

---

# **sentinelSimulator Documentation**

***Release 0.0.1***

**E. Pinnington**

July 31, 2017



## CONTENTS

<b>1</b>	<b>Contents:</b>	<b>1</b>
1.1	sentinelSimulator . . . . .	1
1.2	sentinelSimulator package . . . . .	2
<b>2</b>	<b>Indices and tables</b>	<b>7</b>
	<b>Python Module Index</b>	<b>9</b>
	<b>Index</b>	<b>11</b>



## CONTENTS:

### 1.1 sentinelSimulator

This Python package simulates Sentinel 2 data based on output from the JULES land surface model. Here we provide documentation on this package. Below we illustrate a basic explanation of the package in use.

Example of using sentinelSimulator package:

```
import simulator
# Get your stuff done
simulator.do_example()
```

#### 1.1.1 Features

- Simulates sentinel 2 data from JULES output
- Add additional explanation

#### 1.1.2 Source Code

[github.com/example\\_user/sentinelSimulator](https://github.com/example_user/sentinelSimulator)

#### 1.1.3 Support

If you are having issues, please let us know. Contact: [ewan.pinnington@gmail.com](mailto:ewan.pinnington@gmail.com)

#### 1.1.4 License

Details of licensing information.

## 1.2 sentinelSimulator package

### 1.2.1 Subpackages

#### sentinelSimulator.jules package

##### Submodules

#### sentinelSimulator.jules.py\_importNML module

`sentinelSimulator.jules.py_importNML.importJulesNML(nml)`

Parse a JULES nml file and write it in the style used for the julesNML class.

**Parameters** `nml` (*str*) – JULES NML file name.

**Returns** None

#### sentinelSimulator.jules.py\_jules module

`sentinelSimulator.jules.py_jules.crop_run(sow_date=110, b=6.631, smwilt=0.1866, neff=0.00057)`

Function that runs JULES with crop model turned on and given user defined parameters at Wallerfing site. Output is saved in folder and file specified within function.

**Parameters**

- `sow_date` (*int*.) – Sow date, between 90 and 150.
- `b` (*float*.) – Brooks-Corey exponent factor.
- `smwilt` (*float*.) – Soil moisture wilting point.
- `neff` (*float*.) – Nitrogen use efficiency of crop (Vcmax).

**Returns** 'Done' to notify used JULES run has finished.

**Return type** `str`

**class** `sentinelSimulator.jules.py_jules.jules(jules_exe='/home/ij910917/jules/models/jules4.8/build/bin/jules.exe')`

Bases: `sentinelSimulator.jules.py_jules.julesAllNML`

Class to run JULES.

**Parameters** `jules_exe` (*str*) – location of JULES executable.

---

**Note:** You must have JULES installed on local system with a version of 4.8 or higher.

---

**runJules** ()

Write all NML files to disk. Run JULES in a subprocess. Check output for fatal errors.

**Returns** stdout and stderr output from JULES model run.

**Return type** `str`

**class** `sentinelSimulator.jules.py_jules.julesAllNML`

This class is populated by the contents of a module which contains templates of all the required JULES namelist files

**writeNML** ()

## sentinelSimulator.jules.py\_julesNML module

This module holds JULES namelist files. It has been automatically generated.

```
class sentinelSimulator.jules.py_julesNML.julesNML (template, filename)
    This is the base class for storing and writing JULES namelist files

    update (template)

    write ()
```

## Module contents

### 1.2.2 Submodules

#### 1.2.3 sentinelSimulator.opticalCanopyRT module

```
sentinelSimulator.opticalCanopyRT.canopyRTOptical (state, geom, resln=1.0)
```

A python wrapper to the SemiDiscrete optical canopy RT model of Nadine Gobron. Runs the model for the whole of its valid spectra range at a resolution set by resln.

#### Parameters

- **state** (*instance.*) – Instance of the stateVector class.
- **geom** (*instance.*) – Instance of the sensorGeomety class.
- **resln** (*float.*) – the spectral resolution in nm [optional].

**Returns** Instance of the spectra class.

**Return type** instance.

#### 1.2.4 sentinelSimulator.satelliteGeometry module

```
sentinelSimulator.satelliteGeometry.getSentinel2Geometry (startDateUTC, length-  
                                                           Days, lat, lon, alt=0.0,  
                                                           mission='Sentinel-  
                                                           2a', tle-  
                                                           File='./TLE/norad_resource_tle.txt')
```

Calculate approximate geometry for Sentinel overpasses. Approximate because it assumes maximum satellite elevation is the time at which target is imaged.

#### Parameters

- **startDateUTC** (*object*) – a datetime object specifying when to start prediction.
- **lengthDays** (*int*) – number of days over which to perform calculations.
- **lat** (*float*) – latitude of target.
- **lon** (*float*) – longitude of target.
- **alt** (*float*) – altitude of target (in km).
- **mission** (*str*) – mission name as in TLE file.
- **tleFile** (*str*) – TLE file.

**Returns** a python list containing instances of the sensorGeometry class arranged in date order.

**Return type** list

**class** `sentinelSimulator.satelliteGeometry.sensorGeometry`  
Class to hold sun-sensor geometry information.

**printGeom()**  
Prints currently specified class attributes.

## 1.2.5 sentinelSimulator.spectra module

**exception** `sentinelSimulator.spectra.UnknownFileType`  
Bases: `exceptions.Exception`

Exception class for unknown filetypes

`sentinelSimulator.spectra.convolve(s1orig, s2orig, resln=1.0, s2norm=True)`  
Convolve one spectra with another, for example to apply a band pass, or a spectral response function.

### Parameters

- **s1orig** (*object*) – A spectra object.
- **s2orig** (*float*) – A spectra object.
- **resln** (*float*) – The spectral resolution to use.
- **s2norm** (*bool*) – If True normalise the second spectra (e.g. to apply a spectra response function).

**Returns** Convolved spectra.

**Return type** object

`sentinelSimulator.spectra.sentinel2(s, mission='a')`

**class** `sentinelSimulator.spectra.spectra` (*fname=None, ftype='SVC', wavlCol=0, reflCol=1, hdrLines=1*)

Bases: `object`

Spectra class for sentinel simulator.

**interpolate** (*resltn=0.1*)  
Interpolate spectra to the given resolution. Overwrites exisiting data.

**Parameters** **resltn** (*float*) – resolution of the interpolation.

**loadCSV** (*f, wavlCol=0, reflCol=1, hdrLines=1*)  
Read in data from a standard CSV file object.

### Parameters

- **f** (*file*) – File object.
- **wavlCol** (*int*) – Column containing wavelengths.
- **reflCol** (*int*) – Column containing reflectance data.
- **hdrLines** (*int*) – Number of lines to skip at start of file.

**loadSVCSig** (*f*)  
Read in data from an SVC .sig ascii file.

**Parameters** **f** (*file*) – File object.



**loadSpectra** (*fname*, *wavlCol*=0, *reflCol*=1, *hdrLines*=1)

Load in the spectra from a given file using a method appropriate to the type of file.

---

**Note:** Current supported formats are:

SVC - SCV .sig ascii file

CSV - standard ascii comma seperated values

---

**Parameters**

- **fname** (*str*) – Valid filename containing spectra.
- **wavlCol** (*int*) – Column containing wavelengths.
- **reflCol** (*int*) – Column containing reflectance data.
- **hdrLines** (*int*) – Number of lines to skip at start of file.

**trim** (*wlmin*, *wlmax*)

Trim the spectra so it is between two specified wavelengths. Destroys the original data.

**Parameters**

- **wlmin** (*float*) – The lowest wavelength of the new spectra.
- **wlmax** (*float*) – The highest wavelength of the new spectra.

## 1.2.6 sentinelSimulator.stateVector module

`sentinelSimulator.stateVector.read` (*file\_format*='jules', *file\_str*=None, *year*=None)

Reads output data to a dictionary of state vectors indexed by time.

---

**Note:** This function requires sub-functions capable of reading specified file format.

---

**Parameters**

- **file\_format** – format of output to read.
- **file\_str** (*str*) – location of file.
- **year** (*int*) – year of data to extract, if equal to None whole time series extracted

**Returns** state dictionary.

**Return type** dict

`sentinelSimulator.stateVector.read_jules` (*nc\_file*=None, *year*=None)

Reads jules output from netCDF file and writes it to a dictionary indexed by date.

**Parameters**

- **nc\_file** (*str*) – location of nc\_file.
- **year** (*int*) – year of data to extract, if equal to None whole time series extracted.

**Returns** state dictionary.

**Return type** dict

**class** `sentinelSimulator.stateVector.stateVector`

Class to hold state vector data for optical and microwave canopy RT models.

## 1.2.7 Module contents

## INDICES AND TABLES

- *genindex*
- *modindex*
- *search*



**S**

sentinelSimulator, 6  
sentinelSimulator.jules, 3  
sentinelSimulator.jules.py\_importNML, 2  
sentinelSimulator.jules.py\_jules, 2  
sentinelSimulator.jules.py\_julesNML, 3  
sentinelSimulator.opticalCanopyRT, 3  
sentinelSimulator.satelliteGeometry, 3  
sentinelSimulator.spectra, 4  
sentinelSimulator.stateVector, 5



## C

canopyRTOptical() (in module sentinelSimulator.opticalCanopyRT), 3  
 convolve() (in module sentinelSimulator.spectra), 4  
 crop\_run() (in module sentinelSimulator.jules.py\_jules), 2

## G

getSentinel2Geometry() (in module sentinelSimulator.satelliteGeometry), 3

## I

importJulesNML() (in module sentinelSimulator.jules.py\_importNML), 2  
 interpolate() (sentinelSimulator.spectra.spectra method), 4

## J

jules (class in sentinelSimulator.jules.py\_jules), 2  
 julesAllNML (class in sentinelSimulator.jules.py\_jules), 2  
 julesNML (class in sentinelSimulator.jules.py\_julesNML), 3

## L

loadCSV() (sentinelSimulator.spectra.spectra method), 4  
 loadSpectra() (sentinelSimulator.spectra.spectra method), 4  
 loadSVCSig() (sentinelSimulator.spectra.spectra method), 4

## P

printGeom() (sentinelSimulator.satelliteGeometry.sensorGeometry method), 4

## R

read() (in module sentinelSimulator.stateVector), 5  
 read\_jules() (in module sentinelSimulator.stateVector), 5  
 runJules() (sentinelSimulator.jules.py\_jules.jules method), 2

## S

sensorGeometry (class in sentinelSimulator.satelliteGeometry), 4  
 sentinel2() (in module sentinelSimulator.spectra), 4  
 sentinelSimulator (module), 6  
 sentinelSimulator.jules (module), 3  
 sentinelSimulator.jules.py\_importNML (module), 2  
 sentinelSimulator.jules.py\_jules (module), 2  
 sentinelSimulator.jules.py\_julesNML (module), 3  
 sentinelSimulator.opticalCanopyRT (module), 3  
 sentinelSimulator.satelliteGeometry (module), 3  
 sentinelSimulator.spectra (module), 4  
 sentinelSimulator.stateVector (module), 5  
 spectra (class in sentinelSimulator.spectra), 4  
 stateVector (class in sentinelSimulator.stateVector), 5

## T

trim() (sentinelSimulator.spectra.spectra method), 5

## U

UnknownFileType, 4  
 update() (sentinelSimulator.jules.py\_julesNML.julesNML method), 3

## W

write() (sentinelSimulator.jules.py\_julesNML.julesNML method), 3  
 writeNML() (sentinelSimulator.jules.py\_jules.julesAllNML method), 2