Towards a Generic Radiative Transfer Model for the Earth's Surface-Atmosphere System: ESAS-Light

ESTEC Contract No AO/1-5433/07/NL/HE

WP2100:

Technical specification for libRadtran demonstration version

Claudia Emde, Ulrich Hamann, Arve Kylling,
Bernhard Mayer

Deutsches Zentrum für Luft- und Raumfahrt
Wessling, Germany

March 25, 2009

Contents

1 Introduction

From the requirements document (Emde et al., 2008, WP1300 report) for the *libradtran* demostration version to be delivered during this study the technical specifications have been derived.

libRadtran already includes a large number of options to simulate radiative transfer in Earth atmosphere-surface system. Detailed descriptions of all options are provided in the *libRadtran* documentation (Mayer et al., 2007). In the following a summary is given, which shows that libRadtran will meet the requirements that have been consolidated by the end of the ESAS-Light study.

2 Structure of the *libRadtran* toolbox

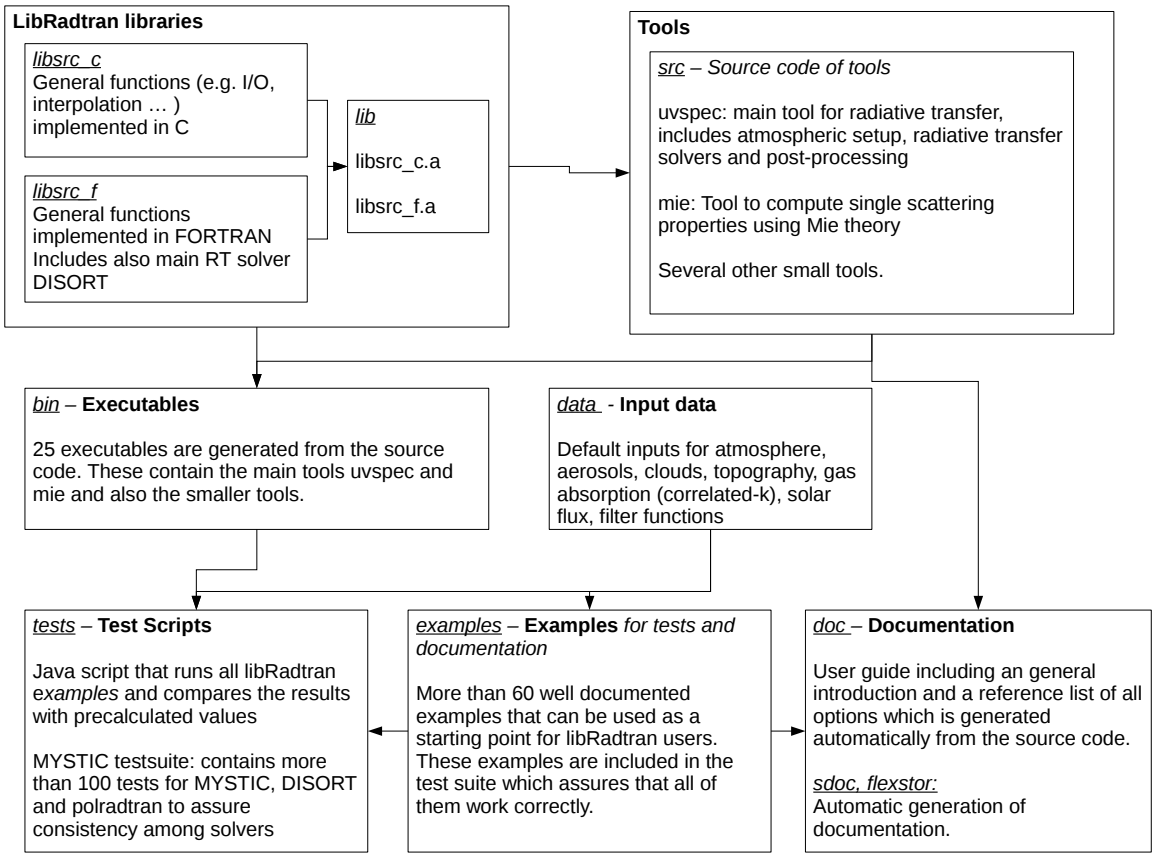


Figure 1: Structure of the *libRadtran* toolbox.

An overview of the structure of the *libRadtran* package is given in Fig. 1. *libRadtran* includes three source code directories: `src`, `libsrc_c` and `libsrc_f`.

The source code of the tools including the main tool `uvspec` (radiative transfer), the `mie` tool (compute optical properties of water clouds), and several small tools, for instance the `zenith` tool (compute the sun position for a given time and location) is contained in the `src` directory.

The directories `libsrc_c` and `libsrc_f` include the source code from which the libRadtran libraries are generated. The libraries consist of general functions like I/O routines, interpolation routines, and also the radiative transfer solvers. The compiled libraries can be found in the `lib` directory.

The `bin` directory includes the executables of all tools.

The `data` directory contains input data, like standard atmospheric models, solar flux, parameterizations of cloud and aerosol optical properties etc.

The `examples` directory contains well documented examples (currently more than 60) that can be used as a starting point by new *libRadtran* users. These examples are frequently tested using the script contained in the `test` directory. The tests are simply executed by the command `makecheck`.

Finally the `doc` directory includes the documentation of the program. The user guide consists of an introduction to *libRadtran* where the basic usage is explained. Furthermore it includes a complete reference of all options which is automatically generated from the code using the tool flex. The information is extracted from the file `src/uvspec_lex.l`, from which the reading routine of the input file is also generated.

3 Structure of the *uvspec* tool

The *libRadtran* model is structured into the following three essential parts: (1) An atmospheric shell which converts atmospheric properties like ozone profile, surface pressure, or cloud microphysical parameters into optical properties required as input to (2) the radiative transfer equation solver which calculates radiances, irradiances, actinic fluxes and heating rates for the given optical properties; and (3) post-processing of the solver output including multiplication with the extraterrestrial solar irradiance correction of Earth-Sun distance, convolution with a slit-function, or integration over wavelength (depending on the choice of the user). For an overview see Figure 2.

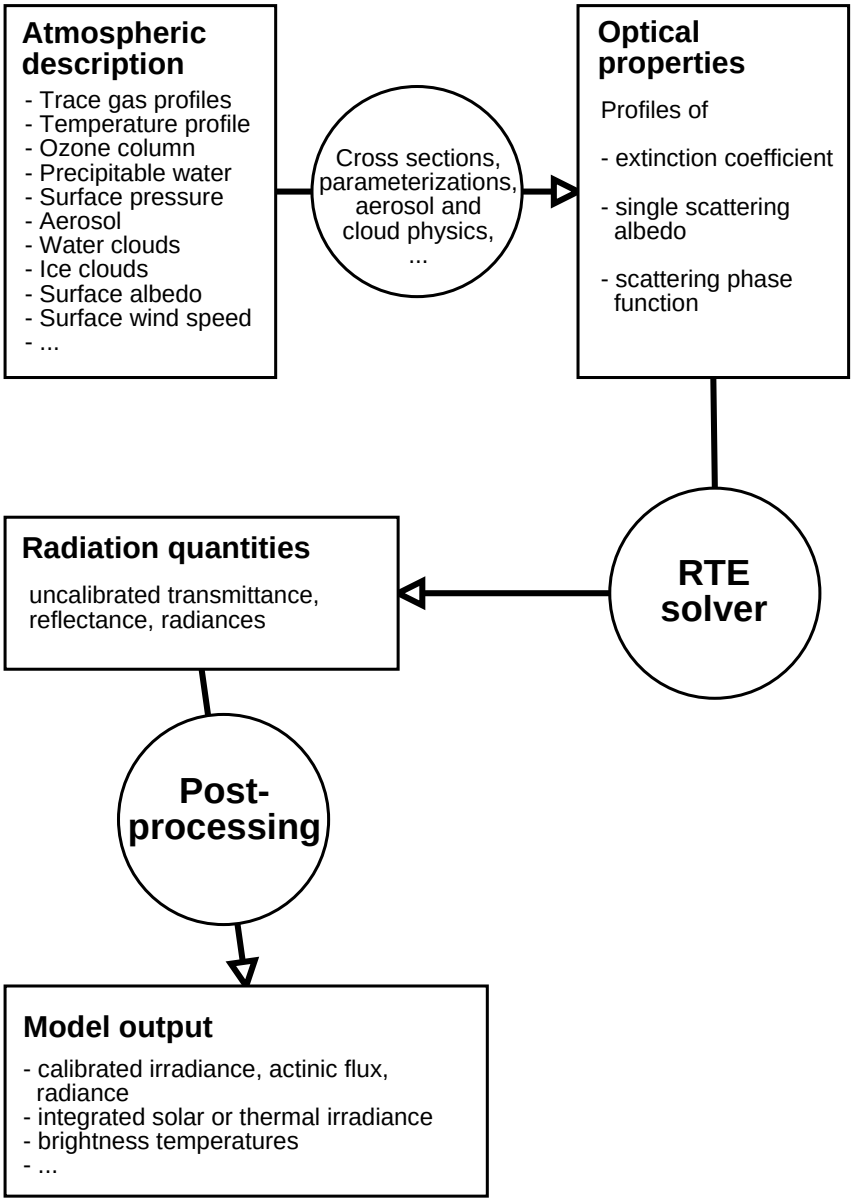


Figure 2: Structure of the *libRadtran* model

4 Radiative transfer solvers

LibRadtran includes a number of different radiative transfer solvers, so that the user can choose the most appropriate one for specific applications. Table 1 shows the available radiative transfer solvers. A more detailed table showing which options are available for which solvers is available on the project webpage: `http://esaslight.libradtran.org/internal/Wiki/doku.php?id=rtsolvers`. Among other solvers the *libRadtran* demonstration version will include

- a Monte Carlo model including polarisation for a three-dimensional atmosphere (*mystic*)
- a fast plane-parallel vector code that is well validated (*polradtran*)
- as a fast plane-parallel scalar code (*disort2*)
- a solver to calculate Raman scattering (*qdisort*)
- a fully spherical solver which enables simulation of refraction through the atmosphere (*mystic*)

5 Spectral resolution

libRadtran will provide the following methodologies to compute the absorption cross section by atmospheric gases

- Correlated-k parameterizations (Kato et al., 1999; Fu and Liou, 1992; Kratz and Varanasi, 1995; Ricchiazzi et al., 1998).
- Read line-by-line absorption coefficients calculated by the external line-by-line model ARTS (Buehler et al., 2005). A python interface is provided that generates *libRadtran* input files from the ARTS output.

6 Earth atmosphere-surface system

libRadtran is coded in a modular fashion, so that several of the options below may be combined to compute radiative transfer in the Earth atmosphere-surface system:

- Pre-defined clear sky atmospheres (Anderson et al., 1986) (pressure, temperature and trace gas profiles) or any user-defined atmospheric profiles, e.g. radiosonde profiles
- Pre-defined aerosol models (Shettle, 1984; Hess et al., 1998); type, optical thickness, ÅngstrÃ¶m coefficient, single scattering albedo, visibility etc. may be specified by the user.

Table 1: The radiative transfer equation solvers available in *libRadtran*.

RTE solver	Geometry	Radiation quantities	Reference	Comments
DISORT 1.3	1D, PP	E, F, L	Stamnes et al. (1988)	discrete ordinate
DISORT 2.0	1D, PP	E, F, L	Stamnes et al. (2000)	discrete ordinate
polradtran	1D, PP	**E, F, L**	Evans and Stephens (1991)	polarization included
twostr	1D, PS	E, F	Kylling et al. (1995)	two-stream; pseudo-spherical correction
sdisort	1D, PS	E, F, L	Dahlback and Stamnes (1991)	pseudo-spherical correction, double precision, customized for airmass calculations
spsdisort	1D, PS	E, F, L	Dahlback and Stamnes (1991)	pseudo-spherical correction, single precision, not suitable for cloudy conditions
tzs	1D, PP	L(TOA)		thermal, zero scattering
MYSTIC	3D, SP	E, F, L	Mayer (1999, 2000); Emde and Mayer (2007)	Monte Carlo$^{(a)}$
qdisort	1D, PS	E, F, L	to be developed in ESAS-Light study	discrete ordinate, Raman scattering

$^{(a)}$ not yet included in the free package; will become freely available at the end of ESAS-Light study

Explanation:
- PP, plane-parallel
- PS, pseudo-spherical
- SP, fully spherical
- 1D, one-dimensional
- 3D, three-dimensional
- E, irradiance
- F, actinic flux
- L, radiance
- L(TOA), radiance at top of atmosphere

Bold face **E**, **F**, and **F** indicate vector quantities.

- 1D or 3D water and ice cloud profiles can be specified by the user. Optical properties may be specified explicitely by the user or may be selected among various parameterizations: For ice clouds Baum et al. (2005a,b, 2007); Key et al. (2002); Yang et al. (2000); Fu (1996); Fu et al. (1998) are available, and for liquid water pre-calculated Mie tables and the parameterization by Hu and Stamnes (1993) are included in the *libRadtran* toolbox. Cloud optical thickness and effective particle size may be specified by the user.
- Water-air and surface-air interface reflectance can be modelled by various BRDFs for water (Cox and Munk, 1954a,b; Ebuchi and Kizu, 2002) , vegetation, snow and ice, urban surfaces and bare soils (Rahman et al. (1993) and MODIS data, that are provided in *libRadtran*. Alternatively, user-defined BRDFs can be used for simulations.

7 Graphical user interface and plotting tools

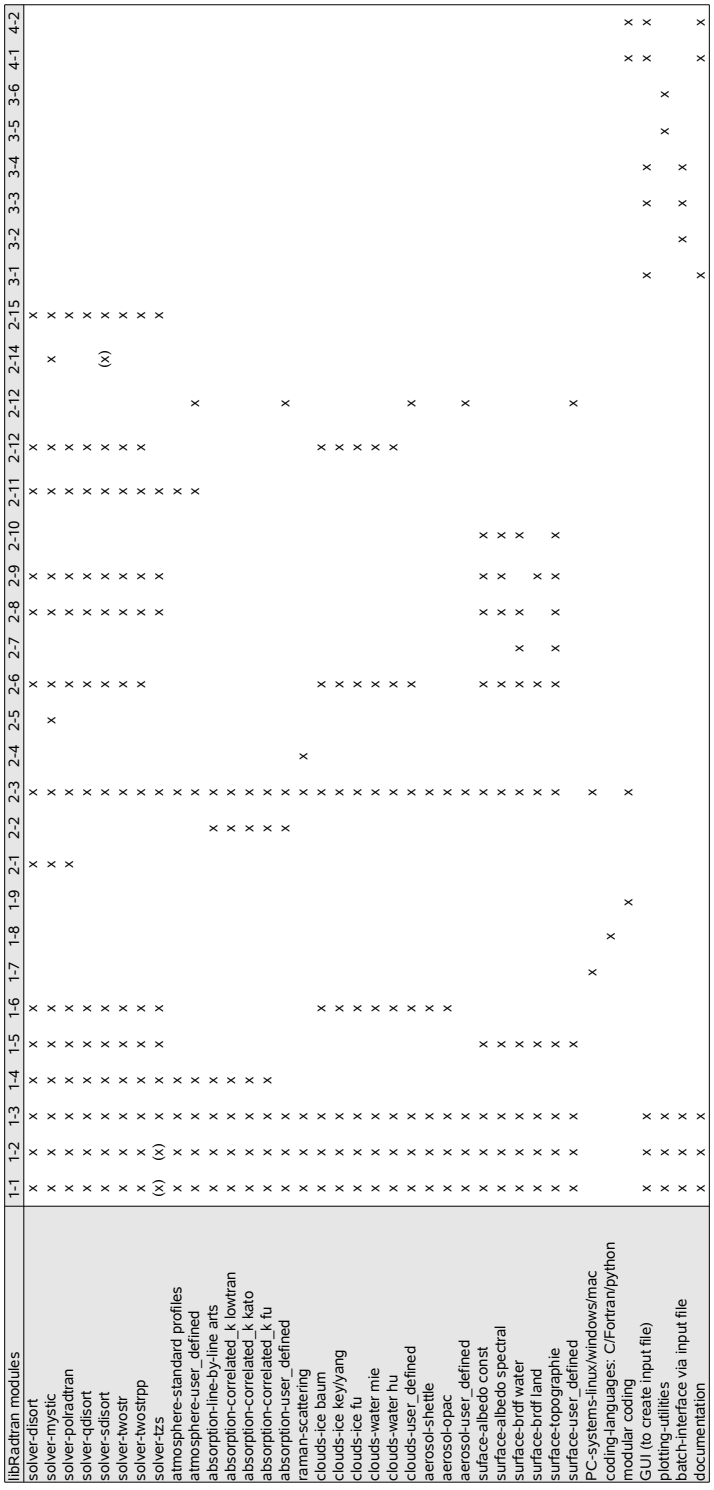
The *libRadtran* toolbox has a large number of input options. The large number of options reflects the flexibility and power of the tool, but may at the same time be overwhelming, especially for new users. A graphical user interface including a large subset, ideally all, of these options may be a more friendly interface to the uvspec model for many users. Such a GUI for uvspec should at least fulfill the following requirements:

- The GUI should be able to read existing input and corresponding output files.
- The GUI should be able to create new input files and change existing input files.
- The GUI should be able to run the uvspec model.
- The GUI should be able to plot appropriate data in both input and output files.
- As with the libRadtran package, the GUI should be portable and run under Linux, Mac OSX, and Windows operating systems. This may require installation of third party software by the user.
- The GUI should adopt the user documentation that already is in the uvspec tool in order to avoid having documentation in several locations.
- The GUI should utilize the existing input file error checking in the uvspec tool.

Some of these requirements are general and to which detail they will be developed has to be decided. However, the development of the GUI is envisaged as a continuous ongoing process once the basic framework is designed and implemented.

8 Compliancy matrix

The matrix shows which of the *libradtran* modules can be applied to meet the requirements, that have been consolidated in Emde et al. (2008, WP1300 report).



libRadtran modules	1-1	1-2	1-3	1-4	1-5	1-6	1-7	1-8	1-9	2-1	2-2	2-3	2-4	2-5	2-6	2-7	2-8	2-9	2-10	2-11	2-12	2-12	2-14	2-15	3-1	3-2	3-3	3-4	3-5	3-6	4-1	4-2
solver-disort	x	x	x	x	x	x				x		x			x		x	x		x		x		x								
solver-mystic	x	x	x	x	x	x						x		x	x		x	x		x		x	x	x								
solver-polradtran	x	x	x	x	x	x				x		x			x		x	x		x		x		x								
solver-qdisort	x	x	x	x	x	x						x			x		x	x		x		x		x								
solver-sdisort	x	x	x	x	x	x						x			x		x	x		x		x	(x)	x								
solver-twostr	x	x	x	x	x	x						x			x		x	x		x		x		x								
solver-twostrpp	x	x	x	x	x	x						x			x		x	x		x		x		x								
solver-tzs	(x)	(x)	x	x	x	x						x					x	x		x				x								
atmosphere-standard profiles	x	x	x	x								x								x												
atmosphere-user_defined	x	x	x	x								x								x												
absorption-line-by-line arts	x	x	x	x							x	x									x											
absorption-correlated_k lowtran	x	x	x	x							x	x																				
absorption-correlated_k kato	x	x	x	x							x	x																				
absorption-correlated_k fu	x	x	x	x							x	x																				
absorption-user_defined	x	x	x	x							x	x									x											
raman-scattering	x	x	x	x								x	x																			
clouds-ice baum	x	x	x	x	x	x						x			x					x		x										
clouds-ice key/yang	x	x	x	x	x	x						x			x					x		x										
clouds-ice fu	x	x	x	x	x	x						x			x					x		x										
clouds-water mie	x	x	x	x	x	x						x			x					x		x										
clouds-water hu	x	x	x	x	x	x						x			x					x		x										
clouds-user_defined	x	x	x	x		x						x			x																	
aerosol-shettle	x	x	x	x		x						x																				
aerosol-opac	x	x	x	x		x						x																				
aerosol-user_defined	x	x	x	x	x							x			x							x										
suface-albedo const	x	x	x	x	x							x			x		x	x	x													
surface-albedo spectral	x	x	x	x	x							x			x		x	x	x													
surface-brdf water	x	x	x	x	x							x			x	x	x															
surface-brdf land	x	x	x	x	x							x			x			x														
surface-topographie	x	x	x	x	x							x			x	x		x	x													
surface-user_defined	x	x	x	x	x																x											
PC-systems-linux/windows/mac							x					x																				
coding-languages: C/Fortran/python								x																								
modular coding									x			x																				
GUI (to create input file)	x	x	x																						x		x	x			x	x
plotting-utilities	x	x	x																									x	x			
batch-interface via input file	x	x	x																							x	x	x		x		
documentation	x	x	x																						x						x	x

9 Documentation of source code

The source code files include a documentation header which gives an information about which functions belong into this file.

The file header of `libsrc_c/ascii.c` is for example:

```
/*****************************************************************************/
/* In order to use the functions provided by the ascii library,   @30_10c@  */
/* #include <ascii.h> in your source code and link with libRadtran_c.a.     */
/*                                                                           */
/* @strong{Example:}                                                         */
/* Example for a source file:                                                */
/* @example                                                                  */
/*                                                                           */
/*   ...                                                                     */
/*   #include "../src_c/ascii.h"                                            */
/*   ...                                                                     */
/*                                                                           */
/* @end example                                                              */
/*                                                                           */
/* Linking of the executable, using the GNU compiler gcc:                    */
/* @example                                                                  */
/*                                                                           */
/*   gcc -o test test.c -lRadtran_c -L../lib                                 */
/*                                                                           */
/* @end example                                                 @c30_10@  */
/*****************************************************************************/


/*****************************************************************************/
/* The ASCII library provides functions for parsing ASCII files   @30_20c@  */
/* containing arrays of data.                                                */
/* An ASCII file is read line by line. Each line is split into fields;       */
/* a field is an arbitrary combination of characters which does              */
/* neither contain the line separator ('CARRIAGE RETURN') nor the field      */
/* separator ('SPACE').                                                      */
/*                                                                           */
/* In detail:                                                                */
/* @itemize @asis                                                            */
/*   @item Lines are separated by 'CARRIAGE RETURN'.                         */
/*   @item Tokens (or columns) are separated by one or more 'SPACE's.        */
/*   @item Empty lines are simply ignored.                                   */
/*   @item \% and # are comment symbols; text between a comment symbol       */
/*     and the next line separator is ignored                                */
/*   @item A comment symbol which is not at the beginning of a line is       */
/*     only recognized after a field separator, but not within a field       */
/*   @item The number of fields may differ from line to line.                */
/* @end itemize                                                              */
/*                                                                           */
/* A simple example for an ASCII file, which would be recognized as a        */
/* valid one-column or two-column ASCII file:                                */
/* @example                                                                  */
/*                                                                           */
/*   % This is an example for the input ASCII file for sdose,                */
/*   % the time integration program                                          */
/*   11.0     13.0   % the two hours around noon                            */
/*   10.0     14.0                                                          */
/*    9.0     15.0                                                          */
/*                                                                           */
/*   # total dose                                                           */
/*   -1.0     24.0    % integrate over maximum available time interval      */
/*                                                                           */
/*   # the following line shows, that an extra column does not matter       */
/*   2  3.4  17                                                             */
/*                                                                           */
/* @end example                                                              */
/*                                                                           */
/* For most purposes, ASCII_file2double and ASCII_free_double provide a      */
/* convenient way for parsing files.                                         */
/* @strong{Example:}                                                         */
/* @example                                                                  */
/*                                                                           */
/*  #include <stdio.h>                                                       */
/*  #include <ascii.h>                                                       */
/*                                                                           */
/*  int main(int argc, char ** argv)                                         */
/*  @{                                                                       */
/*    int rows=0, max_columns=0, min_columns=0;                              */
/*    int i=0, status=0;                                                     */
/*    double **value=NULL;                                                   */
/*                                                                           */
/*    status = ASCII_file2double ("test.dat",                                */
/*                            &rows,                                  */
/*             &max_columns,                                 */
/*             &min_columns,                                 */
/*             &value);                                      */
```

```
/*                                                                    */
/*    for (i=0; i<rows; i++) @{                                       */
/*      ... do something for each row of the matrix                   */
/*   @}                                                               */
/*                                                                    */
/*    ASCII_free_double (value, rows);                                */
/*                                                                    */
/*    return 0;                                                       */
/* @}                                                                 */
/*                                                                    */
/* @end example                                                       */
/*                                                                    */
/* For special purposes (ASCII files with 1,2,3, or 5 columns) there are   */
/* additionally functions read_1c_file, ... which facilitate the      */
/* access even more.                                                  */
/*                                                          @c30_20@ */
/**********************************************************************/
```

All functions should include a function header, which describes the purpose of the function, the input and output arguments, and it should name the author(s) of the function. Not all functions are fully documented yet, missing function headers will be included within the ESASLight study. The header of the function `ASCII_checkfile` can serve as an example for the documentation of all other functions:

```
/*****************************************************************************/
/* Function: ASCII_checkfile                                       @30_30i@ */
/* Description:                                                              */
/*  Check an ASCII file: count rows, minimum number of columns,              */
/*  maximum number of columns and the maximum length of a string;            */
/*  empty rows and characters after one of the comment symbols               */
/*  (either ASCII_COMMENT_1 or ASCII_COMMENT_2) are ignored.                 */
/*                                                                           */
/* Parameters:                                                               */
/*  char *filename:    name of the file which should be checked              */
/*  int  *rows:        number of rows found                                  */
/*  int  *min_columns: minimum number of columns, set by function            */
/*  int  *max_columns: maximum number of columns, set by function            */
/*  int  *max_length:  maximum length of a field, set by function            */
/*                                                                           */
/* Return value:                                                             */
/*  0  if o.k., <0 if error                                                  */
/*                                                                           */
/* Example:                                                                  */
/* Files:                                                                    */
/* Known bugs:                                                               */
/* Author: Arve Kylling                                                      */
/*                                                                 @i30_30@ */
/*****************************************************************************/
```

10 Overview of program flow

The *uvspec* main function is found in the file `/src/uvspec.c`. This function shows step by step, which routines are used in a *uvspec* calculation and where they can be found.

```
/*************************************************************/
/* The main uvspec function.                                 */
/*************************************************************/

int uvspec (input_struct input, output_struct *output)
{
  int status = 0;
  char function_name[]="uvspec";
  char file_name[]="uvspec.c";


  /**** Check the model input data ****/
  pmesg (" ... calling uvspec_check(), checking model input data\n", input.verbose);
  status = uvspec_check(input); /* in this file: uvspec.c */
  if (status!=0) {
    fprintf (stderr, "%d error(s) in uvspec input-file, aborting\n", abs(status));
    return status;
  }


  /**** Set wavelength grid for the transmittance calculation ****/
  pmesg (" ... calling setup_wlgrid(), generating model wavelength grid\n", input.verbose);
  status = setup_wlgrid (input, output);  /* in ancillary.c */
  if (status!=0) {
    fprintf (stderr, "Error %d setting up wavelength grid in %s (%s)\n", status, function_name, file_name);
    return status;
  }


  /**** Setup sza and phi0 ****/
  pmesg (" ... calling setup_sza(), generating model solar zenith and azimuth\n", input.verbose);
  status = setup_sza (input, output); /* in sza.c */
  if (status!=0) {
    fprintf (stderr, "Error %d setting up sza and phi0 in %s (%s)\n", status, function_name, file_name);
    return status;
  }


  /**** Independent pixel stuff ****/
  pmesg (" ... calling setup_ipa(), generating independent pixels\n", input.verbose);
  status = setup_ipa (input, output); /* in ipa.c*/
  if (status!=0) {
    fprintf (stderr, "Error %d setting up IPA properties in %s (%s)\n", status, function_name, file_name);
    return status;
  }


  /**** get surface altitude ****/
  pmesg (" ... calling setup_altitude(), determine surface elevation \n", input.verbose);
  status = setup_altitude (input, output); /* in atmosphere.c */
  if (status!=0) {
    fprintf (stderr, "Error %d determining surface altitude in %s (%s) \n", status, function_name, file_name);
    return status;
  }


  /**** Water cloud stuff (1D) ****/
  pmesg (" ... calling setup_wcloud(), generating water clouds\n", input.verbose);
  status = setup_wcloud (input, output); /* in cloud.c */
  if (status!=0) {
    fprintf (stderr, "Error %d setting up water cloud properties in %s (%s)\n", status, function_name, file_name);
    return status;
  }


  /**** Ice cloud stuff (1D) ****/
  pmesg (" ... calling setup_icloud(), generating ice clouds\n", input.verbose);
  status = setup_icloud (input, output); /* in cloud.c */
  if (status!=0) {
    fprintf (stderr, "Error %d setting up ice cloud properties in %s (%s)\n", status, function_name, file_name);
    return status;
  }


  /**** Cloudless sky atmosphere ****/
  pmesg (" ... calling setup_atmosphere(), generating model atmosphere\n", input.verbose);
  status = setup_atmosphere (input, output); /* in atmosphere.c */
  if (status!=0) {
    fprintf (stderr, "Error %d setting up atmosphere in %s (%s)\n", status, function_name, file_name);
    return status;
  }
```

```
/**** Setup temperature at computing levels ****/
pmesg (" ... calling setup_temperature(), generating model temperature profile\n", input.verbose);
status = setup_temperature (input, output); /* in atmosphere.c */
if (status!=0) {
  fprintf (stderr, "Error %d setting up temperature in %s (%s)\n", status, function_name, file_name);
  return status;
}


/**** Read cross section files, like O3, BrO, OclO, NO2, etc. ****/
pmesg (" ... calling setup_crs(), reading cross section files\n", input.verbose);
status = setup_crs (input, output); /* in molecular.c */
if (status!=0) {
  fprintf (stderr, "Error %d setting up absorption cross sections in %s (%s)\n", status, function_name, file_name);
  return status;
}


/**** Calculate Rayleigh scattering cross section ****/
pmesg (" ... calling setup_rayleigh(), calculating Rayleigh scattering\n", input.verbose);
status = setup_rayleigh (input, output); /* in molecular.c */
if (status!=0) {
  fprintf (stderr, "Error %d setting up Rayleigh cross sections in %s (%s)\n", status, function_name, file_name);
  return status;
}


/**** Calculate Raman scattering cross section ****/
pmesg (" ... calling setup_raman(), calculating Raman scattering\n", input.verbose);
status = setup_raman (input, output); /* in molecular.c */
if (status!=0) {
  fprintf (stderr, "Error %d setting up Raman cross sections in %s (%s)\n", status, function_name, file_name);
  return status;
}

/****  Optical properties of trace gases ****/
pmesg (" ... calling setup_gases(), generating optical properties of trace gases\n", input.verbose);
status = setup_gases (input, output);  /* in atmosphere.c */
if (status!=0) {
  fprintf (stderr, "Error %d setting up optical properties of trace gases in %s (%s)\n", status, function_name, file_name);
  return status;
}


/****   Aerosol stuff   ****/
pmesg (" ... calling setup_aerosols(), generating aerosols\n", input.verbose);
status = setup_aerosol (input, output); /* in aerosol.c */
if (status!=0) {
  fprintf (stderr, "Error %d setting up aerosol properties in %s (%s)\n", status, function_name, file_name);
  return status;
}


/**** read 3D water and ice clouds for MYSTIC ****/
pmesg (" ... calling setup_atmosphere3D(), generating 3D atmosphere\n", input.verbose);
status = setup_cloud3D (input, output); /* in cloud3d.c */
if (status!=0) {
  fprintf (stderr, "Error %d setting up 3D atmosphere in %s (%s)\n", status, function_name, file_name);
  return status;
}


/**** Redistribute profiles to required vertical resolution ****/
pmesg (" ... calling setup_redistribute(), redistributing vertical profiles\n", input.verbose);
status = setup_redistribute (input, output); /* in redistribute.c */
if (status!=0) {
  fprintf (stderr, "Error %d doing vertical redistribution of optical properties in %s (%s)\n", status, function_name, file_name);
  return status;
}


/**** 3D atmosphere for MYSTIC - to be called after redistribution! ****/
pmesg (" ... calling setup_atmosphere3D(), generating 3D atmosphere\n", input.verbose);
status = setup_atmosphere3D (input, output); /* in cloud3d.c */
if (status!=0) {
  fprintf (stderr, "Error %d setting up 3D atmosphere in %s (%s)\n", status, function_name, file_name);
  return status;
}


/****  Read extraterrestrial spectrum and correct for Earth-Sun distance ****/
pmesg (" ... calling setup_extraterrestrial(), reading extraterrestrial spectrum\n", input.verbose);
status = setup_extraterrestrial (input, output); /* in extraterrestrial.c */
if (status!=0) {
  fprintf (stderr, "Error %d setting up extraterrestrial irradiance in %s (%s)\n", status, function_name, file_name);
  return status;
}


/**** Setup wavelength dependent albedo ****/
pmesg (" ... calling setup_albedo(), generating surface albedo\n", input.verbose);
```

```
  status = setup_albedo (input, output); /* in albedo.c */
  if (status!=0) {
    fprintf (stderr, "Error %d setting up wavelength dependent albedo in %s (%s)\n", status, function_name, file_name);
    return status;
  }


  /****     Solve RTE     ****/
  pmesg (" ... calling solve_rte(), solving RTE\n", input.verbose);
  status = solve_rte (input, output); /* in solve_rte.c*/
  if (status!=0) {
    fprintf (stderr, "Error %d solving RTE in %s (%s)\n", status, function_name, file_name);
    return status;
  }


  /**** Interpolate transmittance to high resolution wavelength grid ****/
  pmesg (" ... calling interpolate_transmittance(), interpolating transmittance\n", input.verbose);
  status = interpolate_transmittance (input, output); /* in ancillary.c */
  if (status!=0) {
    fprintf (stderr, "Error %d interpolating model output to hires wavelength grid in %s (%s)\n", status, function_name, file_name);
    return status;
  }


  /**** Multiply with extraterrestrial irradiance ****/
  pmesg (" ... calling multiply_extraterrestrial()\n", input.verbose);
  status = multiply_extraterrestrial (input, output); /* in ancillary.c */
  if (status!=0) {
    fprintf (stderr, "Error %d multiplying with extraterrestrial irradiance in %s (%s)\n", status, function_name, file_name);
    return status;
  }


  /**** Convolve with slit function ****/
  if (input.convolve) {
    pmesg (" ... calling convolve(), convoluting with slit function\n", input.verbose);
    status = convolve (input, output); /* in ancillary.c */
    if (status!=0) {
      fprintf (stderr, "Error %d convolving hires spectrum with slit function in %s (%s)\n", status, function_name, file_name);
      return status;
    }
  }


  /**** Finally interpolate to output grid ****/
  if (input.spline) {
    pmesg (" ... calling spline_interpolate(), interpolating to output grid\n", input.verbose);
    status = spline_interpolate (input, output); /* in ancillary.c */
    if (status!=0) {
      fprintf (stderr, "Error %d interpolating spectrum to output grid in %s (%s)\n", status, function_name, file_name);
      return status;
    }
  }


  /**** Process 3D output (wavelength integration, ...) ****/
  if (output->mc.sample.passback3D) {
    pmesg (" ... calling processing3D(), post processing of 3D fields\n", input.verbose);
    status = processing3D (input, output); /* in ancillary.c */
    if (status!=0) {
      fprintf (stderr, "Error %d integrating 3D output over wavelength in %s (%s)\n", status, function_name, file_name);
      return status;
    }
  }


  /**** Process 1D output (wavelength integration, ...) ****/
  pmesg (" ... calling processing1D(), post processing\n", input.verbose);
  status = processing1D (input, output);   /* in ancillary.c */
  if (status!=0) {
    fprintf (stderr, "Error %d processing 1D output in %s (%s)\n", status, function_name, file_name);
    return status;
  }


  return 0;
}
```

References

Anderson, G., Clough, S., Kneizys, F., Chetwynd, J., and Shettle, E.: AFGL atmospheric constituent profiles (0-120 km), *Tech. Rep. AFGL-TR-86-0110*, Air Force Geophys. Lab., Hanscom Air Force Base, Bedford, Mass., 1986.

Baum, B., Heymsfield, A., Yang, P., and Bedka, S.: Bulk scattering models for the remote sensing of ice clouds. Part 1: Microphysical data and models, J. of Applied Meteorology, 44, 1885–1895, 2005a.

Baum, B., Yang, P., Heymsfield, A., Platnick, S., King, M., Hu, Y.-X., and Bedka, S.: Bulk scattering models for the remote sensing of ice clouds. Part 2: Narrowband models, J. of Applied Meteorology, 44, 1896–1911, 2005b.

Baum, B., Yang, P., Nasiri, S., Heidinger, A., Heymsfield, A., and Li, J.: Bulk scattering properties for the remote sensing of ice clouds. Part 3: High resolution spectral models from 100 to 3250 cm-1, J. of Applied Meteorology, 46, 423–434, 2007.

Buehler, S. A., Eriksson, P., Kuhn, T., von Engeln, A., and Verdes, C.: ARTS, the atmospheric radiative transfer simulator, Journal of Quantitative Spectroscopy and Radiative Transfer, 91, 65–93, 2005.

Cox, C. and Munk, W.: Measurement of the roughness of the sea surface from photographs of the sun's glitter, Journal of the Optical Society of America, 44, 838–850, 1954a.

Cox, C. and Munk, W.: Statistics of the sea surface derived from sun glitter, Journal of Marine Research, 13, 198–227, 1954b.

Dahlback, A. and Stamnes, K.: A new spherical model for computing the radiation field available for photolysis and heating at twilight, Planet. Space Sci., 39, 671–683, 1991.

Ebuchi, N. and Kizu, S.: Probability Distribution of Surface Wave Slope Derived Using Sun Glitter Images from Geostationary Meteorological Satellite and Surface Vector Winds from Scatterometers, J.Ocean., 58, 477–486, 2002.

Emde, C. and Mayer, B.: Simulation of solar radiation during a total solar eclipse: A challenge for radiative transfer, Atmos. Chem. Phys., 7, 2259–2270, 2007.

Emde, C., Hamann, U., Kylling, A., and Mayer, B.: Consolidation of a nominal set of requirements for libRadtran demonstration version, Tech. Rep. ESTEC Contract No AO/1-5433/07/NL/HE, Deutsches Zentrum für Luft- und Raumfahrt, Wessling, Germany, 2008.

Evans, K. F. and Stephens, G. L.: A new polarized atmospheric radiative transfer model, J. Quant. Spectrosc. Radiat. Transfer, 46, 413–423, 1991.

Fu, Q.: An accurate parameterization of the solar radiative properties of cirrus clouds for climate models, J. of Climate, 9, 2058–2082, 1996.

Fu, Q. and Liou, K.: On the correlated k-distribution method for radiative transfer in nonhomogeneous atmospheres, J. Atmos. Sci., 49, 2139–2156, 1992.

Fu, Q., Yang, P., and Sun, W. B.: An accurate parameterization of the infrared radiative properties of cirrus clouds for climate models, J. of Climate, 11, 2223–2237, 1998.

Hess, M., Koepke, P., and Schult, I.: Optical Properties of Aerosols and Clouds: The Software Package OPAC, Bulletin of the American Meteorological Society, 79, 831–844, 1998.

Hu, Y. X. and Stamnes, K.: An accurate parameterization of the radiative properties of water clouds suitable for use in climate models, J. of Climate, 6, 728–742, 1993.

Kato, S., Ackerman, T. P., Mather, J. H., and Clothiaux, E.: The k–distribution method and correlated–k approximation for a shortwave radiative transfer model, J. Quant. Spectrosc. Radiat. Transfer, 62, 109–121, 1999.

Key, J. R., Yang, P., Baum, B. A., and Nasiri, S. L.: Parameterization of shortwave ice cloud optical properties for various particle habits, J. Geophys. Res., 107, doi:10.1029/2001JD000 742, 2002.

Kratz, D. P. and Varanasi, P.: The correlated k-distribution technique as applied to the AVHRR channels, J. Quant. Spectrosc. Radiat. Transfer, 53, 501–517, 1995.

Kylling, A., Stamnes, K., and Tsay, S.-C.: A reliable and efficient two–stream algorithm for spherical radiative transfer: documentation of accuracy in realistic layered media , J. of Atmospheric Chemistry, 21, 115–150, 1995.

Mayer, B.: I3RC phase 1 results from the MYSTIC Monte Carlo model, in: Intercomparison of three-dimensional radiation codes: Abstracts of the first and second international workshops, University of Arizona Press, iSBN 0-9709609-0-5, 1999.

Mayer, B.: I3RC phase 1/2 results from the MYSTIC Monte Carlo model, in: Intercomparison of three-dimensional radiation codes: Abstracts of the first and second international workshops, edited by Cahalan, R. and Davis, R., pp. 49–54/107–108, Institute of Atmospheric Physics, University of Arizona, iSBN 0-9709609-0-5, 2000.

Mayer, B., Kylling, A., Hamann, U., and Emde, C.: libRadtran, library for radiative transfer calculations, 2007.

Rahman, H., Pinty, B., and Verstraete, M. M.: Coupled surface–atmosphere reflectance (CSAR) model 2. semiempirical surface model usable with NOAA advanced very high resolution radiometer data, J. Geophys. Res., 98, 20,791–20,801, 1993.

Ricchiazzi, P., Yang, S., Gautier, C., and Sowle, D.: SBDART: A research and Teaching software tool for plane-parallel radiative transfer in the Earth's atmosphere, Bulletin of the American Meteorological Society, 79, 2101–2114, 1998.

Shettle, E. P.: Optical and radiative properties of a desert aerosol model, Proceedings of the Symposium on Radiation in the Atmosphere, pp. 74–77, 1984.

Stamnes, K., Tsay, S., Wiscombe, W., and Jayaweera, K.: A numerically stable algorithm for discrete-ordinate-method radiative transfer in multiple scattering and emitting layered media, Applied Optics, 27, 2502–2509, 1988.

Stamnes, K., Tsay, S.-C., Wiscombe, W., and Laszlo, I.: DISORT, a General-Purpose Fortran Program for Discrete-Ordinate-Method Radiative Transfer in Scattering and Emitting Layered Media: Documentation of Methodology, Tech. rep., Dept. of Physics and Engineering Physics, Stevens Institute of Technology, Hoboken, NJ 07030, 2000.

Yang, P., Liou, K. N., Wyser, K., and Mitchell, D.: Parameterization of the scattering and absorption properties of individual ice crystals, J. Geophys. Res., 105, 4699–4718, 2000.