Py6S: A Python interface to the 6S Radiative Transfer Model

R.T. Wilson

Geography and Environment, University of Southampton, Highfield Campus, Southampton, SO17 1BJ, UK

Keywords: radiative transfer model, 6S, atmospheric correction, python, scripting, methodology

1. Introduction

Radiative Transfer Models (RTMs) are widely used to simulate the passage of solar radiation through atmospheres on Earth and other planets. They have a range of uses including atmospheric research and solar energy system design, and are widely used within remote sensing and Earth observation, but they are often seen as difficult to use with respect to the numerous input and outputs parameters. This paper outlines Py6S, a Python interface to the 6S RTM (Vermote et al., 1997) designed to address these issues. Py6S allows the user to write simple Python scripts which can set 6S input variables, run simulations and access the outputs. Methods are also provided to perform common tasks, such as running a simulation for a range of wavelengths. It is envisaged that Py6S will provide a useful framework in which research on atmospheric radiative transfer can be conducted using 6S, as well as opening the use of 6S to a wider audience including students.

2. 6S

Second Simulation of the Satellite Signal in the Solar Spectrum (6S; Vermote et al., 1997) is a radiative transfer model which has established itself as one of the standard RTMs used for both remote sensing research and the creation of operational products. The model is intermediate in complexity, between simple RTMs such as SPCTRAL2 (Bird and Riordan, 1993) and FAR (Seidel et al., 2010), which do not produce results of the required accuracy for many applications, and very complex and computationally-intensive models such as SCIATRAN (Rozanov et al., 2005), libRADTRAN (Mayer and Kylling, 2005) and LIDORT/VLIDORT (Spurr, 2008). It has been used in the development of new algorithms and spectral indices (for example Ceccato et al., 2002) and is often combined with other models to produce fully-integrated models - for example it was used in the Kuusk and Nilson (2000) integrated forest reflectance model.

The current version is 6SV1.1, a vector version of the original 6S code which can simulate the atmospheric radiative transfer of polarised and non-polarised visible and infra-red radiation under different atmospheric conditions. Parameters include the atmospheric conditions, altitude of the sensor and target, wavelength and ground reflectance (with the ability to use a number of built-in BRDF models). An atmospheric correction mode allows the calculation of a ground reflectance, given an at-sensor radiance or reflectance value and a set of atmospheric parameters.

The primary simulation outputs are at-sensor reflectance and radiance, broken down into their individual components, as well as a number of other calculated atmospheric parameters. Validation has shown differences of less than 0.1% when compared with MODTRAN4 (Kotchenova et al., 2006; Kotchenova and Vermote, 2007), and Kotchenova et al. (2008) studied a number of standard RTMs and found that 6S demonstrated the best agreement with a Monte Carlo benchmark (within 1%). The ability of the latest version of 6S to take into account polarisation of light is thought to be behind its increased accuracy in many situations (Kotchenova et al., 2008).

6S is used operationally as part of the atmospheric correction procedure for Landsat TM (Ouaidrari and Vermote, 1999), and for generating lookup tables in the MODIS atmospheric correction procedure (Vermote and Vermeulen, 1999). 6S is also frequently used for atmospheric correction of images from a number of sensors by end-users (for example Alencar et al., 2011; Steven et al., 2003).

2.1. Limitations in the interface

The user interface to 6S is provided through text input and output files (see Listings 1 and 2) which have a number of issues:

- Every parameter in the input file is specified using a number, even categorical parameters, making the file difficult to read and edit.
- The input file must have exactly the correct format (including whitespace) if it is to be read correctly by 6S, thus any small errors can lead to problems ranging from software crashes to subtly incorrect outputs.
- It is only possible to specify one parameter set in the 6S input file, thus running 6S for a range of parameter values (for example multiple wavelengths, or multiple atmospheric conditions) requires manually editing the input file between each simulation.
- The format of the 6S output file is easy for humans to read, but hard to extract values from automatically.

Listing 1: Sample 6S input file

```
0 (User defined)
40.0 100.0 45.0 50.0 7 23 (geometrical conditions)
8 option for Water Vapor and Ozone
3.0 3.5 Water Vapor and Ozone
4 User's Components
0.25 0.25 0.25 0.25
0
0.5 value
-0.2  (target level, negative value)
-3.3 (sensor level)
-1.5 -3.5 (water vapor and ozone)
0.25 (aot)
11 (chosen band)
1 (Non homogeneous surface)
2 1 0.5 (ro1 ro2 radius)
1 BRDF
-0.1 radiance (positive value)
```

3. Py6S

Py6S is a Python interface to the 6S model which has been developed to address the limitations described above. By not re-implementing the model itself, we can ensure that results produced using Py6S will be exactly the same as results produced using 6S by itself, thus significantly reducing the amount of testing and validation required of the Py6S code.

3.1. Features

Py6S provides a superset of the 6S features: any parameter that can be set manually in a standard 6S input file can also be set through Py6S. Thus, the description of features here will not focus on the scientific features of the standard 6S model, but will focus on the improvements that Py6S provides.

- **User-friendly parameter setting, with easily-accessible documentation:** 6S parameters can be set using a simple Python interface rather than a cryptic input file. For example, to set the wavelength for the simulation to that of Landsat TM band 1, you simply run `s.wavelength = Wavelength(PredefinedWavelengths.Landsat_TM_B1)`. Extensive documentation is provided regarding the parameters which can be set, and this documentation can be accessed interactively through the Python interpreter.

- **Helper functions making common operations simple:** Manually running 6S simulations for many wavelengths across a certain wavelength range is a common need, but it is normally very time-consuming as it requires much manual editing of the 6S input files. In Py6S this can be accomplished with a single call to `run_vnir`, or `run_landsat_tm`. Similarly, running a simulation with many solar or view angles to produce a polar plot showing directional reflectance effects can be accomplished with a single call to `plot_and_run_360`.

- **Plotting capabilities:** Py6S links with the Matplotlib plotting library (Hunter, 2007), allowing the results from 6S simulations to be easily plotted using functions such as `plot_wavelengths` and `plot_360`.

Listing 2: Extract from a sample 6S output file

```
*******************************************************************************
*                                                                             *
*                          integrated values of  :                            *
*                          --------------------                               *
*                                                                             *
*       apparent reflectance  0.0330894  appar. rad.(w/m2/sr/mic)   12.749    *
*                   total gaseous transmittance  0.675                        *
*                                                                             *
*******************************************************************************
*                                                                             *
*                          coupling aerosol -wv  :                            *
*                          --------------------                               *
*           wv above aerosol :   0.033     wv mixed with aerosol :   0.033    *
*                       wv under aerosol :   0.033                            *
*******************************************************************************
*                                                                             *
*                          integrated values of  :                            *
*                          --------------------                               *
*                                                                             *
*       app. polarized refl.  0.0014    app. pol. rad. (w/m2/sr/mic)    0.065 *
*             direction of the plane of polarization-27.40                    *
*                   total polarization ratio     0.043                        *
*                                                                             *
*******************************************************************************
*                                                                             *
*                          int. normalized  values  of  :                     *
*                          ---------------------------                        *
*                       % of irradiance at ground level                       *
*     % of direct  irr.    % of diffuse irr.    % of enviro. irr              *
*               0.773               0.221               0.005                 *
*                        reflectance at satellite level                       *
*     atm. intrin. ref.   environment ref.  target reflectance                *
*               0.015               0.004               0.014                 *
*                                                                             *
*                          int. absolute values of                            *
*                          -----------------------                            *
*                       irr. at ground level (w/m2/mic)                       *
*     direct solar irr.    atm. diffuse irr.    environment  irr              *
*             453.572             127.136               3.157                 *
*                       rad at satel. level (w/m2/sr/mic)                     *
*     atm. intrin. rad.    environment rad.    target radiance                *
*               5.649               1.633               5.468                 *
*                                                                             *
*                                                                             *
*          int. funct filter (in mic)              int. sol. spect (in w/m2)  *
*             0.1174545                                185.589                *
*                                                                             *
*******************************************************************************

*******************************************************************************
*                                                                             *
*                           integrated values of  :                           *
*                           --------------------                              *
*                                                                             *
*                              downward        upward          total          *
*      global gas. trans. :     0.68965        0.97248        0.67513         *
*      water   "     "    :     0.98573        0.98623        0.97592         *
*      ozone   "     "    :     0.70609        0.99079        0.70008         *
*      co2     "     "    :     1.00000        1.00000        1.00000         *
*      oxyg    "     "    :     0.99344        0.99533        0.99179         *
*      no2     "     "    :     1.00000        1.00000        1.00000         *
*      ch4     "     "    :     1.00000        1.00000        1.00000         *
*      co      "     "    :     1.00000        1.00000        1.00000         *
*                                                                             *
*                                                                             *
*      rayl.  sca. trans. :     0.96494        0.93809        0.90520         *
*      aeros. sca.   "    :     0.72090        0.82111        0.59194         *
*      total  sca.   "    :     0.69208        0.81074        0.56110         *
*                                                                             *
*                                                                             *
*                                                                             *
*                              rayleigh       aerosols         total          *
*                                                                             *
*      spherical albedo   :     0.04939        0.04918        0.06820         *
*      optical depth total:     0.05550        0.42021        0.47570         *
*      optical depth plane:     0.01848        0.23606        0.25454         *
*      reflectance I      :     0.01098        0.01327        0.02175         *
*      reflectance Q      :     0.00118        0.00037        0.00122         *
*      reflectance U      :    -0.00156        0.00000       -0.00173         *
*      polarized reflect. :     0.00195        0.00037        0.00212         *
*      degree of polar.   :       17.77           2.76           9.75         *
*      dir. plane polar.  :      -26.48           0.00         -27.43         *
*      phase function I   :     1.26026        0.27565        0.39051         *
*      phase function Q   :    -0.21911       -0.00611       -0.03096         *
*      phase function U   :    -1.19913       -0.15957       -0.28084         *
*      primary deg. of pol:    -0.17386       -0.02215       -0.07927         *
*      sing. scat. albedo :     1.00000        0.52284        0.57850         *
*                                                                             *
*                                                                             *
*******************************************************************************
```

- **Access to all other Python functionality:** Py6S does not provide a Graphical User Interface (GUI) to 6S, as MODO (Schläpfer, 2001) does for MODTRAN4, but instead provides an API for the Python language, which allows a lot more flexibility. GUIs are easy to use for simple tasks, but can make it very difficult to perform more complicated tasks which the author may not have anticipated. By providing a Python API, code using Py6S can do anything that is possible within the Python language. For example, it can use all of the built-in functionality of the Python Standard Library, as well as access other Python modules commonly used in scientific computing (for example, `numpy`, `scipy`, `matplotlib` and `python-statslib`), allowing analysis of 6S outputs to be performed within the Python environment.

- **Ability to import parameters from external data sources:** Py6S allows detailed 6S parameterisation from real-world measurements. Currently supported sources are radiosonde data from the University of Wyoming Atmospheric Sciences department (available at `http://weather.uwyo.edu/upperair/sounding.html`) and sun photometry data from the AERONET network (Holben et al., 1998). Importing this data manually would require interpolation, unit conversion and date/time subsetting, all of which is done automatically by the Py6S functions.

- **Reproducibility:** There has been a increased emphasis recently on improving the reproducibility of research conducted using computational approaches in many fields (for example Baiocchi, 2007; Vandewalle et al., 2009). By allowing a whole series of 6S simulations to be run from a single script, and linking with other Python modules for further analysis, Py6S allows entire research projects using 6S to be reproducible from a single Python script.

3.2. Usage

3.2.1. Installation

As Py6S is purely an interface to 6S, the original 6S executable is required to run Py6S. Full instructions for 6S compilation on Windows, Mac OS X and Linux are included in the Py6S documentation. Py6S and its dependencies can be installed from the Python Package Index with a single call to the `pip` utility.

3.2.2. Example Py6S scripts

This section provides a number of examples to introduce Py6S functionality; the first example shows how to set a number of parameters through Py6S, run the simulation, and extract some outputs:

```
from Py6S import *
# Create an object to hold the 6S parameters
s = SixS()
# Set the atmospheric profile to Tropical
s.atmos_profile = AtmosProfile.PredefinedType(AtmosProfile.Tropical)
```

```
# Set the wavelength to 0.357um
s.wavelength = Wavelength(0.357)
# Run the model and print some outputs
s.run()
print s.outputs.pixel_radiance
print s.outputs.background_radiance
print s.outputs.single_scattering_albedo
print s.outputs.transmittance_water.downward
```

This may not seem significantly easier than writing 6S input files manually – although it is less error-prone – but another example shows the power of Py6S when running a number of simulations for many wavelengths and plotting the results, shown in Figure 1.

```
from Py6S import *
s = SixS()
# Run the 6S simulation defined by this SixS object across the
# whole VNIR wavelength range, extracting the pixel reflectance
# from the model output
wavelengths, results = SixSHelpers.Wavelengths.run_vnir(s,
        output_name="pixel_radiance")
# Plot these results, with the y axis label set to "Pixel Radiance"
SixSHelpers.Wavelengths.plot_wavelengths(wavelengths, results,
        r"At-sensor Spectral Radiance ($W/m^2\!/\mu m$)")
```

Similarly, simulations can be run for many angles, with the results plotted as a polar contour plot (shown in Figure 2), a time-consuming task which would require much editing of input files using the standard 6S interface, but which is far simpler in Py6S.

```
from Py6S import *
s = SixS()
# Set solar azimuth and zenith angles and wavelength
s.geometry.solar_a = 0
s.geometry.solar_z = 30
s.wavelength = Wavelength(0.550)
# Set the directional ground reflectance to be modeled
# by the Roujean BRDF model, using parameters for a pine forest
# (parameters taken from Roujean et al., 1992)
s.ground_reflectance = GroundReflectance.HomogeneousRoujean(0.037, 0.0, 0.133)
# Run the model and plot the results, varying the view angle (the other
# option is to vary the solar angle) and plotting the pixel radiance.
SixSHelpers.Angles.run_and_plot_360(s, 'view', 'pixel_radiance',
        colorbarlabel=r"At-sensor Spectral Radiance ($W/m^2\!/\mu m$)")
```

Further examples including importing of real-world data and more detailed parameterisations are provided in the documentation.

3.3. Design and Implementation

Py6S is a set of Python classes combined into the module `Py6S`. The main class is `SixS` which has attributes for setting parameters, and a `run` method

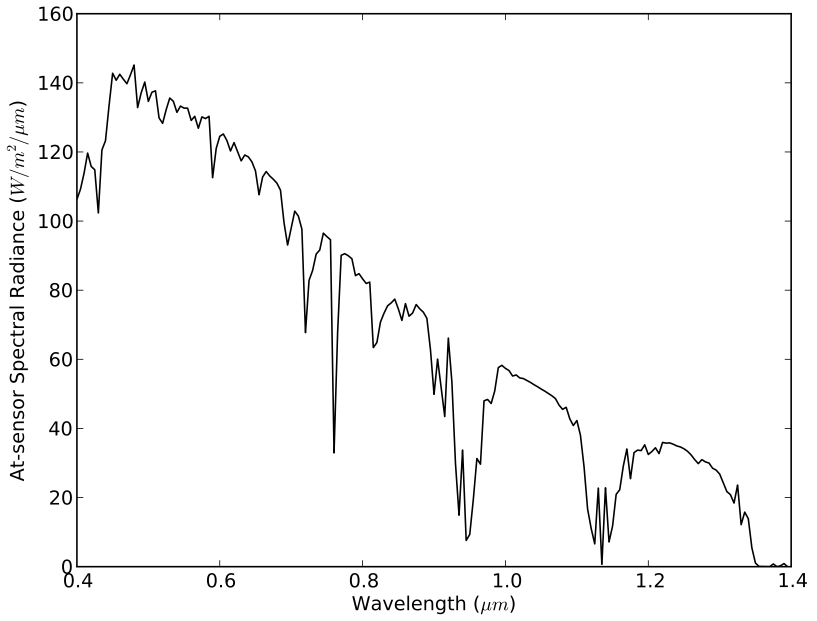

Figure 1: An example output from the Py6S commands `run_vnir` and `plot_wavelengths`.

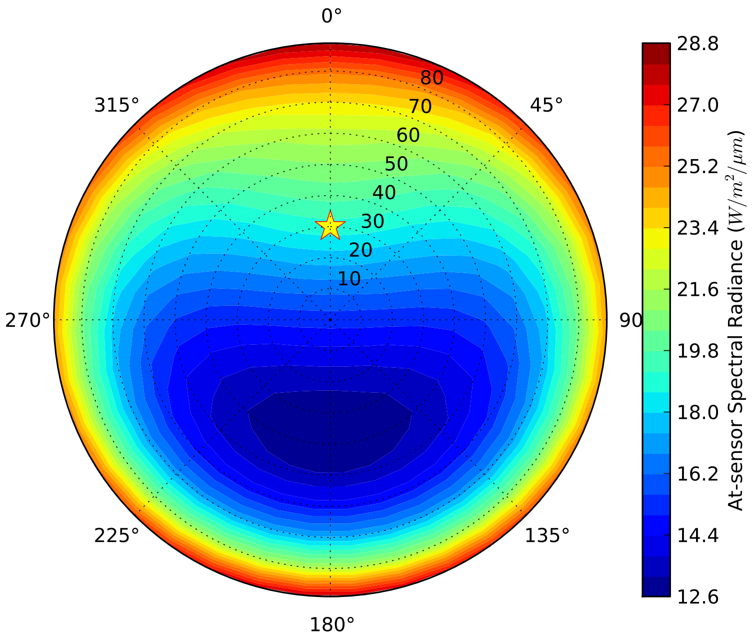

Figure 2: An example output from the Py6S command `run_and_plot_360`. This shows the radiance of a surface parameterised with the Roujean Bidirectional Reflectance Distribution Function model for a pine forest ($k_0 = 0.037$, $k_1 = 0$, $k_2 = 0.133$), at 500nm. Simulations were performed every 10 deg for both azimuth and zenith, and the yellow star denotes the location of the sun.

to run the model and parse the outputs. When this method is called, the parameters are written to a temporary input file and the standard 6S executable is then called on the input file. The full text output from 6S is captured and parsed to extract the numerical values of each output parameter which are then stored in the `values` dictionary.

Python's flexibility as a dynamically-typed language, allows functions to respond to a number of different types of input parameters. This has been used to provide a simple interface to the user, for example the `Wavelength` function can be called in any of the following manners:

```
# Wavelength of 0.43um
Wavelength(0.43)
# Band from 0.43-0.50um, with a flat response function of 1.0
Wavelength(0.43, 0.50)
# Band from 0.4-0.41um, with a custom response function
Wavelength(0.400, 0.410, [0.7, 0.9, 1.0, 0.3])
# A pre-defined sensor band wavelength
Wavelength(PredefinedWavelengths.LANDSAT_TM_B1)
```

This is far simpler to understand than a number of separate functions for setting different types of wavelength ranges. Similarly, although the output values are stored in a Python dictionary, they can be accessed as if they were attributes of the output class. For example:

```
# The standard way to access an item from a dictionary
s.outputs.values['pixel_reflectance']
# A cleaner and simpler way to access the output
s.outputs.pixel_reflectance
```

4. Conclusions

In conclusion, Py6S provides a modern environment for the scripting of 6S, a respected radiative transfer model in the remote-sensing community. Its features allow easy setting and modification of input variables and parameters and running of 6S simulations, it provide methods to import real-world measurements to 6S parameters, and it makes common operations such as running a simulation for all bands of a particular sensor easy. Py6S is released under the Lesser GNU Public License, and is available from `www.rtwilson.com/academic/py6s`.

5. Acknowledgements

This work was supported by an EPSRC Doctoral Training Centre grant (EP/G03690X/1). Thanks are due to EJ Milton and JM Nield for testing Py6S and providing advice on the writing of this paper and OE Wilson for further testing and advice on code design and structure.

6. References

Alencar, A., Asner, G., Knapp, D., Zarin, D., 2011. Temporal variability of forest fires in eastern Amazonia. Ecological Applications 21, 2397–2412.

Baiocchi, G., 2007. Reproducible research in computational economics: guidelines, integrated approaches, and open source software. Computational Economics 30, 19–40.

Bird, R., Riordan, C., 1993. Simple solar spectral model for direct and diffuse irradiance on horizontal and tilted planes at the earth's surface for cloudless atmospheres. SPIE milestone series 54, 171–181.

Ceccato, P., Gobron, N., Flasse, S., Pinty, B., Tarantola, S., 2002. Designing a spectral index to estimate vegetation water content from remote sensing data: Part 1: Theoretical approach. Remote Sensing of Environment 82, 188–197.

Holben, B., Eck, T., Slutsker, I., Tanre, D., Buis, J., Setzer, A., Vermote, E., Reagan, J., Kaufman, Y., Nakajima, T., et al., 1998. AERONET – a federated instrument network and data archive for aerosol characterization. Remote Sensing of Environment 66, 1–16.

Hunter, J.D., 2007. Matplotlib: A 2D Graphics Environment. Computing in Science and Engineering 9, 90–95.

Kotchenova, S., Vermote, E., 2007. Validation of a vector version of the 6S radiative transfer code for atmospheric correction of satellite data. Part II. Homogeneous Lambertian and anisotropic surfaces. Applied Optics 46, 4455–4464.

Kotchenova, S., Vermote, E., Levy, R., Lyapustin, A., 2008. Radiative transfer codes for atmospheric correction and aerosol retrieval: intercomparison study. Applied Optics 47, 2215–2226.

Kotchenova, S., Vermote, E., Matarrese, R., Klemm Jr, F., et al., 2006. Validation of a vector version of the 6S radiative transfer code for atmospheric correction of satellite data. Part I: Path radiance. Applied Optics 45, 6762–6774.

Kuusk, A., Nilson, T., 2000. A directional multispectral forest reflectance model. Remote Sensing of Environment 72, 244–252.

Mayer, B., Kylling, A., 2005. Technical note: The libRadtran software package for radiative transfer calculations? description and examples of use. Atmospheric Chemistry and Physics 5, 1319–1381.

Ouaidrari, H., Vermote, E., 1999. Operational atmospheric correction of Landsat TM data. Remote Sensing of Environment 70, 4–15.

Rozanov, A., Rozanov, V., Buchwitz, M., Kokhanovsky, A., Burrows, J., 2005. SCIATRAN 2.0 – A new radiative transfer model for geophysical applications in the 175–2400nm spectral region. Advances in Space Research 36, 1015–1019.

Schläpfer, D., 2001. MODO: an interface to MODTRAN for the simulation of imaging spectrometry at-sensor signals, in: 10th JPL Airborne Earth Science Workshop. JPL, Pasadena (CA), Vol. Publication, pp. 02–1.

Seidel, F., Kokhanovsky, A., Schaepman, M., 2010. Fast and simple model for atmospheric radiative transfer. Atmospheric Measurement Techniques 3, 1129–1141.

Spurr, R., 2008. LIDORT and VLIDORT: Linearized pseudo-spherical scalar and vector discrete ordinate radiative transfer models for use in remote sensing retrieval problems. Light Scattering Reviews 3 , 229–275.

Steven, M., Malthus, T., Baret, F., Xu, H., Chopping, M., 2003. Intercalibration of vegetation indices from different sensor systems. Remote Sensing of Environment 88, 412–422.

Vandewalle, P., Kovacevic, J., Vetterli, M., 2009. Reproducible research in signal processing. IEEE Signal Processing Magazine 26, 37–47.

Vermote, E., Tanré, D., Deuze, J., Herman, M., Morcette, J., 1997. Second simulation of the satellite signal in the solar spectrum, 6S: An overview. IEEE Transactions on Geoscience and Remote Sensing 35, 675–686.

Vermote, E., Vermeulen, A., 1999. Atmospheric correction algorithm: spectral reflectances (MOD09). Algorithm Theoretical Basis Document 4.