The arts_scat Python module

Cory Davis

January 29, 2004

1 The SingleScatteringData class

The main component of the arts_scat module is the SingleScatteringData class. The data members of this object are identical to the class of the same name in ARTS; it includes all the single scattering properties required for polarized radiative transfer calculations: the extinction matrix, the phase matrix, and the absorption coefficient vector. The angular, frequency, and temperature grids for which these are defined are also included. Another data member - "ptype", describes the orientational symmetry of the particle ensemble, which determines the format of the single scattering properties. The data structure of the ARTS SingleScatteringData class is described on pages 80-83 of the ARTS User Guide.

The methods in the arts_scat SingleScatteringData class enable the calculation of the single scattering properties, and the output of the SingleScatteringData structure in the ARTS XML format (see example file). The main methods are:-

- **calc()**: calculates the phase matrix, extinction matrix, and absorption coefficient vector
- **file_gen(filename)**: outputs the single scattering data to *filename* in ARTS XML format
- **generate()**: performs *calc()* and *file_gen* with a filename generated from particle parameters

A SingleScatteringData object is initialised with a dictionary of {"keyword":value} parameters. If this dictionary is omitted, or if any required parameters are

missing from the dictionary, default parameters are used. *This is a little dangerous - I might disable the defaults.* After initialisation these parameters (eg: "equiv_radius", "aspect_ratio",...) can be changed by modifying the **params** member dictionary, although I prefer to create a new SingleScatteringData object for each set of parameters.

The SingleScatteringData class also has a **load(filename)** method, which loads a SingleScatteringData object from an existing file. Note that this can only import data members that are actually in the file - so the scattering properties won't be consistint with the *params* data member. This method is useful when combined with the **combine** and **compress** functions described below.

Examples of Use

- Simple generation of single scattering data from a python interpreter session. First import the required modules

```
bash-2.05b$ python
Python 2.2.2 (#1, Feb 24 2003, 19:13:11)
[GCC 3.2.2 20030222 (Red Hat Linux 3.2.2-4)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from Numeric import * #import the numeric python module
>>> import arts_scat      #import the arts_scat module
```

  and define the necessary parameters

```
>>> scat_params={
... "NP":-1,    #particle type as in Mischenko's T-matrix code;
...                       #-1=spheroids,-2=cylinders
... 'ptype': 20,          #random orientation (30=horizontal)
... 'equiv_radius': 200,  #equivalent volume radius in microns
... 'aspect_ratio': 3,    #oblate (<1=prolate)
... "T_grid":[250],       #temperature is needed to calculate the
...                 #complex refractive index of ice. Currently only
...                 #one value is supported in ARTS.
... 'za_grid':arange(0,181,10),    #zenith angle grid
... 'aa_grid':arange(0,181,10),    #azimuth angle grid
... 'f_grid': [240e9, 242e9]       #frequency grid
... }
```

Now create a SingleScatteringData object, calculate scattering properties, and write an ARTS XML file.

```
>>> my_scat_data=arts_scat.SingleScatteringData(scat_params)
>>> my_scat_data.calc()
>>> my_scat_data.file_gen("an_example_file.xml")
```

Now try the *generate()* method and see where the data ended up

```
>>> my_scat_data.generate()
>>> print my_scat_data.filename
/home/cory/data/p20f240r200NP-1ar3.xml
```

2 Extras

The arts_scat module also includes some useful functions for manipulating and testing SingleScatteringData objects.

combine and compress

If for a given 1D profile you have more particle types than non-zero ice water content (IWC) levels, it makes sense to create one artificial SingleScattering-Data object for each of these levels. For example, in the JPL code there are 40 size-bins, and profile 2101 in the 1996 simulated data set, which is a very big cloud, there are only 7 non-zero IWC levels. In ARTS radiative transfer calculations, the reduction from 40 to 7 particle types gives a considerable reduction in memory use *and* CPU time. The *combine* and *compress* functions in the arts_scat module enable this.

Tests

The arts_scat module includes several tests, which, along with Pythons unittest module, can demonstrate that everything is working properly and also test the quality of the single scattering properties. The entire suite of tests can either be run from the python interpreter by typing "arts_scat.run_tests()", or from the shell with the command "python [python library path]/unittest.py arts_scat.test_suite".

3 Prerequisites

- **Python 2.?** This seems to come standard with Redhat Linux >8
- **Numeric python add-on (numpy)** http://sourceforge.net/projects/numpy
- **f2py** http://cens.ioc.ee/projects/f2py2e/ This is used to created compiled extension modules from fortran code based on Michael Mishchenko's T-matrix code.