOut)	list of gas paths to be output

————-MIXED PATH INFO————-

integer	iNpmix	number of mixed paths
	if iLongOrShort > 0	
integer	iMixFileLines	num of lines in mixed table sets
	if iNpMix > 0	
char*130	caMix(iI),iI=1,iMixFileLines	copied straight from WEIGHT
real array	raMixVertTemp(iI),iI=1,iNpmix	mixed path temps
	if iNpMix > 0	
integer	iNumMPOut	number of mix paths to be output
	if iNumMPOut > 0	
integer array	(iaMixPathsOut(iI),iI=1,iNumMPOut)	list of mixed paths to be output

————-ATMOSPHERE INFO————-

integer	iNatm	Number of Atmospheres specified in nm_radnce
integer	kEmsRegions	Number of different emissivity regions that can be set
	do loop over each atmosphere	
integers	iI,iNumLayer	atm number, number of mixed mixed path layers in atmosphere
integer array	(iaRadLayers(iJ),iJ=1,iNumLayer)	list of mixed

		paths (layers) in the atmosphere
r r r r	T_{sp}, T_{surf},SVA,SH	space temp, surface temp satellite view angle, satellite height
i r r i r i	iSolar,rSolarAngle,rSolarRefl, iThermal,rThermalAngle, iThermalJacob	details for solar and backgnd thermal
integer	iEms	Number of emissivity data points
	for each emissivity data point	
reals	frequency emissivity	
integer	iAtmOut	number of radiating layers to be output
	if iAtmOut $>$ 0	
integer array	(iaOutLayer(iJ),iJ=1,iAtmOut)	list of mixed paths where to output radiance
real array	(raOutPress(iJ),iJ=1,iAtmOut)	list of pressures where to output radiance

————-SUMMARY INFO————-

i i i	iTotal iOutTypes	number of kCompressed files and number of output options
integer array	(iaOutNum(iJ),iJ=1,iOutTypes)	list of number of outputs for each OutputOption

iTotal is the total number of sets of kCompressed files that are processed by the particular run. For example, if the start/stop frequencies are 805.0 and 905.0 respectively, this means that 4 kCompressed chunks are processed : 805-830,830-855,855-880 and 880-905.

iOutTypes is the total number of print options specified. For example, if the program is to output 3 path spectra, 1 radiance for atmosphere number 1 and 5 radiances for atmosphere number 2, there are a total of 3 print options (iOutTypes = 3). The total number of outputs to be expected for each of this print option is set in iaOutNum. For our example, this means iaOutNum(1)=3,iaOutNum(2)=1 and iaOutNum(3)=5.

17.2.2 The path spectra/mixed path spectra/radiance DATA

For each set of data files that have been uncompressed, the entire relevant set of data is output to the data file. No further subdivision is possible. For example, if the current wavenumber block is 855-880 cm^{-1}, then the program will output all 10000 points for each path/mixed path/radiance/ jacobian it has been asked to output.

Each logical set of data is preceded by its own header. For the standard kCARTA output, the logical sets are divided into paths, mixed paths and the separate atmosphere. For the Jacobian output, for each of the atmospheres, the logical sets are divided into the separate gas jacobians, the temperature jacobian, the weighting functions and the surface jacobians.

The header is in the following format :

```
i i i     iMainType,iSubMainType,iNumberOut
i r r r   kMaxPts,rFrLow,rFrHigh,rDelta
```

- iMaintype = 1,2,3
  This reflects what OutputOption is being processed. 1 means paths, 2 means mixed paths and 3 means radiances

- iSubMainType
  If paths or mixed paths are being output (iMainType=1,2) then this is set to kLayer2Sp (so we easily know if these are transmittances, optical depths ...)
  If radiances are being output (iMainType=3) then this is set to the current atmosphere number

- iNumberOut
  This is the number of outputs to expect in this logical set of data. For example, if kProfLayers mixed paths are to be output, this number will be kProfLayers

- kMaxPts
  THis tells how many points are in each output; this is 10000

- rFrLow is the lower frequency bound for this particular kCompressed set

- rFrHigh is the upper frequency bound for this particular kCompressed set

- rDelta is the point spacing for this particular kCompressed set

Having written out the header, the program then proceeds to output the required data.

If gas abs spectra are being output, then the following is output for the required iNumPathOut gas paths

real array	(raAbs(iFr),iFr=1,10000)
real array	...
real array	(raAbs(iFr),iFr=1,10000)

If mixed path spectra are being output, then the following is output for the required iMixOut mixed paths

real array	(raAbs(iFr),iFr=1,10000)
real array	...
real array	(raAbs(iFr),iFr=1,10000)

If radiances are being output, then the following is output for each of the required atmospheres, each needing iOutRad radiances

real array	(raAbs(iFr),iFr=1,10000)
real array	...
real array	(raAbs(iFr),iFr=1,10000)

Example 1: Suppose the frequency endpts specified by the user are 855 to 905 cm^{-1} and kCARTA is expected to output four mixed path spectra, and one radiance for each of two atmospheres. Assume kLayerToSpace=-1.

Thus the program has to process two 10000 point chunks, the first from 855 to 880 cm^{-1}, and the second from 880 to 905 cm^{-1}.

description	variables	values
	iMainType,iSubMainType,iNumberOut	2 -1 4
	kMaxPts,rFrLow,rFrHigh,rDelta	10000 855.0 879.9975 0.0025

mixed path 1		(raMix1(iFr),iFr=1,kMaxPts)
mixed path 2		(raMix1(iFr),iFr=1,kMaxPts)
mixed path 3		(raMix3(iFr),iFr=1,kMaxPts)
mixed path 4		(raMix4(iFr),iFr=1,kMaxPts)
	iMainType,iSubMainType,iNumberOut	3 1 1
	kMaxPts,rFrLow,rFrHigh,rDelta	10000 855.0 879.9975 0.0025
atm1,rad1		(raRad1_1(iFr),iFr=1,kMaxPts)
	iMainType,iSubMainType,iNumberOut	3 2 1
	kMaxPts,rFrLow,rFrHigh,rDelta	10000 855.0 879.9975 0.0025
atm2,rad1		(raRad2_1(iFr),iFr=1,kMaxPts)
	iMainType,iSubMainType,iNumberOut	2 -1 4
	kMaxPts,rFrLow,rFrHigh,rDelta	10000 880.0 904.9975 0.0025
mixed path 1		(raMix1(iFr),iFr=1,kMaxPts)
mixed path 2		(raMix2(iFr),iFr=1,kMaxPts)
mixed path 3		(raMix3(iFr),iFr=1,kMaxPts)
mixed path 4		(raMix4(iFr),iFr=1,kMaxPts)
	iMainType,iSubMainType,iNumberOut	3 1 1
	kMaxPts,rFrLow,rFrHigh,rDelta	10000 880.0 904.9975 0.0025
atm1,rad1		(raRad1_1(iFr),iFr=1,kMaxPts)
	iMainType,iSubMainType,iNumberOut	3 2 1
	kMaxPts,rFrLow,rFrHigh,rDelta	10000 880.0 904.9975 0.0025
atm2,rad1		(raRad2_1(iFr),iFr=1,kMaxPts)

17.3 The Jacobian file

If kJacobian == 1 then an auxiliary data file (could have 'JAC' appended to it) will be produced. It has the same header info as the main output file, and is in the format General Header, Special Header, Data. As usual, the output Jacobian file is in the following binary format
MAIN HEADER
DATA

————-GENERAL INFO————-

char*80	comment	entered in nm_output section
integer	kProfLayer	
reals	Frequency min, max	set in nm_frqncy

integers	iSetMin, iSetMax	kcomp block numbers corresponding to above freqs

————SPECIAL HEADER ————-

integer	iNumDQ	number of gases we do d/dq for
integer	iAtmJac	number of atmospheres we do rad, jacs for
integer array	iaLayerJac(iI),iI=1,iAtmJac	
integer	kProfLayer	
reals	Frequency min, max	set in nm_frqncy
integers	kcomp min, max	kcomp block numbers corresponding to above freqs
i r r	do loop over iNumDQ gases iGasID rTemp rAmt	stating GASID, temperature, amount

—————SUMMARY INFO—————-

i i i	iTotal iNatmJac iNumDQ	number of kCompressed files, number of atmospheres and number of gases whose d/dq computed
integer array	(iaLayerJac(iJ),iJ=1,iNatmJac)	list of number of layers in the atmospheres whose Jacobians computed

iTotal is the total number of sets of kCompressed files that are processed by the particular run. For example, if the start/stop frequencies are 805.0 and 905.0 respectively, this means that 4 kCompressed chunks are processed : 805-830,830-855,855-880 and 880-905.

iNatmJac is the total number of print options that ask for radiances to be output, so that we know this is the number of forward models that will be iterated. iNumDQ, as stated above, is the number of gases whose Jacobians have been specified in nm_jacobn. For each of the iNatmJac atmospheres, iaLayerJac contains the number of mixed paths in the atmosphere. For example, if the program is to output radiances for 2 atmospheres, one that has 90 mixedpaths and the other 43 mixedpaths. Then iNatmJac=2, and iaLayerJac(1)=90 and iaLayerJac(2)=43.

17.4 The Jacobian DATA

——— DATA ————————

As above, the DATA consists of 10000 point blocks, repeated in the format described above (header + data) for each of the Jacobians. The jacobians have been calculated in the following order (for each atmosphere)

d/dq gas 1 layer atm(1,lower) ... layer atm(1,upper)
d/dq gas 2 layer atm(1,lower) ... layer atm(1,upper)
... ...
d/dq gas N layer atm(1,lower) ... layer atm(1,upper)
d/dT layer atm(1,lower) ... layer atm(1,upper)
WGT FCN layer atm(1,lower) ... layer atm(1,upper)
SURFACE 1 2 3 4

where the 4 surface jacobians are d/d(Surface Temp),d/d(Surface Emissivity), d(Thermal BackGnd)/d(Surface Emissivity) and d(Solar)/d(Surface Emissivity)

As usual, the header is in the following format :

i i i	iMainType,iSubMainType,iNumberOut
i r r r	kMaxPts,rFrLow,rFrHigh,rDelta

- iMaintype
  This tells which of the atmospheres is being processed

- iSubMainType
  This reflects what is being processed.
  1,2,3 ... are the GasID whose jacobian is being computed and output
  0 means this is the temperature jacobian
  -10 means this is the weighting functions
  -20 means this is the set of surface jacobians

- iNumberOut
  This is the number of outputs to expect in this logical set of data. For exam-

ple, if kProfLayers mixed paths are to be output, this number will be kProfLayers

- kMaxPts
  THis tells how many points are in each output; this is 10000

- rFrLow is the lower frequency bound for this particular kCompressed set

- rFrHigh is the upper frequency bound for this particular kCompressed set

- rDelta is the point spacing for this particular kCompressed set

Hence if there are 5 gases, and atmosphere 1 has 10 layers, there will be [(5+2)*10] + 4 = 74 blobs of data for that atmosphere, as follows.

- For each of the 5 gases, a set of gas amount Jacobians are done; in addition, a set of temperature Jacobians and a set of weighting functions are computed. This takes up (5+1+1)*10 = 70 blobs of data
- In addition, there are four surface parameter Jacobians output (surface temp, surface emissivity,thermal background wrt surface emissivity and solar contribution wrt surface emissivity).

Example 1: Suppose the frequency endpts specified by the user are 855 to 880 cm^{-1} and kCARTA is expected to output four mixed path spectra, and one radiance for each of two atmospheres. Assume atmosphere 1 has 10 layers, while the other has 15 layers, and the user wants the gas jacobian for gasID 7 to be output.

Thus the program has to process one 10000 point chunks, from 855 to 880 cm^{-1}.

description	variables	values
Atm 1		
gas 1	iMainType,iSubMainType,iNumberOut	1 7 10
	kMaxPts,rFrLow,rFrHigh,rDelta	10000 855.0 879.9975 0.0025
		(raGas7(iFr),iFr=1,kMaxPts)
		...
		(raGas7(iFr),iFr=1,kMaxPts)

temperature	iMainType,iSubMainType,iNumberOut	1 0 10
	kMaxPts,rFrLow,rFrHigh,rDelta	10000 855.0 879.9975 0.0025
		(raTempr(iFr),iFr=1,kMaxPts)
		...
		(raTempr(iFr),iFr=1,kMaxPts)
wgt fcn	iMainType,iSubMainType,iNumberOut	1 -10 10
	kMaxPts,rFrLow,rFrHigh,rDelta	10000 855.0 879.9975 0.0025
		(raWgt(iFr),iFr=1,kMaxPts)
		...
		(raWgt(iFr),iFr=1,kMaxPts)
surface param	iMainType,iSubMainType,iNumberOut	1 -20 4
	kMaxPts,rFrLow,rFrHigh,rDelta	10000 855.0 879.9975 0.0025
		(raSurf(iFr),iFr=1,kMaxPts)
		...
		(raSurf(iFr),iFr=1,kMaxPts)
Atm 2		
gas 1	iMainType,iSubMainType,iNumberOut	2 7 15
	kMaxPts,rFrLow,rFrHigh,rDelta	10000 880.0 904.9975 0.0025
		(raGas7(iFr),iFr=1,kMaxPts)
		...
		(raGas7(iFr),iFr=1,kMaxPts)
temperature	iMainType,iSubMainType,iNumberOut	2 0 15
	kMaxPts,rFrLow,rFrHigh,rDelta	10000 880.0 904.9975 0.0025
		(raTempr(iFr),iFr=1,kMaxPts)
		...
		(raTempr(iFr),iFr=1,kMaxPts)
wgt fcn	iMainType,iSubMainType,iNumberOut	2 -10 15
	kMaxPts,rFrLow,rFrHigh,rDelta	10000 880.0 904.9975 0.0025
		(raWgt(iFr),iFr=1,kMaxPts)
		...
		(raWgt(iFr),iFr=1,kMaxPts)
surface param	iMainType,iSubMainType,iNumberOut	2 -20 4
	kMaxPts,rFrLow,rFrHigh,rDelta	10000 880.0 904.9975 0.0025
		(raSurf(iFr),iFr=1,kMaxPts)
		...
		(raSurf(iFr),iFr=1,kMaxPts)

Notice how for each atmospher, the gas jacobian, temperature jacobian and weighting function have the required number of layers, while the surface parameter jacobian always has 4 items that are output.

17.5 The Flux file

If kFlux = 1,2,3,4,5 then an auxiliary data file, with 'FLUX' or 'OLR' appended to it, will be produced. It has the same header info as the Jacobian file, and is in the format General Header, Special Header, Data. As usual, the output Flux file is in the following binary format
MAIN HEADER
DATA

————-GENERAL INFO————-

char*80	comment	entered in *nm_output* section
integer	kProfLayer	
reals	Frequency min, max	set in *nm_frqncy*
integers	iSetMin, iSetMax	kcomp block numbers corresponding to above freqs

———–SPECIAL HEADER ————-

integer	iFluxDiff	number of flux diffs == 1
integer	iAtmRad	number of atmospheres we do rad for
integer array	iaLayerRad(iI),iI=1,iAtmRad	
integer	kProfLayer	
reals	Frequency min, max	set in *nm_frqncy*
integers	kcomp min, max	kcomp block numbers corresponding to above freqs

————-SUMMARY INFO————-

i i i	iTotal iNatmJac iNumFlux	number of kCompressed files, number of atmospheres and number of flux diffs (1) computed

integer array	(iaLayerRad(iJ),iJ=1,iNatmRad)	list of number of layers in the atmospheres where radiances computed

iTotal is the total number of sets of kCompressed files that are processed by the particular run. For example, if the start/stop frequencies are 805.0 and 905.0 respectively, this means that 4 kCompressed chunks are processed : 805-830,830-855,855-880 and 880-905.

iNatmRad is the total number of print options that ask for radiances to be output, so that we know this is the number of forward models that will be iterated. iNumFluxes, is the number of fluxes (== 2, for upward and downward) computed. For each of the iNatmRad atmospheres, iaLayerRad contains the number of mixed paths in the atmosphere. For example, if the program is to output radiances for 2 atmospheres, one that has 90 mixedpaths and the other 43 mixedpaths. Then iNatmRad=2, and iaLayerRad(1)=90 and iaLayerRad(2)=43.

17.6 The Flux DATA

——— DATA ———————

As above, the DATA consists of 10000 point blocks, repeated in the format described above (header + data) for each flux set. The fluxes have been calculated in the following order (for each atmosphere)

net=upward-downward flux layer atm(1,lower) ... layer atm(1,upper)

As usual, the header is in the following format :

i i i	iMainType,iSubMainType,iNumberOut
i r r r	kMaxPts,rFrLow,rFrHigh,rDelta

- iMaintype
  This tells which of the atmospheres is being processed

- iSubMainType
  This reflects what is being processed.
  +1 means this is net=(upward-downward) flux

- iNumberOut
  This is the number of outputs to expect in this logical set of data. For example, if kProfLayers mixed paths are to be output, this number will be kProfLayers

- kMaxPts
  THis tells how many points are in each output; this is 10000

- rFrLow is the lower frequency bound for this particular kCompressed set

- rFrHigh is the upper frequency bound for this particular kCompressed set

- rDelta is the point spacing for this particular kCompressed set

Hence if atmosphere 1 has 10 layers, there will be 10 blobs of data for that atmosphere, as follows.

- For each of the 10 layers, there is a net=(upward-downward) flux

Example 1: Suppose the frequency endpts specified by the user are 855 to 880 cm^{-1} and kCARTA is expected to output four mixed path spectra, and one radiance for each of two atmospheres. Assume atmosphere 1 has 10 layers, while the other has 15 layers.

Thus the program has to process one 10000 point chunks, from 855 to 880 cm^{-1}.

description	variables	values
Atm 1		
up-down	iMainType,iSubMainType,iNumberOut	1 1 10
	kMaxPts,rFrLow,rFrHigh,rDelta	10000 855.0 879.9975 0.0025
		(netflux(iFr),iFr=1,kMaxPts)
		...
		(netflux(iFr),iFr=1,kMaxPts)
Atm 2		

up-down	iMainType,iSubMainType,iNumberOut	2 1 15
	kMaxPts,rFrLow,rFrHigh,rDelta	10000 855.0 879.9975 0.0025
		(netflux(iFr),iFr=1,kMaxPts)
		...
		(netflux(iFr),iFr=1,kMaxPts)

17.7 Reading the binary output file from a **kCARTA** run

For those fortunate MATLAB users, a set of read*.m files have been written so that the binary file can be read in. The main file names are readkcarta.m and readkcstd.m.

17.7.1 readkcstd.m

This is the more powerful $MATLAB$ reader. However it reads in the entire kCARTA output file, and so it try to use too much memory. To overcome this problem, it does allow the user to store the results in file "dfile", instead of in memory.

```
[data, wnums] = readkcstd(kfile, dfile)
```

The wavenumbers are stored in "wnums" while the entire data set is stored in "data"

17.7.2 readkcarta.m

readkcarta.m can be used on computers which have memory limitations, as it allow the user to choose specific sets of data to be read in. At present, due to these possible memory limitations, there are two options that can be run after the MAIN HEADER has been read in:

1. (1) read in only one COMPLETE 10000 point block of data. In other words, save in matrix raaData, all the path/mixed path absorption spectra AND the radiances for the atmospheres/layers, for the chosen k-comp file

2. (2) for ONE of the stored paths/mixed paths/radiating layers, store the ENTIRE information, from freq_min to freq_max, in array raEntire

The program first reads in the path information (gas profiles and temperatures, gas paths to output), the mixed path information (weights and mixed path temperatures, mixed paths to be output) and then the atmosphere information (atmosphere definitions, such as start and stop pressures, satellite viewing angle and pressures at which to output radiances).

Suppose there are *iPathOut* paths to be output, *iMixPathOut* mixed paths to be output and *iRadianceOut* radiances to be output, giving a total of *iTotal* = *iPathOut* + *iMixPathOut* + *iRadianceOut*. If the user chooses (1), the program asks *which* of the 10000 point blocks the user wants to read in (...,805,830, ...). Assuming the user chooses a block that is within the frequency range set in nm_frqncy, then all *iTotal* paths, mixed paths and radiances for this block are read in and saved in matrix *raaData*, with the corresponding frequency being stored in array *raFreq*.

If the user chooses (2), the program asks *which* of the *iTotal* spectra/radiances in (1 ... iTotal) the user wishes to read in. For this choice, the data spanning the entire frequency range is read in and saved in array *raEntire*, with the corresponding frequency being stored in array *raFreq*.

17.7.3 readkcarta.f

For non-MATLAB users, a similar FORTRAN file has been written, readkcarta.f. This reads in a specified binary file, and outputs one of two possibilities. The first is a binary file that has ALL the output data in it (without the patm/mixed path/ atmosphere header information). The data for any path/mixed path/radiance is concatenated together so that the user does not have to "sort" through. For example, if the start/stop frequencies spanned 50 cm^{-1} (making 10000*2=20000 points), then the data relating to each output option would be output in one 20000 point array.

Suppose there are 20000 data points (eg the start/stop freqs were 605-655 cm^{-1}), and that 3 absorption spectra, 2 mixed path spectra and 1 radiance are output per each 10000 point chunk (making a total of 6 data blobs output each time). The format of the resulting file is

(raFreq(i) i=1,20000)
((raaData(i,j) i=1,20000),j=1,6)

Another possibility is that readkcarta.x can also be run so that it produces a text file containing two columns : the frequency points and *one* of the paths/ mixed paths/ radiances contained in the binary file produced by kCARTA. The sequence of user inputs for this type of run is the same as that described a few paragraphs above for choice (2) in readkcarta.m, with the arrays raFreq, raEntire being produced.

17.7.4 readjacob.f,readjacob.m and readkcjac.m

The Jacobian files can similarly be read in, using readjacob.m, readkcjac.m for a MATLAB afficiando, or readjacob.f for a FORTRAN diehard. The MATLAB Jacobian readers proceed in exactly the same fashion as the the MATLAB kCARTA readers described above, while readjacob.x will produce a file of the same binary format as readkcarta.x above.

Note that the finite difference jacobians (column gas amounts) is dumped out in the same format as a "basic" kCARTA file, and so can be read in using readKcBasic.m or readKcBasic.f (described above)

However, the main (layers) jacobian data file is dumped out in a slightly different format, and so needs to be read in differently. Suppose there was only one atmosphere in the kCARTA run, and that it had 80 layers, with the user asking for 3 gas Jacobians. Then for each 10000 point chunk, there would be (3*80) gas jacobians, 80 temperature jacobians, 80 weighting functions and 4 surface parameter derivatives, making a total of 404 each time. So if readjacob.x was being used, the format of the resulting file is

(raFreq(i) i=1,20000)
((raaData(i,j) i=1,20000),j=1,404)

Ditto for a flux file; readflux.m readflux.f would allow the user to read in the flux data.

18 Additional Programs and Readme Files

There are a number of useful programs that are supplied in UTILITY :

Table 34: Utility programs.

compdatabase97.f	creates a summary of the available k-compressed files, by running script comp.sc. The results stored in comp97.param (xsec.param is a similar summary for the data in the XSEC database, and is also required by kcarta.x)
readkcarta.f	allows one to read in portions of the output binary file, and save the results in a simpler binary or text file
readjacob.f	allows one to read in portions of the output jacobian file, and save the results in a simpler binary file

readflux.f	allows one to read in portions of the output flux file, and save the results in a simpler binary file
makeinp.f	using a template input file, this can be used as part of a script to automate running kcarta.x. This file replaces starts/stop frequencies, and output data file names, as well as surface temperatures and input profile names
makeprofile.f	using a set of test profiles, this file, along with makeinp.f, can be used as part of a script to automate testing kcarta.x

readkcarta.f, readjacob.f and readflux.f are FORTRAN files that directly read the output from kcarta.x and save the results to a data file that can be more easi;ly read in, as it has been stripped of all the header information duplicating the driver namelist file and profile.

readkcstd.m, readkcjac.m and readkcflux.m are MATLAB files that directly read the output from kcarta.x. This would be the$MATLAB$ readers that we would suggest to be used.

readkcarta.m, readjacob.m and readflux.m are MATLAB files that directly read the output from kcarta.x. Since they allow the user to choose what part of the file to read in, they could be useful to users whose computers have limited amounts of memory; however, we would still recommend using the afore mentioned set of readers.

rdairs.m is a MATLAB reader that can load in a file produced by running readkcarta.f, readflux.f or readjacob.f (it assumes that the files that are written out by the read*.f programs have "CON" appended to the original filename).

For people familiar with the older versions of kCARTA, which are driven by a GENLN2 style file, we have provided a "translator" code that takes the old input file and translates it to a namelist file. The set of files that do this are in SRC/MAKENAMELIST; by typing "make" when in this subdirectory, executable "makelist.x" is produced and stored in /BIN. To use this code to translate file "old_driver_file" to "new_namelist.nm" by typing

```
makelist.x old_driver_file  new_namelist.nml
```

We also supply KLAYERS, which contains a set of files that take in a point profile supplied by a user, and changes it to to kProfLayers layer path averaged profile. The

user is referred to the documentation in that subdirectory for instructions on how to run the necessary programs, in particular, the files description.txt and sci.txt.
To change a radiosonde point profile to a $KLAYERS$ file, go to subdirectory "/KCARTA/SCRIPTS" Assuming everything is OK (eg paths are set correctly and so on), run your sonde profile through $klayers.x$ by typing:

```
makeprof.sc  sonde_profile_in rtp_layers_out
```

where "sonde_profile_in" contains the input (levels) point profile and "rtp_layers_out" is the resulting KLAYERS layer averaged RTP profile. This is the profile that can be used by kCARTA by specifying its name in the namelist driver file.

In addition to this document, other helpful files exist. In the /DOC subdirectory, the following files can be found

- JQSRT_kCompress.pdf: Paper describing the compressed database, in pdf format
- JQSRT_kCompress.ps: Paper describing the compressed database, in ps format
- netcdfinfo: for the brave NetCDF user, how to download and compile updated versions
- readme.tex: how to set up and run the kCARTA distribution
- gasids: list of HITRAN gas IDs

The KLAYERS/Doc subdirectory has the following text files

- junkreadme.txt: how to set up and compile the source code
- description.txt: description of the source code files
- sci.txt: description of the physics/math used in programming up KLAYERS

The UTILITY subdirectory also has a file, Readme.txt, that gives a brief description of the auxiliary files contained there.

19 Science : Radiance and Jacobian calculations

To allow the user flexibility in creating an atmosphere, kCARTA allows the user to define the start/stop pressure boundaries arbitrarily. In other words, the user does not have to worry about these pressures being at the AIRS pressure layers themselves - they can be in the midst of a pressure layer.

The direction of radiation travel to the instrument is determined by the start/stop pressures defined by the user. For example, s start/stop pressure combination of 1000.0 40.0 would imply a downward looking instrument, as the radiation travels from a start pressure of 1000.0 mb (ground) to a stop pressure of 40.0 mb (instrument posn).

The program does check to ensure that the start/stop pressures are within the upper/lower bounds of the AIRS pressure layers - if not, they are reset to the appropriate value. For example, a stop pressure of 0.0 mb is reset to 0.005 mb, while any pressure above 1100.0 mb is reset to that value. Within the defined atmosphere, the user can ask kCARTA to output the radiance at any pressure level. The only restriction is that for any ONE atmosphere, the program can only output radiances at a maximum of kProfLayers pressures.

Note that the Jacobian calculations are performed only for one scenario i.e. radiance between top and bottom of atmosphere. In other words, if you ask the program to compute the radiances at a number of different pressures, for any of the defined atmospheres, the only Jacobian that is performed and output is for an instrument at the very top (downward looking instr) or very bottom (upward looking instrument).

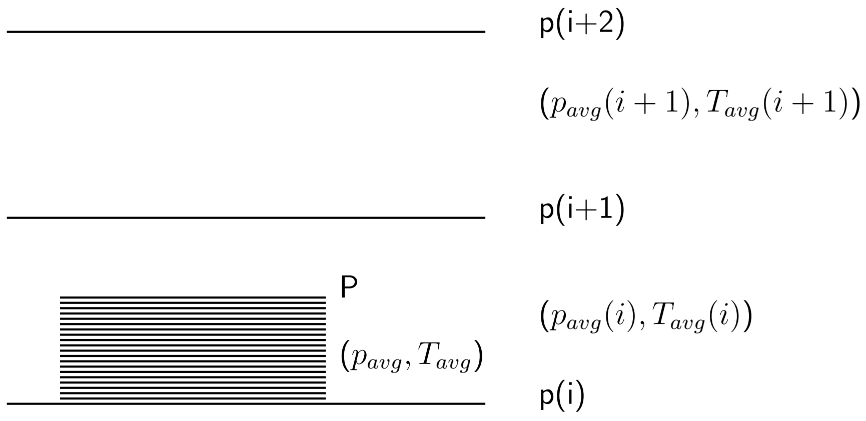
19.1 Fractional layers

The lower/upper fractions that define the atmosphere are taken into account as follows:

1) The specified endpoint pressure of the bottom layer (start pressure for downward looking instrument, or stop pressure for a upward looking instrument) defines the fractional bottom layer, rFracBot. Using this information, a modified temperature of this layer is used for the Planck radiance — an average of the temperature computed at this pressure and the temperature computed at the top of the layer for a downward looking instrument (or bottom, for a upward looking instrument).

Similarly, the specified endpoint pressure of the top layer (stop pressure for downward looking instrument, or start pressure for a upward looking instrument) defines the fractional top layer, rFracTop. Using this information, a modified temperature of this layer is used for the Planck radiance — an average of the temperature computed at this pressure and the temperature computed at the bottom of the layer for a downward looking instrument (or top, for a upward looking instrument).

2) Consider a downward looking instrument. Suppose the user wants to output the radiance at pressure P, which lies in the i th AIRS pressure layer, between AIRS pressure levels p(i),p(i+1), as shown.



From the AIRS layering, we know the pressures p(i+2),p(i+1),p(i) We therefore know the average pressures of layers i-1, i,i+1, as well as that of the fractional shaded layer, are given by

$$p_{avg}(i) = \frac{p(i+1) - p(i)}{ln(p(i+1)/p(i))} \qquad p_{avg}(i+1) = \frac{p(i+2) - p(i+1)}{ln(p(i+2)/p(i+1))}$$

$$p_{avg}(i-1) = \frac{p(i) - p(i-1)}{ln(p(i)/p(i-1))} \qquad p_{avg} = \frac{P - p(i)}{ln(P/p(i))}$$

From the profile, we know the avg temp of the i-1,i,i+1th layers are $T_{avg}(i+1), T_{avg}(i), T_{avg}(i-1)$ respectively.

Assuming that the average temperature of a layer varies linearly with $ln(p_{avg})$, we can then quadratically interpolate to find the temperature of the fractional layer, T_{avg}. (the quadratic interpolation uses the fractional layer, and the two closest other layers). This temperature is used as the Planckian temp of the shaded portion of layer i.

The scaling of the absorption coefficients for the full layer versus the fractional layer also needs to be determined. This fraction rF is computed simply by scaling linearly in pressure

$$rF = \frac{p(i) - P}{p(i) - p(i+1)}$$

where as usual, $p(i), p(i+1)$ are the AIRS pressure levels and P is the user specified pressure. The absorption coefficients for the fractional layer are then given by the product of absorption coefficient for the full layer × required fraction rF.

For the downward looking instrument, the bottom portion of the layer is the required fractional part, as shown. For an upward looking instrument, the top portion of the layer would be the required fractional part.

3) For a down looking instrument, when computing background thermal/solar contributions, ALL the layers above ground are used (ie top of layer kProfLayers - gnd) However, when the upwelling radiation incident at the instrument is computed, the fractional bottom layer (and any other fractional layers where the user specifies radiances to be output) is used so that

$$\begin{aligned}\text{Rad(pressure P)} &= \text{Emission(fractional layer)} + \\ &\quad \text{Rad(bottom of layer)*Trans(thru fractional layer)}\end{aligned}$$

where an average temperature is computed for the fractional part of the layer in a manner similar to that described above. The absorption coefficients used are also scaled by the required fraction.

4) For a up looking instrument, when the downwelling radiation incident at the instrument is computed, the fractional top part of the relevant layer

$$\begin{aligned}\text{Rad(pressure P)} &= \text{Emission(fractional layer)} + \\ &\quad \text{Rad(top of layer)*Trans(thru fractional layer)} + \\ &\quad \text{solar effects(thru fractional layer)}\end{aligned}$$

where an average temperature is computed for the fractional part of the layer in a manner similar to that described above. The absorption coefficients used are also scaled by the required fraction.

19.2 Broadening of the lines

The broadening of each line, due to the self component and the foreign component, is computed as

$$brd_{air} = (P - PS) \times abroad$$

$$brd_{self} = (PS) \times sbroad$$

$$brd = brd_{air} + brd_{self}$$

$$brd \rightarrow brd \times (296/T)^{abcoef}$$

For the atmosphere, since nitrogen and oxygen are dominant gases, the self broadening component would conceivably be important for these two. However, due to their structure, they are very weakly active in the infrared, and so for all practical purposes, there is hardly any change. Similarly, almost all the other gases in the atmosphere are mixed in very weakly, and so the contribution to the line width, due to the self component, is very small. Thus if there is a slight perturbation to the self pressure of almost any gas in the atmosphere, there would hardly be any change in the broadening. Water vapor is a special case, as sometimes an especially dry or wet profile is encountered, leading to noticeable changes in the broadening. Depending on the self pressure, the self width can be almost 10 times the foreign width. as the amount of water vapor in the atmosphere can be upto a few percent of the air in the (lower) atmosphere. Furthermore, water vapor concentrations have large temporal and spatial variation, and can vary by upto an order of magnitude. For these reasons, to do the spectroscopy of water vapor correctly, kCARTA interpolates compressed spectroscopy tables both in temperature as well as in partial pressure.

19.3 Radiative transfer algorithm

The standard Schwartschild equation for time independent radiation transfer through a plane atmosphere, can be written as [?, ?]

$$\mu \frac{dI(\nu)}{dk_a} = -I(\nu) + J(\nu)$$

where J is the source function, usually taken to be the Planck function. However, it could also include scattering terms, or need to account for non local thermodynamic equilibrium as well.

For a CLEAR SKY divided into parallel layers, this can be solved to give

$$R(\nu) = R_s(\nu) + R_{layeremission}(\nu) + R_{th}(\nu) + R_{solar}(\nu) \tag{2}$$

where the four terms are the surface, layer emissions, downward thermal and solar respectively. Using an isotropic reflectance of $(1-\epsilon)/\pi$, and denoting the Planck function as $B(T)$, ϵ as the surface emissivity, T_s as the surface temperature, the satellite viewing angle as $\theta_{satellite}$, the sun zenith angle as θ_{solar}, and discretizing the radiative transfer equation, the above four terms are written out as

$$R_s(\nu) \;=\; \epsilon B(\nu, T_s)\tau_{1\to\infty}(\nu, \theta_{satellite})$$

$$R_{layeremission}(\nu) = \sum_{i=1}^{i=N} B(\nu,T_i)(\tau_{i+1\rightarrow\infty}(\nu,\theta_{satellite}) - \tau_{i\rightarrow\infty}(\nu,\theta_{satellite}))$$

$$\begin{aligned} R_{th}(\nu) &= \frac{1-\epsilon}{\pi}\sum_{i=N}^{i=1}\int_0^{2\pi} d\phi \int_0^{\pi/2} d(cos(\theta))cos(\theta) \times \\ & B(\nu,T_i)(\tau_{i-1\rightarrow ground}(\nu,\theta) - \tau_{i\rightarrow ground}(\nu,\theta)) \end{aligned}$$

$$\begin{aligned} R_{solar}(\nu) &= \frac{1-\epsilon}{\pi} B(\nu,T_{solar})cos(\theta_{solar}) \times \\ & \tau_{N\rightarrow ground}(\nu,\theta_{solar}))\tau_{ground\rightarrow N}(\nu,\theta_{satellite}))\Omega_{solar} \end{aligned}$$

The above terms have been written in terms of layer to space transmittances. Alternatively the forward radiative transfer algorithm can easily be written iteratively; for example, the first two terms would be rewritten as :

$$R(\nu) = \epsilon_s B(T_s,\nu)\Pi_{i=1}^{i=N}\tau_i(\nu) + \sum_{i=1}^{i=N} B(T_i,\nu)(1.0-\tau_i(\nu))\Pi_{j=i+1}^{N}\tau_j(\nu) \qquad (3)$$

In what follows, the discretized form of the radiative transfer equation is used.

19.3.1 Background thermal radiation

The thermal background is included by incorporating the diffusion approximation (refer Liou etc). This involves replacing an integration over the half plane with using a single zenith diffusive angle θ_d (along with the 2π factor that arises from the azimuthal integration). The zenith integral of the form

$$d(\theta)\ sin(\theta)\ cos(\theta)\ B(T(i))\ \tau_{i\rightarrow ground}(\nu,\theta) \qquad (4)$$

can be replaced, using the mean value theorem, to

$$\frac{1}{2}B(T(i))\ \tau_{j\rightarrow ground}(\nu,\theta_d) \qquad (5)$$

where the angular integration has been reduced to a optimum diffusivity angle, so requiring the transmittance to be evaluated only for this angle $\tau_{j\rightarrow ground}(\nu,\theta_d)$. Note this angle varies for each frequency, for each layer.

This reduces the computation for the downward thermal contribution to the form

$$\frac{1}{2}B(T_i)\left[\tau_{i-1\rightarrow ground}(\theta_{d1}) - \tau_{i\rightarrow ground}(\theta_{d2})\right] \tag{6}$$

where based on the sum of the absorption coefficient upto the $i, i-1$ th layers, θ_{d1}, θ_{d2} are the optimum diffusion angles.

These optimum diffusion angles are computed as follows. Rewriting the transmission $\tau_{i\rightarrow ground}(\nu, \theta))$ as $exp(-\sum_{j=i-1}^{j=1} k_j/cos(\theta)) =$ $exp(-k^{(i)}/cos(\theta))$, this is seen to be the exponential integral of the third kind $E_3(k^{(i)})$, where $k^{(i)} = \sum_{j=i-1}^{j=1} k_j$. The exponential integral can easily be performed (e.g. Numerical Recipes, MATLAB toolbox), and the optimum diffusion angle for layer i, θ_d obtained from

$$\theta_d(k^{(i)}) = \frac{-k^{(i)}}{ln(2E_3(k^{(i)}))} \tag{7}$$

In the limit of $k^{(i)} \ll 1$, $\theta_d \rightarrow \arccos(0.5)$, while in the limit of $k^{(i)} \gg 1$, $\theta_d \rightarrow \arccos(1.0)$.

For a discrete set of values of $k^{(i)}$ between 0 and 10, the diffusion angles were computed and saved. A polynomial fit to the data, such that errors between the computed diffusion angle and the polynomial approximations were always less than 0.5%, was then made. In this fashion, kCARTA can very quickly compute θ_d for an arbitrary $k^{(i)}$.

The accuracy of this computation was checked by propagating the thermal background between the top of the atmosphere and the ground using this polynomial approximation, and comparing it to the results from a 40 point Gaussian quadrature. This was performed over the 605 to 2805 cm^{-1} region, for a variety of AIRS regression profiles, The typical brightness temperature error was less than 0.001 K.

However, this means that at each layer, for each wavenumber bin, the program has to compute the optimum diffusive angle. A way of speeding this up was decided upon, as follows. The value of θ_d that is often used is that of $\arccos(3/5)$ see e.g. Liou, especially for $k \leq 1$. As we wished to have a maximum error $\leq 0.1K$ in the brightness temperature measured by the instrument at the top of the atmosphere, throughout the wavenumber region encompassed by our spectroscopic kCARTA database, an investigation of the adequacy of the simple diffusion approximation (using $\arccos(3/5)$ was carried out, using the US Standard Profile and a selection of some of the AIRS regression profiles. Neglecting solar radiation, the total radiation at the top of the atmosphere was computed using the forward model and the reflected background thermal. The truth for the background thermal was an angular integration carried out using the exact diffusion angles at each layer, using the polynomial approximation instead of computing the exponential integral. The test value was obtained by using the diffusive angle of $\arccos(3/5)$ in the downward thermal at each layer, before reflecting it and including it in the overall radiance. As expected, in the wave number regions where the atmosphere was blacked out, the diffusion approximation was perfectly acceptable. However, in the regions where the atmosphere was transparent, such as the ozone region, the errors could be as large as 0.2K, especially if one used realistic surface emissivity values of 0.8 in this region.

A more desirable thermal approximation was then searched for. Recall that the layering structure of kCARTA assumes an atmosphere divided into kProfLayers pressure layers. Due to the $\tau(layer \rightarrow ground)$ factor at each level, one sees that the most significant contribution to the thermal background is from the bottom layers. For the topmost layers ($kProfLayers$ down to $J+1$), the simple diffusive approximation was used (one angle, $\arccos(3/5)$ at all layers). For the bottom J layers, the accurate diffusion angle for calculated for each layer, based on the polynomial approximation to $\theta_d(k(i \rightarrow ground))$. Depending on the wavenumber region, the value of J used produces less than 0.1 K errors in all profiles. For instance, where the atmosphere is blacked out (e.g. in the water region, about 1500 cm^{-1}), a value of $J = 6$ was sufficient, while a transparent region such as about 2500 cm^{-1}, a larger value of $J(= 30)$ was used. For the sampled profiles, using a surface emissivity value of 0.8, this always produced less than 0.1k brightness temperature errors, and had the advantage of being faster than if the angle were computed at all layers, for all frequencies.

Note that when the background thermal contribution to the Jacobians is computed, the topmost (kProfLayers-J/2) layers are assumed to have no contribution. The bottommost J/2 layers are then assumed to have an acos(3/5) diffusive angle, and this is the value used. This method is chosen because we are interested in the sign of the Jacobian, not the absolute magnitude. In addition, only the bottom few layers have a large thermal contribution to the Jacobian (typically less than 30%).

In section *RADNCE, the thermal background parameter kThermal, has three possible values the user can use: -1,0,1. If a value of -1 is chosen then no background thermal is included. If a value of 0 is chosen, the diffusive approximation is used (using acos(3/5) at the top layers, and then on a monochromatic basis, using the optimum diffusive angle at the lower layers). If a value of 1 is chosen, the code does an analytic integration (Gaussian Quadrature) over the zenith angle, which is accurate but very slow.

19.3.2 Solar radiation

The solar contribution is much easier to include than the thermal contribution; assuming the sun radiates as a blackbody whose temperature is 5600 K, the solar term that in incident at the earth's surface is given by

$$B(5600K,\nu)\ \Omega_{solar}\ \tau(top \rightarrow ground)\ cos(\theta_{solar}) \tag{8}$$

where $\Omega_{solar} = \pi(r_{se}/r_e)^2$ is the geometry factor that accounts for the sun-earth distance and radius of sun. The $cos(\theta_{solar})$ is the geometry factor accounting for the solar radiation coming in at an angle with respect to the vertical. This solar radiation is then reflected back up to the instrument, where, as for the thermal background, an isotropic reflectance factor of $(1-\epsilon)/\pi$ is used.

Both these thermal/solar radiation terms can be easily turned on/off just before runtime by simply setting relevant parameter switches.

19.3.3 Upward looking instrument

The code can compute radiance measured by an upward looking instrument. This is achieved by only computing the downward flux at *one* angle (the satellite viewing angle). Once again, this feature can be turned on/off by simply setting a parameter before runtime.

The present AIRS layering is too coarse to give an accurate estimate of the radiation measured by an upward looking instrument. However, one can ask the code to give an estimate, using the following simple equation :

$$R(\nu) = R_{layer}(\nu) + R_{solar}(\nu) \tag{9}$$

where the two terms are the layer emissions and solar respectively. Again denoting the Planck function as $B(T)$, the satellite viewing angle as θ_{sat}, the sun zenith angle as θ_{solar} (which is equal to the satellite viewing angle if the sun fills the field of view), and discretizing the radiative transfer equation, the two terms are written out as

$$\begin{aligned} R_{layer}(\nu) &= \sum_{i=N}^{i=1} B(\nu,T_i)\,[\tau_{i-1\rightarrow ground}(\nu,\theta_{sat}) - \tau_{i\rightarrow ground}(\nu,\theta_{sat})] \\ R_{solar}(\nu) &= B(\nu,T_{solar})\ \tau_{N\rightarrow ground}(\nu,\theta_{solar}) \end{aligned}$$

19.4 Jacobian algorithm

Here we describe the algorithm for a downward looking instrument. Consider only the upward terms in the radiance equation (the layer emission and the surface terms), reproduced here for convenience. Assuming a nadir satellite viewing angle we have :

$$R(\nu) = \epsilon_s B(T_s,\nu)\tau_{1\to N}(\nu) + \Sigma_{i=1}^{i=N} B(T_i,\nu)(1.0-\tau_i(\nu))\tau_{i+1\to N}(\nu) \qquad (10)$$

Then, one obtains, after differentiation with respect to the m-layer variable s_m, (where the differentiation can be with respect to gas amount or layer temperature $s_m = q_{m(g)}, T_m$)

$$\frac{\partial R(\nu)}{\partial s_m} = \epsilon_s B(T_s)\frac{\partial \tau_{1\to N}(\nu)}{\partial s_m} + \sum_{i=1}^{N} B(T_i,\nu)(1.0-\tau_i(\nu))\frac{\partial \tau_{i+1\to N}(\nu)}{\partial s_m} +$$

$$\sum_{i=1}^{N} \tau_{i+1\to N}\ \frac{\partial}{\partial s_m}\left[B(T_i,\nu)(1.0-\tau_i(\nu))\right] \qquad (11)$$

As usual, $\tau_m(\nu) = exp^{-k_m(\nu)}$, $\tau_{m\to N}(\nu) = \Pi_{j=m}^{N} exp^{-k_j(\nu)}$. Performing the above differentiation,

$$\begin{aligned}\frac{\partial R(\nu)}{\partial s_m} \quad = \quad & \left[\epsilon_s B(T_s)\tau_{1\to N}\right](-1)\frac{\partial k_m(\nu)}{\partial s_m} + \\ & \left[\sum_{i=1}^{m-1}(1.0-\tau_i(\nu))B_i(\nu)\tau(\nu)_{i+1\to N}\right](-1)\frac{\partial k_m(\nu)}{\partial s_m} + \\ & \left[(1.0-\tau_m(\nu))\frac{\partial B_m(\nu)}{\partial s_m} - B_m(\nu)\frac{\partial \tau_m(\nu)}{\partial s_m}\right]\tau_{m+1\to N}(\nu)\end{aligned}$$

The individual Jacobian terms in kCARTA code can then by obtained as follows. Recall the layer transmission are related to absorption coefficients by

$$\tau_m(q_{m(g)}) = exp^{-k(T_m)q_{m(g)}/q_{m(g)}^{ref(g)}} \qquad (12)$$

Then for all gases other than water, using the SVD compressed notation,

$$k_{m(g)}(\nu) = \frac{q_{m(g)}}{q_{m(g)}^{ref}}\sum_{l=1}^{L} c_{l(g)}(T_m,m)\Psi_l \qquad (13)$$

from which the gas amount derivative is simply

$$\frac{\partial k_m}{\partial q_{m(g)}} = \frac{k_m}{q_{m(g)}} \tag{14}$$

while for water,

$$k_{m(w)}(\nu) = \sum_{l=1}^{L} c_{l(w)}(T_m, m, q_m)\Psi_l \tag{15}$$

from which the water amount derivative is

$$\frac{\partial k_m}{\partial q_{m(w)}} = \sum_{l=1}^{L} \frac{\partial c_{l(w)}}{\partial q_{m(w)}}\Psi_l \tag{16}$$

The temperature derivative can similarly be written as

$$\frac{\partial k_m}{\partial T_m} = \sum_{g=1}^{g=G}\sum_{l=1}^{L} \frac{\partial c_{l(g)}}{\partial T_m}\Psi_l \tag{17}$$

where the double sum is over the singular vectors and the gases.

While doing the spline interpolations of the coefficients $c_{l(g)}$, the derivatives $\frac{\partial c_{l(g)}}{\partial T_m}$, $\frac{\partial c_{l(w)}}{\partial q_{m(w)}}$ can be obtained concurrently [?] in the compressed space. (These Jacobians can also be calculated, in compressed space, but by "perturbing" the gas amounts/layer temperatures and then doing a finite difference derivative, before performing the uncompression). Multiplying by the orthonormal basis matrix U then immediately gives the analytic derivatives. Performing the calculations of the Jacobians in the compressed representation is therefore easily achieved. As these radiance Jacobians are obtained in 10000 point chunks, for all kProfLayers layers, they are easier to obtain than finite difference Jacobians.

19.5 Cross section and water continuum jacobians

As described above, the cross-section Jacobians are also obtained exactly, by analytic differentiation of the terms used to compute these quantities in the first place.

The water continuum jacobians are a little more complicated. The continuum optical depth contribution is given by

$$k_{con} = k_{self} + k_{forn} = \gamma(T)q\{c_s(T)p_s + c_f(T)p_f\}$$

where $\gamma(T)$ are the Van-Huber corrections to the lineshape (essentially consiting on tanh and other terms), while the c_s, c_f are the self and foreign coefficients respectively. In general, c_s depends on temperature; while the standard CKD models do not have c_f depending on temperature, our laboratory data analysis does indicate some dependancy in the 6 μm band, which is reflected in the above formulation. p_s, p_f are the self (water) and foreign (mainly nitrogen) pressures in the layer.

Looking at the above equation, naively the amount jacobian for the self continuum is $dk_{self}/dq = \gamma(T)c_s(T)p_s$. However, if we recall that $p_s = nKT$, then the (water vapor) layer gas amount in $q = nL = (p_s/KT)L$, where L is the layer thickness. This means that the self continuum contribution to optical depth is

$$k_{self} = \gamma(T)qc_s(T)p_s = \gamma(T)q^2c_s(T)(KT/L)$$

from which

$$dk_{self}/dq = \gamma(T)2qc_s(T)(kT)/L = 2\gamma(T)c_s(T)p_s$$

which is a factor of two larger than the previous estimate.

19.6 solar and background thermal Jacobians

The solar and background thermal terms for inclusion in the Jacobian calculations are also included in the algorithm. However, due to the increase in run-time of the code when computing the Jacobians, at present the only possible computation for the thermal background Jacobians is using the diffusive approximation $\arccos(3/5)$ at $lower$ levels, independent of whether the forward model radiative transfer algorithm used the accurate computation or the diffusive/accurate combination. Because of this, there would be slight differences if one compared the computed Jacobians to those obtained using finite differences between two almost similar parameterizations of the forward model.

The Jacobians obtained using the compressed representation are much faster than uncompressing the coefficients, doing a radiative transfer, perturbing the relevant layer, and doing another radiative transfer, after which a finite difference radiance Jacobian is obtained. The reason is easy to see – one would have to do these perturbed calculations for $each$ gas amount, at $each$ layer, instead of obtaining the Jacobians in big chunks.

19.7 Weighting functions

Additionally, weighting functions $W_i(\nu)$ are also computed and output as part of the overall Jacobian file :

$$\begin{aligned} R_{layeremission}(\nu) &= \Sigma_{i=1}^{i=N} B(T_i,\nu)(1.0 - \tau_i(\nu))\tau_{i+1\rightarrow N}(\nu) \\ &= \Sigma_{i=1}^{i=N} B(T_i,\nu)W_i(\nu) \end{aligned}$$

19.8 Miscellaneous notes about kCARTA Jacobians

In addition to the gas amount/layer temperature Jacobians and weighting functions described above, the Jacobians with respect to the surface temperature and surface emissivity are also computed. The Jacobian of the background thermal contribution with respect to the surface emissivity, and the Jacobian of the solar contribution with respect to the surface emissivity are also output. When the instrument is upward looking, these last four derivatives are all meaningless and are set to zero.

Another feature of the code is that the Jacobians can be output in any of three modes. The first is a raw $dR/d(var)$ mode, where R is a radiance, and var could be gas amount such as layer temperature etc. Another mode is a $dR/d(var) \times \Delta(var)$ mode, where if var is a gas amount, then we have appropriately weighted the Jacobian with the gas amount at that layer. The third mode is $d(BT)/d(var) \times \Delta(var)$ mode, where all the results now are Jacobians with respect to brightness temperatures, BT.

Once again, the turning on or off of the Jacobians can be achieved simply by setting the appropriate parameter at run time. Furthermore, the inclusion of thermal background to the Jacobian can be turned off (resulting in a significant decrease in run time) at the expense of incorrectly estimating the Jacobians at the lowest levels (as these are where the bulk of the background thermal contribution comes from). One can use either the analytic or the finite difference methods to calculate the Jacobians, or set any of the three modes to output the Jacobian results. In addition these computations can be performed for both down and up looking instruments.

20 Science : NLTE and TWOSTREAM scattering

20.1 Non LTE computations

Higher up in the atmosphere, the lower gas densities imply fewer collisions, which means that the lower and higher state populations of some molecules might be better described with a different, non local temperature. This usually happens quite high up in the atmosphere (eg above 90 km for the 15 μm CO$_2$ band), where there are very few molecules; this means that the change in optical depth in the upper atmosphere is insignificant for a spaceborne nadir viewing instrument, leading to unnoticeable changes in observed brightness temperature. However, previous studies of the atmosphere by limb viewers have shown that for the 4 μm CO$_2$ band, the solar pumping very strongly affects the vibrational temperatures of the transitions in this region; NLTE can be seen above heights as low as 45 km, where there are enough molecules present to noticeably alter the optical depths. While this will not affect the daytime observations of instruments on board aircraft, spaceborne instruments such as HIRS and AIRS will certainly see the enhancement in observed brightness temperatures. HIRS is a radiometer based instrument, with very low resolution; AIRS is a much higher resolution instrument (the channel widths in this region are about 2 cm^{-1}), from which it should be possible to make spectral comparisons between observations and NLTE models.

kCARTA allows the user to define a separate NON LTE profile for the (vibrational) states of one or more bands of (different) molecules. With this information, it can compute the NLTE optical depths and Planck function modifiers for these user specified choices "on the fly," adding on the "background LTE" optical depths of the rest of the molecules plus the rest of the states of the molecule(s) in question. Having done all this, kCARTA then computes a TOA radiance. At present, the NLTE capabilities of kCARTA are optimized for the 4 μm band of CO$_2$; additionally, instead of using linemixing, the Cousin lineshape is used as it is a simpler model to incorporate.

20.1.1 Computing the optical depths

Most of the modifications to the code use the standard nonLTE analysis [?, ?, ?, ?]. Let T_l be the local thermodynamic temperature of layer l, while $T^{g,l}_{vib}(i)$ be the NLTE vibrational temperature of the ith band in question, for gas g at the same layer l. With the vibrational band center denoted by ν_0, the optical depths at NLTE is related to the LTE optical depth by

$$k^{g,l}_{nlte}(i,\nu_0)q^{g,l} = k^{g,l}(i,\nu_0)\alpha^{g,l}(i,\nu_0)q^{g,l}$$

where $k^{g,l}(i,\nu_0)$ is the LTE absorption coefficient, $q^{g,l}$ is the gas amount in lthe layer and $\alpha^{g,l}(i,\nu_0)$ is an adjustment factor, that depends on the population enhancememt or depletion in the lower and upper levels of the vibrational transition under consideration. Let $r_j, j = 1,2$ be the population ratios of the lower level $(j = 1)$ and upper level $(j = 2)$; these population ratios are the ratios between the level populations n_j at NLTE vs LTE :

$$r_1 = \frac{n_1^{NLTE}(T_{vib})}{n_1^{LTE}(T_l)} \quad r_2 = \frac{n_2^{NLTE}(T_{vib})}{n_2^{LTE}(T_l)}$$

Letting g_1, g_2 be the Boltzmann statistical weights of the transition, the equilibrium population ratio between the upper and lower levels is given by [?, ?]

$$\Gamma = \frac{g_1 n_2(T_l)}{g_2 n_1(T_l)} = exp(-hc\nu_0/K_B T_l)$$

The above terms can be combined to [?, ?] give an expression for the adjustment factor

$$\alpha^{g,l}(i,\nu_0) = \frac{r_1 - r_2\Gamma}{1-\Gamma} \times f_i$$

where f_i is the correction to the vibration contribution to the partition function [?, ?]. As the vibrational temperature approaches the local kinetic temperature, the adjustment factor goes to unity.

Summing over all gases and bands, and using $\zeta(\nu_0,\nu)$ to denote the effects of lineshape, we have the following expression for the total optical depth τ_l of layer l

$$\begin{aligned}\tau_l &= \sum_{g,i} \alpha^{g,l}(i,\nu_0)k^{g,l}(i,\nu_0)q^{g,l}\zeta(\nu_0,\nu) \\ &= \sum_{g,i(LTE)} k^{g,l}(i,\nu_0)q^{g,l}\zeta(\nu_0,\nu) + \sum_{g,i(NLTE)} \alpha^{g,l}(i,\nu_0)k^{g,l}(i,\nu_0)q^{g,l}\zeta(\nu_0,\nu)\end{aligned}$$

where we have broken the optical depth into the NLTE contribution (consisting of the vibrational bands of the gas(es) in question) and the LTE contribution ($\alpha = 1$, consisting of the weaker bands as well as other gases). As the vibrational temperature approaches the local kinetic temperature, the adjustment factor goes to unity, which leaves the optical depths unchanged.

20.1.2 Computing the source term for radiative transfer equation

The simple, nonscattering 1D radiative transfer equation is given by

$$\mu \frac{dI(\nu)}{dk_a} = -I(\nu) + J(\nu)$$

where J is the source function, usually taken to be the Planck function.

The general source function for a two level system is given by [?, ?, ?]

$$J(\nu, T_l) = 2hc^2\nu^3[\frac{n_1 g_2}{n_2 g_1} - 1]^{-1}$$

Using the expression for Γ above, at LTE this reduces to the usual Planck source function

$$B(\nu, T_l) = 2hc^2\nu^3[exp(+hc\nu_0/K_B T_l) - 1]^{-1}$$

while for the general NLTE case, this term can be written as

$$J(\nu, T_l) = 2hc^2\nu^3[\frac{r_1}{r_2} exp(+hc\nu_0/K_B T_l) - 1]^{-1}$$

The source term in the solution to the radiative transfer equation can then be rewritten as [?, ?, ?, ?] $\beta^{g,l}(i,\nu_0)B(\nu, T_l)$ where for one individual line,

$$\beta^{g,l}(i,\nu_0) = \frac{r_2^{g,l} k^{g,l}(i,\nu_0) q^{g,l}}{\alpha^{g,l}(i,\nu_0) k^{g,l}(i,\nu_0) q^{g,l}}$$

Generalizing for a sum over many lines,

$$\beta_l = \sum_{g,i} \beta^{g,l}(i,\nu) = \frac{\sum_{g,i} r_2^{g,l}(i,\nu_0) k^{g,l}(i,\nu_0) q^{g,l} \zeta(\nu_0,\nu)}{\sum_{g,i} \alpha^{g,l}(i,\nu_0) k^{g,l}(i,\nu_0) q^{g,l} \zeta(\nu_0,\nu)}$$

Just as was done for τ_l above, both the numerator and denominator can be broken down into sums over the LTE and NLTE components, so that an overall numerical answer for β_l can be computed easily. As the vibrational temperature approaches the local kinetic temperature, the adjustment factor goes to unity, which makes the Planck modification factor also tend to unity.

20.1.3 Solution to the Radiative Transfer Equation

Assume that the radiation incident one one side (say the bottom) of a layer is $I(\nu)_{l-1}$. The complete solution to the radiative transfer equation, which gives the radiation exiting the other side (say the top) of the layer is then [?, ?, ?, ?]

$$I(\nu)_l = I(\nu)_{l-1} exp(-\tau_l) + B(T_l)\beta[1 - exp(-\tau_l)]$$

where τ_l is given by the expression for total optical depth, and β_l is the Planck function modifier, both given above.

Figure 10 shows a comparison plot of kCARTA vs GENLN, compared to some actual NLTE data seen in daylight viewing conditions on the AIRS instrument. Plotted for AIRS, is actual NLTE dayime observations minus LTE Fast Model computations, averaged over about 100 spectra taken on August 31, 2002. For KCARTA and GENLN2, we plot (NLTE - LTE) calculations, with the TOA at about 85 km. The main CO2 $\Sigma - \Sigma, \Delta - \Delta, \Pi - \Pi$ bands are in NLTE while the weak background lines are in LTE. The Cousin lineshape is used in the simulations.

20.2 TWOSTREAM scattering

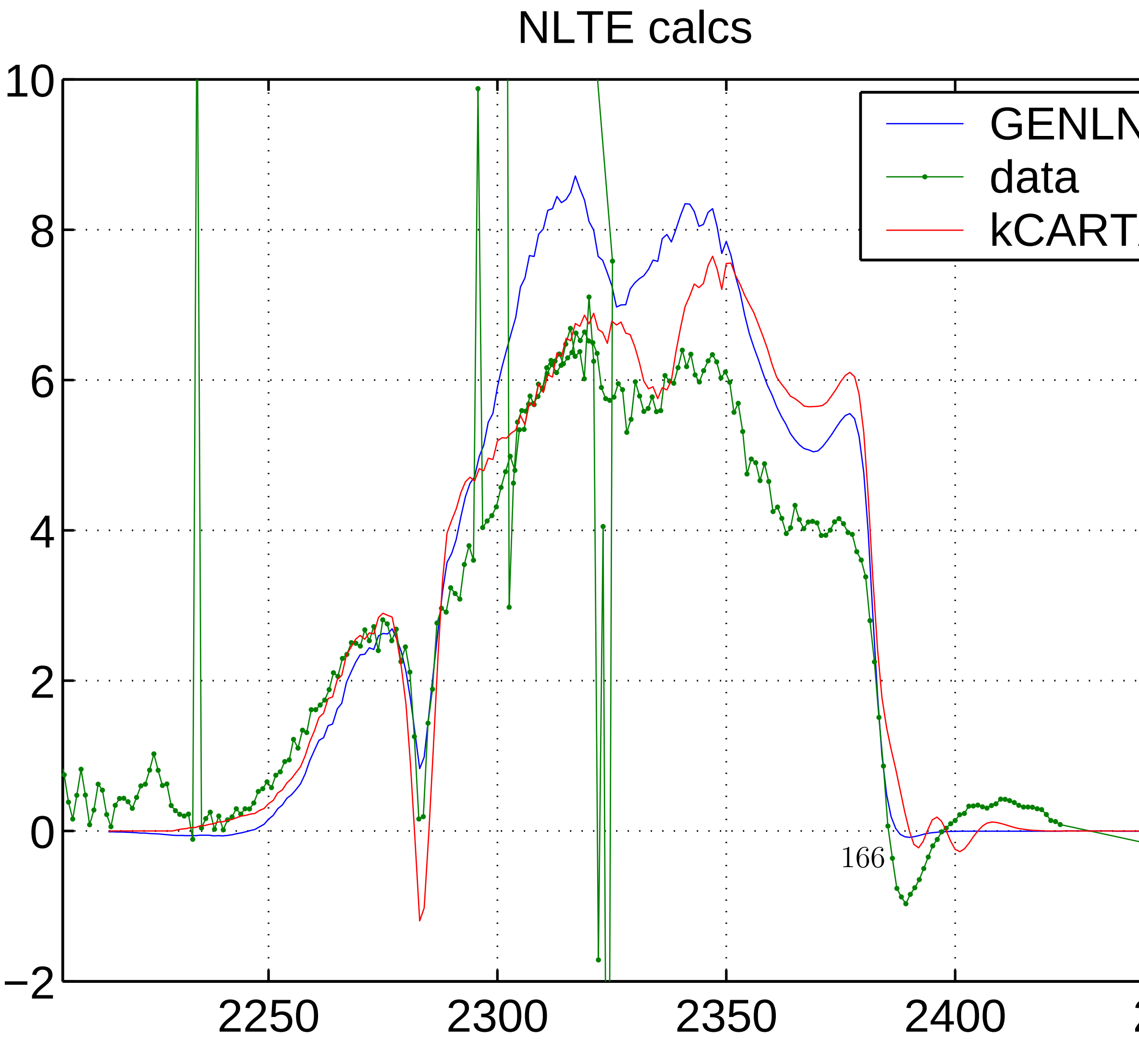
Time independent radiative transfer can be described by Schwartzchild's equation [?, ?]. As a beam propagates through a medium, the change in diffuse beam intensity $I(\nu)$ in a plane parallel medium is given by

$$\mu \frac{dI(\nu)}{dk_a} = -I(\nu) + J(\nu)$$

where μ is the viewing angle, k_a is the optical depth due to absorption, ν is the wavenumber and $J(\nu)$ is the source function. If the medium is nonscattering, such as would be expected in a "clear sky," the source function is simply the Planck emission $B(\nu, T)$ at the layer temperature T, implying that there is absoprtion attenuating the beam, and Planck emission from the layer adding to the beam. If we assume that the temperature of the layer is constant, and that the incident intensity is $I(\nu, 0)$, the equation is trivial to solve :

$$I(\nu, k_a) = I(\nu, 0)e^{-k_a/\mu} + B(\nu, T)(1 - e^{-k_a/\mu})$$

$1 - e^{-k_a/\mu}$ is the emissivity E of the layer, $e^{-k_a/\mu}$ is the transmission of the layer, and the reflection R is 0. One can see that $R + T + E = 1$ in this simple case.


NLTE calcs
10
8
6
4
2
0
−2
2250
2300
2350
2400
GENLN
data
kCART

Since the atmosphere is not isothermal, it is best modeled by dividing it up into layers thin enough that the temperature variation across each layer does not give significant spectroscopic variation between the layer top and bottom. Having obtained the one layer solution, it is trivial to propagate the radiation through successive layers and compute the radiation incident at the instrument.

If the atmosphere is to be modeled more realistically, the effects of clouds and/or aerosols should be included. As above, there will be a reduction of the diffuse intensity $I(\nu, k_e)$ by single scattering and absorption (where k_e is the extinction crosssection, which is the sum of absorption k_a and scattering k_s cross sections) [?, ?]

$$\mu \frac{dI(\nu)}{dk_e} = -I(\nu)$$

The layer Planck emission $B(\nu, T)$ still contributes to the source function $J(\nu)$. However, to maintain thermal equilibrium, only the absorptive portion of the extinction is included, and so the contribution to the source term is now

$$B(\nu, T)\frac{k_a}{k_e} = B(\nu, T)\left(1 - \frac{k_s}{k_e}\right)$$

In addition, we need to include scattering of diffuse intensities at other angles $\mu\prime$ into the viewing angle, which in three dimensions would be given by [?, ?]

$$dI(\nu, \Omega, k) = k_s \mu \int_{4\pi} I(\Omega, \Omega\prime, k) P(\Omega, \Omega\prime) d(\Omega\prime)$$

as well as the scattering of the direct solar beam into the viewing beam [?, ?]

$$dI(\nu, \Omega, k) = k_s \mu I_{sun}(\Omega, \Omega_{sun}, k) P(\Omega, -\Omega_{sun})$$

Here $P(\Omega, \Omega\prime)$ is the phase function, which gives the probability of scattering from solid angle $\Omega\prime$) to solid angle Ω. The phase function and the extinction properties of the layer are computed using electromagnetic theory; if one assumes that the particles are spheres such as would be the case of raindrops in a cloud, then Mie theory [?, ?, ?] can be used to determine these properties; if one wants to describe the scattering properties of ice particles in a high altitude cirrus cloud, one could use more elaborate ray tracing programs to determine these properties.

If we consider the azimuthally symmetric case, the phase function is now [?]

$$P(\mu, \mu\prime) = \frac{1}{2\pi} \int_0^{2\pi} P(\mu, \phi; \mu\prime\phi\prime) d\phi\prime$$

Defining the single scattering albedo as $\omega_0 = \frac{k_s}{k_s+k_a}$, and pulling together all of the above, we finally have the radiative transfer equation to be solved [?, ?]

$$\begin{array}{rcl} \mu\frac{dI(\nu)}{dk_e} & = & I(\nu) - B(\nu,T)(1-\omega_0) - \\ & & \frac{\omega_0}{2}\int_{-1}^{+1} I(\nu,k_e,\mu\prime)P(\mu,\mu\prime)d(\mu\prime) - \frac{\omega_0}{4\pi}\pi I_{sun}P(\mu,-\mu_{sun})e^{-k_e/\mu_{sun}} \end{array}$$

This is an integrodifferential equation, which means that obtaining the intensity at an arbitary viewing angle μ requires knowledge of the intensity at various angles, as one needs to perform an integral of these intensities, weighted by the phase function. One way of evaluating the integral is by Gaussian Legendre quadrature, which minimises the error in the integral by picking a set of points over the $[-1,+1]$ interval. Depending on the number of quadrature points chosen, we have an n stream solution. Some scattering packages such as DISORT and CHARTS allow the user to pick the number of streams used. Others such as RTSPEC have a fixed number of streams. This should not be a very serious problem, as the large number of scatterers actually smooths out the phase function [?], and a twostream solution can be quite accurate. The RTSPEC package includes both the radiative transfer algorithm as well as Mie scattering code to compute the particle scattering properties (more accurately, the scattering properties of a distribution of particles). This package, as well as DISORT has been interfaced with kCARTA.

To be able to compute the radiance when a cloud is present, as well as a solar beam, we also developed a simple multilayer kTWOSTREAM scattering package. This combines the twostream speed of RTSPEC and allows the user to include solar beam scattering (DISORT also allows beam scattering, but is more slow). The atmosphere is divided up into three regions : clear from Top-Of-Atmosphere to CloudTop, cloudy, and clear from CloudBottom to Ground. While simple clear sky radiative transfer is computed in the clear layers, the reflection, transmission, emission and solar components at the twostream angles (R,T,E,B) and viewing angle (r,t,e,b) are computed for each cloudy layer, with the layers being added together if the cloud is a multilayer one. This gives the overall reflection, transmission, emission and beam scattering parameters of the cloud.

For both a downlook as well as an uplook instrument, we first compute the background thermal radiation that makes it down to the surface. When including the cloud layer in this initial computation, only the absorptive contribution of the cloud extinction depth is included. If the sun is "on", a similar computation of the direct solar beam component at the Earth's surface is performed. Together with surface emission, and the reflection of the background thermal and solar radiations, we propagate two beams back to the cloud bottom : one beam at viewing angle θ and a beam at the twostream angle $\arccos(1/\sqrt{3})$. Similarly we compute the radiation incident downwards at the cloud top at stream angle $\arccos(1/\sqrt{3})$ (and if necessary, the direct solar beam intensity that is incident at the cloud top, at solar angle θ_{sun}).

Having initialised the boundary conditions, we can propagate the up- and downgoing stream radiations (at $\pm arccos(1/\sqrt{(3)}))$ throught the cloud, after which we can compute either the upgoing radiation at cloud top, or down going radiation at cloud bottom, at the viewing angle. The third and final stage is to compute the radiation to the instrument.

Since the sun creates a natural asymmetry in the radiative transfer, we choose to merge the cloud layers from top to bottom. Another point to mention is that we compute the reflection, transmission and emission coefficients for arbitrary viewing angle, and so a casual check of these coefficients would make it seem that $r + t + e$ is not energy conserving (i.e. is not 1). However, if one limits the computations to a viewing angle that corresponds to that of the two streams, then energy is indeed conserved ($R + T + E = 1$, the uupercase denoting the coefficients at the stream angles while the lower case denotes them at arbitrary viewing angle).

To agree with the clear sky outputs of RTSPEC and DISORT, the only layer temperature variation is exponential-in-optical depth in the cloudy layers; for the clear layers, we use the average temperature of the layer (which agrees very well with the linear-in-tau clear layer variation used in the above mentioned packages. However, depending on the wavenumber region, it is apparent that the authors of the various scattering packages might need to agree on the exponential-in-tau variation both in cloudy and clear layers, as this could lead to brightness temperature differences of upto 0.6 K.

The layer addition is done in much the same fashion as is presented in Goody and Young [?]. The two stream equations are exactly solved for the layer in question (note that we define $k = 0$ at the bottom of the layer, and that there is an implicit wavenumber dependence ν) :

$$\begin{aligned}\mu_{+}\tfrac{dI^{+}}{dk} &= -I^{+} + \tfrac{\omega_0}{2}(I^{+}(1 + 3g\mu_{+}\mu_{+}) + I^{-}(1 - 3g\mu_{+}\mu_{+}))+ \\ &\quad B_b(1 - \omega_0)e^{\beta k} + \tfrac{\omega_0}{4}S_T e^{-(T-k)/\mu_{sun})} P(\mu_{+}, -\mu_{sun})\end{aligned}$$

$$\begin{aligned}-\mu_{+}\tfrac{dI^{-}}{dk} &= -I^{-} + \tfrac{\omega_0}{2}(I^{+}(1 - 3g\mu_{+}\mu_{+}) + I^{-}(1 + 3g\mu_{+}\mu_{+}))+ \\ &\quad B_b(1 - \omega_0)e^{\beta k} + \tfrac{\omega_0}{4}S_T e^{-(T-k)/\mu_{sun}} P(-\mu_{+}, -\mu_{sun})\end{aligned}$$

where we define

μ_+	upgoing stream angle
μ_-	downgoing stream angle $= -\mu_+$
I^+	upgoing stream intensity
I^-	downgoing stream intensity
k	optical depth
T	layer total optical depth (0 at bottom, T at top)
ω_0	layer single scattering albedo
g	layer asymmetry factor
B_b	radiance at bottom of layer
T_b	temperature at bottom of layer
T_t	temperature at top of layer
β	$1/Tlog_e(T_t/T_b)$
S_T	solar radiance at top of layer

The homogeneous part of this set of coupled equations is easily solved, giving the two eigenvalues for the two streams; the inhomogeneous part corresponding to the layer temperature variation and the solar beam incident on the top of the layer is also easily solved. The boundary conditions are the incident upward radiance at the bottom of the layer, and the incident downward radiance at the top of the layer. With this information, the twostream problem is completely solved for one layer.

For a multilayer cloud, at each spectral point, one could make the intensities continuous across layer boundaries. The drawback is that a potentially large matrix (depending on the number of layers the cloud occupies) would need to be inverted for each spectral point, making computations tedious. An alternative is to rewrite the exact solutions for one layer in terms of the monolayer reflection R, transmission T, layer emission E and beam B coefficients. One can show that $R + T + E = 1$ in this case. Having the solution for one layer, we can then add the layers together to obtain the solution for a multilayer cloud. It is easily appreciated that at each spectral point the only computations used in this multilayer model are simple multiplications and additions, instead of matrix inversions.

Using the two stream solution, the problem for arbitrary angles can now be solved. The radiative transfer equation in this case can be written as (for $\mu \geq 0$)

which can more easily be written as

$$\mu\frac{dI}{dk} \;=\; -I + J\prime(k, I^+(k), I^-(k))$$

where $J\prime(k, I^+(k), I^-(k))$ is the (Eddington's second solution) source function

$$\begin{array}{rcc} J\prime(k, I^+(k), I^-(k)) & = & \frac{\omega_0}{2}\left((I^+ + I^-) + 3g\mu\mu_+(I^+ - I^-)\right) \\ & & B_b(1-\omega_0)e^{\beta k} + \frac{\omega_0}{4}S_T e^{-(T-k)/\mu_{sun})}P(\mu, -\mu_{sun}) \end{array}$$

Since we already know the solutions to the twostream radiances I^+, I^-, this general equation can be exactly solved as well. The solution can be written as

$$I(k,\mu) = \left(I(0,\mu) + S_{up}(k)\right)e^{-k/\mu}$$

where $S_{up}(k)$ is a term that includes the scattering from the twostream radiances into the view angle stream, as well as layer emission and scattering from the solar beam into the viewing angle. A similar set of equations can be written and solved for $\mu \leq 0$. Having obtained the one layer solution for arbitrary angles, we can rewrite the solutions in terms of the more general refection, transmission, emission and beam coefficients (r,t,e,b) and then add layers together for a complete solution. Note that because there is scattering from other beams into the viewing beam, $r+t+e$ is not necessarily equal to one in this general case.

Since the code computes the twostream radiation incident at the top and bottom of the multicloud layer, as well as the radiation incident at the viewing angle, it can now propagate the twostream radiances through the cloud in either direction, and use that to compute the radiation exiting the cloud at the viewing angle. The final stage of the computation is to propagate the radiation through the remaining clear sky to the instrument.

21 Significant Changes from v1.10 to v1.11

Main improvement is to introduce NONLTE capability for the 4 um CO2. This will significantly slow down kCARTA, as a line by line computation is now done on the fly for the strongest bands, for the necessary layers. Most of this NonLTE code has been developed using the GENLN3 ideas.

22 Significant Changes from v1.09 to v1.10

Input profile, frequency bound settings and atmosphere definitions can now be set either through the namelist file, or through an AIRS RTP file. The appropriate libraries for the RTP file format can be obtained using our website. In addition, we have improved the optical depth computations using non AIRS layering. Also, we have changed radiance units to mW cm-2 sr-1/cm-1. Spectroscopy has been updated, using a mixture of improved linemixing parameters and chi functions for the CO2 4 um region, and an improved water vapor continuum in the 6.7 um region. Other than this, the usual small bug fixes expected inherent with such a large project, were done as necessary.

23 Significant Changes from v1.08 to v1.09

This version has gone away from any dependancy on the AIRS layering. In order to do this, klayers.x was rewritten (by Scott Hannon) so that cpblev.f contains the block data statement of the new layering. layout.f was modified (by Sergio Machado) so that it outputs outincLAY.param, outpresslevels.param anf outlayers.param with all necessary info.

The kCARTA subroutines that did the pressure interpolations of the subdivided or superdivided AIRS layers, have also been modified so that now they do a stright-forward weighted average layer pressure interpolation. Similarly, subroutine WaterAmountTempJAC had to be modified as well.

The gaussian integration points/weights are no longer hardcoded by gauss.param. Instead, subroutine FindGauss(N,raX,raW) is called as necessary N = 40 for accurate background thermal
= 2 for RTSPEC flux
= 10 for CLEARSKY flux ==

Fluxes for RTSPEC and DISORT can now be computed.

Radiances at pressure level boundaries for RTSPEC and DISORT can be output

```
kLongOrShort = -1 : output shortened binary file, summarizing nml file
               +1 : output longer binary file, which rehashes input nml file
                0 : only output data!!! only works for radiance file
                    (no flux,jac allowed yet)
```

In the Makefile, bkcarta.x is just the basic version of kCARTA. This can do basic optical depths and mixed paths, and clear sky radiances. It cannot do fluxes, jacobians or scattering computations. So the arrays in kcarta.param and scatter.param can be scaled back to size 1, to allow for this (they have been labelled with the comment "space saver dimensions."

24 Significant Changes from v1.07 to v1.08

This version now has both DISORT and RTSPEC interfaced to the code, to allow for scattering computations. When in RTSPEC mode, solar cannot be included either for up or down looking instruments.

The namelist nm_scattr section has been slightly extended, so that
1) we tell the code which scattering model to use
kWhichScatterCode = +1 for TWOSTR
kWhichScatterCode = +2 for RTSPEC
kWhichScatterCode = +3 for DISORT

2) if we are using DISORT, then we have 3 options to speed up the code kScatter = +1, DISORT will do rad tranfer on kDisStep pts (pts 1,1 + J, 1 +2J, 1 + 3J ... etc where J=kMaxPts DIV kDisStep The code will then do a linear interpolation of the chosen pts "interp(raFchosen,raInten) -¿ (raWaves,I)"

kScatter = +2, DISORT will do rad tranfer on kDisStep pts These points are chosen so that they are the lowest optical depth points (in layer closest to gnd) The code will then do a linear interpolation of the chosen points "interp(raKchosen,raInten) -¿ (raK,I)"

kScatter = +3, DISORT will do rad tranfer on kDisStep pts These points are chosen so that they span the min to max optical depth points (in layer closest to gnd) The code will then do a linear interpolation of the chosen points "interp(raKchosen,raInten) -¿ (raK,I)"

Conversely if we are using RTSPEC, then we set the model used (single, eddington or hybrid) by setting kScatter = +1 , +2 or +3

3) for DISORT to run fast, we can set the number of streams used kDis_nstr (defaulted to 16)

4) for DISORT to run fast, we can set the number of wavenumber points stepped over kDis_Pts (defaulted to 400)

5) Introduced a new parameter into *SCATTR, raExp(j), where j is the cloud under consideration. If set to 0 and a cloud is "expanded" from "one" layer to layers

(p1,p2), the IWP of each of these layers is the same, and sums up to IWP. If set to other than 0 and a cloud is "expanded" from "one" layer to layers (p1,p2), the IWP of the individual layers is exponentially decreased roughly as exp(-raExp(j)*p1/p), but the total IWP remains that set by the user

6) Changed RTSPEC so that clouds that occupy completely different regions, can be processed. Eg if cloud 1 is an aerosol cloud layer from KCARTA layers 4-5, and cloud 2 is a cirrus cloud from kCARTA layers 43-46, this is handled by setting a "third" cloud from layers 6-42, with IWP=0.0

The namelist nm_radnce section has had the meanings of settings of some parameters slightly altered, in particular raTSpace and iaKSolar for an uplooking instrument
:

For the nonscattering kCARTA algorithm, raTSpace(i) should always be 2.7K or thereabouts. If iaKSolar(i) = -1 then sun is NOT in FOV, while if iaKSolar(i) = 0,1 then sun IS IN FOV, at satellite view angle. Thus raKSolarAngle(i) is irrelevant.

For the scattering RTSPEC algorithm, raTSpace(i) should always be 2.7K or thereabouts. The sun CANNOT be in the FOV, so iaKSolar(i) = -1 is the only allowed possibility.

For the scattering DISORT algorithm, raTSpace(i) should always be 2.7K or thereabouts. If iaKSolar(i) = -1 then sun is NOT ON, while if iaKSolar(i) = 0,1 then sun is on, at arbitrary angle. Thus raKSolarAngle(i) is VERY relevant

The *SCATTR is more general in that it "expands" a cloud according to user parameters. Eg if cloud is from 259-390 mb, all the user has to do is say cloud has "1" layer, give the IWP/DME and these two start/stop pressures ... the code will automatically figure out that there are more than 1 kLAYERS layers used by this cloud. As long as the cloud has sequential layering, from TOP to BOTTOM, the code is happy .

25 Significant Changes from v1.06 to v1.07

The major change is to create gasIDs 101 and 102 for the self and foreign water continuum. The water continuum is computed using a lookup table based on the CKDv0,2.2,2.3,2.4 codes from LBLRTM. In addition, we have used the HITRAN98 database, along with our latest CO2 spectroscopy, to create a new kCompressed DataBase.

26 Acknowledegement

We wish to thank Dave Edwards of NCAR for allowing us to use his GENLN line-by-line code to test our kCompressed database against, give us lots of advice on how to write our code, and allowing us to use some modifications of his subroutines in our code. In addition he has given us much help regarding the NonLTE computations. We also wish to thank Dave Tobin of U. of Wisconsin-Madison for help in the CO2 line mixing spectroscopy, as well as help in determing the water vapor continuum coefficients. We also thank Istvan Laszlo of U. of Maryland, College Park for answering many questions regarding $DISORT$, and Frank Evans of U. of Colorado, for doing the same for $RTSPEC$. Thanks also go to Pat Arnott of the Desert Research Institute, for advice regarding the twostream code. Ji Gou (UMBC) and Szu-Chia Lee (SSEC/U-Wisc) were amongst the innocent users that reported bugs that required fixes!