**BioSNICAR v0.1: Instruction Manual**

***Dr Joseph Cook, University of Sheffield, 2017***

**Contents**

**1 Introduction**

**2 Creating new biological impurities**

**3 Water**

**4 Updating SNICAR model and driver**

**5 Running the Model**

**6 Usage guidelines**

1. **Introduction**

BioSNICAR refers to coupled bio-optical and radiative transfer models which, together, predict the spectral albedo of a volume of ice or snow. The physical properties of the snow, irradiance conditions and impurities can be defined by the user. The bio-optical scheme is a mixing model that takes user-defined concentrations of various algal pigments to determine the imaginary part of the refractive index for algal cells. This information is then used to determine the optical properties of the cells using Mie theory, given information about the cell size. The optical properties of the cells are then added to a lookup library that can be accessed by an adapted form of the radiative transfer model ‘SNICAR’ (Flanner et al., 2007).

This instruction manual explains how to interact with the various scripts available in the BioSNICAR repository to run model experiments. This is not exhaustive, and will describe only the basic model functionality to begin with, but this will be a live document that will be updated as the model is developed. Instructions are provided for creating new biological impurities, adding them to the lookup library, incorporating them into the RTM, and running the model. There is also a README available in the repository and the code is extensively commented.

The essential scripts are all written for Matlab. This includes the driver software and the radiative transfer code, which are all that are needed to run model experiments using our existing impurity lookup library. However, if new impurities are to be created additional scripts are used. A few of these are currently coded in Python, but will be translated for Matlab. All the necessary data files and scripts are included in this repository. The software versions used to write the codes were Matlab R2016b and Python 3.4 using the Anaconda 3.4.3 distribution (Windows 64bit).

***To run the model with the existing impurities, skip ahead to section 5.***

1. **Creating Biological Impurities**

The first stage in building a new biological impurity is to determine the types of pigments to be included, and in what mass fraction, in the cell. This information is used by the bio-optical model to generate the imaginary part of the refractive index for the cell. The relevant script is ‘bio-optical.py’. This code has several dependencies that must be saved in the working directory. These include the absorption data for each pigment and the python scripts ‘models.py’ and ‘cells.py’.

The pigment mass fractions for each pigment are defined in the following lines of code within BioOptical.py:

cell = cells.generate\_species('algae1', 300, 750)

cell = cells.fully\_pigmented\_cell\_generator(size=size, chlorophylla\_mf=0.015,

chlorophyllb\_mf=0.015,

chlorophylld\_mf=0.0,

chlorophyllf\_mf=0.0,

photoprotective\_carotenoids\_mf=0.05,

photosynthetic\_carotenoids\_mf=0.05,

phycocyanin\_mf=0.00,

phycoerythrin\_mf=0.0,

allophycocyanin\_mf=0.0,

min\_wl=min\_wl,

max\_wl=max\_wl

)

The pigment is expressed as a mass fraction of total cellular dry weight, for example if 10% of the total dry weight of the cell comprises chlorophyll a, then chlorophylla\_mf = 0.01. Running this code creates a csv file in the working directory called ‘KK.csv’ by default, but this can be changed by the user to something relevant to the specific application (e.g. ‘RI\_imag\_high\_pigment.csv’).

From this refractive index data, the optical properties of the cell can be calculated using Mie theory; however, the size distribution of the cells is also required. The particle size distribution is calculated using the Python script ‘Particle\_size\_SNICAR.py’. This can be used in either of two modes. First is for spherical cells, where the mean radius and standard deviation of the cells are prescribed by the user. This can be varied for experimentation or generated empirically from microscopic analyses of field samples. The second mode is for aspherical cells, where an effective spherical radius is calculated from the user defined length and breadth of the cells. Again, this can be derived from empirical work or varied for experimentation The code will return the geometric standard deviation, geometric mean and surface weighted radius.

The size and refractive index information can then be used to calculate the cell optical properties. This is achieved using two scripts which must be saved in the working directory: ‘Mie\_driver.m’ and ‘Mie.m’. Only Mie\_driver.m needs to be opened by the user. Also, the csv file saved by BioOptical.py (named ‘KK.csv’ by default) must be available in the workspace.

In Mie\_driver.m, the cell radius needs to be prescribed in line 23. Otherwise, the script can simply be run without modification, as long as the appropriate file containing the imaginary refractive index data is in the workspace. The optical properties and particle size distributions will then have been created ready to be compiled into a NetCDF file and added to the lookup library. This is achieved using another script, called ‘NetCDF\_updater.m’. The relevant optical parameters should already be available in the workspace, meaning no loading of files is required prior to running the updater script. The particle size information needs to be added manually to the updater. It is easiest to duplicate an existing NetCDF impurity file to use as a template. To do this, copy and paste ‘in place’ then rename an existing impurity file (e.g. duplicate ‘biological\_6.nc’ and rename it ‘biological\_7.nc’). Then use this filename in the updater. This ensures the format and file architecture are common to all the impurities. Saving the data as a NetCDF file in the working directory makes the data available to SNICAR. At this stage, the new impurity has been created and added to the lookup library ready to be integrated into the radiative transfer scheme.

The updater code is as follows:

% Code to take values from Mie solver and use it to update NetCDF file for

% ice, water or impurity

% Inputs: filename of destination NetCDF file required. All other variables

% should be populated and available in workspace after running Mie solver

% (i.e. mieIce.m, mieWATER.m or mie.m)

%

% Joseph Cook, Feb 2017, University of Sheffield, UK.

filename = 'biological\_1.nc';

rr(1:200) = 15e-6; %radius in m

ncwrite(filename,'rds',rr) % particle radius

ncwrite(filename,'ext\_xsc',ExtXC) % extinction cross section

ncwrite(filename,'sca\_xsc',ScaXC) % scattering cross section

ncwrite(filename,'abs\_xsc',AbsXC) % absorption cross section

ncwrite(filename,'ext\_cff\_mss',ExtXCmass) % mass extinction cross section

ncwrite(filename,'sca\_cff\_mss',ScaXCmass) % mass scattering cross section

ncwrite(filename,'abs\_cff\_mss',AbsXCmass) % mass absorption cross section

ncwrite(filename,'ext\_cff\_vlm',ExtXCvol) % volume extinction cross section

ncwrite(filename,'sca\_cff\_vlm',ScaXCvol) % volume scattering cross section

ncwrite(filename,'abs\_cff\_vlm',AbsXCvol) % volume absorption cross section

ncwrite(filename,'ss\_alb',ssa) % single scattering albedo

ncwrite(filename,'asm\_prm',asymmetry) % assymetry parameter

ncwrite(filename,'rds\_swa',1.0000007) % surface weighted radius (analytic)

ncwrite(filename,'rds\_swr',1.0000007) % surface weighted radius (resolved)

ncwrite(filename,'rds\_nma',15e-6) % analytic number-mean radius

ncwrite(filename,'gsd',0.00133) % geometric SD of lognormal distribution

ncwrite(filename,'prt\_dns',1400) % particle density

**3 Ice and Water**

Larger ice grains, spheres of water and water coatings around ice grains can be added to the model using optical properties and particle size distributions in the same way as for impurities, but with adapted codes. The refractive index for ice is from Warren and Brandt (2008) and the refractive index of water is from Segelstein (1981). These are available as datafiles in the repository and should be saved in the working directory to ensure they are available to be loaded into the Mie software.

For new ice grains, a new driver script called MieIce\_driver.m which calls to MieIce.m is required. The script works in the same way as for the optical properties of the biological impurities, but there are additional columns in the relevant NetCDF file. The NetCDF file naming protocol follows those of the default provided in SNICAR, with the grain size in microns at the end of the text.

The script NetCDF\_updater.m is then used to write the optical properties in to a NETCDF file that can be accessed by SNICAR. This is best achieved by copying an existing file for ice of different grain size, resaving with the new filename in the working directory, and updating the new file in place. That way, the file structure is definitely common to all the ice grain files.

Water can be included as spheres interspersed between solid ice grains. We include this method for completeness, as it has been described previously in the literature (e.g. Green et al., 2002); however, it is not recommended as it introduces new scattering surfaces into the medium and therefore probably does a poor job of simulating the effects of interstitial water. Nevertheless, the same procedure as above can be carried out using the script Mie\_WATER.m instead of Mie\_Ice.m.

Following the format of the rest of this document, two scripts are required for calculating optical parameters for coated spheres: Miecoated.m and Miecoated\_driver.m. The former is the actual Mie solver, which has been adapted from Matzler (2002). The driver extracts the real and imaginary refractive indices for ice and water per wavelength (from refracICE.m and refracWATER.m) and computes the associated size parameters for the inner and outer spheres. These are then used as inputs in a call to miecoated.m which returns the efficiency factors. These are then used to calculate cross sections according to the dimensions of the spheres. Importantly, the code can occasionally return NaNs at certain wavelengths, which are interpolated over using a cubic spline in the driver script. The reasons for this are unclear but have been documented before (e.g. by Matzler 2002). The affected wavelengths are only a few, concentrated into the mid NIR, and the interpolation results in very small errors.

Once run, the optical parameters are stored in a NetCDF file as for the homogenous ice crystals, named with the following filename convention:

Ice\_wrn\_coated\_1000100.nc

‘Ice’ is self explanatory, ‘wrn’ is to show that the refractive index of the ice is from Warren and Brandt (2008), ‘coated’ shows that it is a coated sphere. The number refers to the inner core (ice grain) radius and outer sphere (water) thickness in µm. For example, a 1000um ice grain with a 100 um water layer would be named ‘ice\_wrn\_coated\_1000100.nc’ This is because the SNICAR code requires a single integer to read in from the driver software (it is admittedly a little esoteric and should be modified at some point).

Some significant modification to SNICAR has been made to enable the reading in of coated spheres in place of homogenous spheres, including adding the variable rds\_coated() as an alterative to rds\_snw(), which is automatically called when rds\_snw() is set to zero. Whether rds\_coated() or rds\_snw() is used, the other must be set to zero. There is a small block of code in snicar.m that will read in the relevant file from the working directory based upon the number inserted into each layer in rds\_coated(), which can be found between lines 376-381.

**4 Updating SNICAR:**

When additions are made to the lookup library, the radiative transfer scheme SNICAR must be updated accordingly, so that when the new impurities are called by its driver software, it knows where to look and how to incorporate them into the radiative transfer calculations. Both the driver and the model code needs to be updated to account for new impurities. Here, the driver updates will be explained first, then the model updates.

NB. In both explanations below the symbol X is used in place of the impurity number. The line numbers are approximate as any changes to the script will shift the line numbers.

* + 1. **Update driver:**

line 95

change nbr\_aer = 8 to appropriate number of aerosols

line 106

add

mss\_cnc\_biox(1:nbr\_lyr) = [0,0,0,0,0];

line 117

add

fl\_biox = 'filename'

line 120

add

mss\_cnc\_bio1 and fl\_bio1 to function inputs

Once the driver software has been updated, the model itself needs to be modified so that it can read the new input format from the driver and incorporate it into the radiative transfer calculations.

* + 1. **Update SNICAR model:**

line 85

add mss\_cnc\_biox\_in

add fl\_biox\_in

line 122

change nbr\_aer = 8; to appropriate number of aerosols

line 133

add

mss\_cnc\_bio1(1:nbr\_lyr) = 0.0;

line 144

add

fl\_biox = 'filename'

line 190

add

mss\_cnc\_bioX = mss\_cnc\_bioX\_in;

line 199

add

fl\_bioX = fl\_bioX\_in;

line 380

add

fli\_inX = strcat(wkdir,fl\_bioX)

line 412

add

omega\_aer(:,X) = ncread(fl\_in8,'ss\_alb')

g\_aer(:,8) = ncread(fl\_in,'asm\_prm')

near line 420:

add

mss\_cnc\_aer(1:nbr\_lyr,X) = mss\_cnc\_bioX;

**4.2 Incoming Solar Radiation**

The model can also be updated to utilise solar irradiance from specific times and locations. By default, SNICAR uses incoming solar radiation modelled for a clear sky day at a mid-latitude site, modelled using the RTM of Zender et al (1997). To update this, the data in the text file mlw\_sfc\_flx\_frc\_clr.txt must be updated, or the code in SNICAR.m updated to read in data from a new text file (around lines 250-255).

Incoming spectral solar radiation can be measured using the ASD field spec with an upwards-looking cosine collector or a limited FOV fore-optic and a Spectralon reflectance panel, as the assumption that it is an isotropic, 100% reflective scattering surface means that there is no difference between incoming and reflected radiation from the panel (this is an approximation). Alternatively, the incoming solar radiation can be modelled using two well-known atmospheric radiative transfer schemes. Here, we demonstrate the use of COART (<https://cloudsgate2.larc.nasa.gov/jin/coart.html>) to obtain the necessary data. We here simulate incoming radiation for a cloud-free midsummer’s day at Kan-M. The input information used to generate our default incoming irradiance data are:

Calculation Type and output levels:

* Spectral fluxes (w/m2/um) at multiple wavelengths from **0.305**um to **5.0**um at every **0.005**um.
* Want to include the water leaving radiance output? **No**
* Output at: **Surface**

Solar zenith angle calculation or input:

* Julian Day **214** (1st August 2016) GMT: **1300** Latitude(degN): **67.0670** Longitude (degE): **48.8355**

Atmosphere:

* Select and atmospheric model: **Sub-Arctic Summer**
* If checked , use less atmospheric layers ti save computation time: **check**
* Select mixed layer aerosol: **OPAC Arctic** and stratospheric aerosol: **Background**
* By visibility (km): **23**
* Select Cloud: **No Cloud**

The output is saved in two locations. The first is called ‘COART\_default.csv’ and contains the entire output from the COART model run including the metadata. The second is called ‘Incoming\_KANM.csv’ and includes only the total downwelling solar energy per 5nm wavelength from 0.305 to 5um.

This incoming spectrum must be multiplied by the reflectance at each wavelength to determine albedo.

Two files must be made available to SNICAR: one for clear sky conditions for a particular site and one for cloudy conditions at a particular site (to enable the DIRECT = 1 or 0 function to work). This can be measured or modelled. The spectrum for incoming irradiance should be saved at a txt file in the working directory. In this README example, we name the file for the Greenland Kan-M AWS site.

File name (Clear): GRIS\_KM\_16Jul\_sfc\_flx\_frc\_clr.txt (=site\_date\_surface\_flux\_cloud\_fraction\_clear.txt)

File name (Cloud): GRIS\_KM\_16Jul\_sfc\_flx\_frc\_cld.txt (=site\_date\_surface\_flux\_cloud\_fraction\_clear.txt)

The following blocks of code would then need adding to the script snicar8d.m at lines 282 and 296 respectively:

Line 282:

% Kan-M Greenland, clear-sky:

load GRIS\_KM\_16Jul\_sfc\_flx\_frc\_clr.txt;

GRIS\_KM\_16Jul\_sfc\_flx\_frc\_clr(find(GRIS\_KM\_16Jul\_sfc\_flx\_frc\_clr==0))=1e-30;

flx\_slr = GRIS\_KM\_16Jul\_sfc\_flx\_frc\_clr;

Line 296

% KAN-M Greenland, cloudy:

load GRIS\_KM\_16Jul\_sfc\_flx\_frc\_cld.txt;

GRIS\_KM\_16Jul\_sfc\_flx\_frc\_cld(find(GRIS\_KM\_16Jul\_sfc\_flx\_frc\_cld==0))=1e-30;

flx\_slr = GRIS\_KM\_16Jul\_sfc\_flx\_frc\_cld;

**5 Running the Model**

Once the relevant impurities have been added to the lookup library and the driver and model have been updated accordingly, the model can be run. The model is controlled using the driver software ‘BIOSNICAR\_driver\_MASTER.m’. The model (snicar\_8d.m’) must be saved in the working directory but does not need to be open, there is no interaction with this file required.

In version 0.1 there are seven biological impurities and the option to include spheres of water, in addition to the default soot and dust impurities from the original SNICAR model.

The input parameters required to drive the model are as follows:

BND\_TYP: Spectral grid (=1 for 470 bands. This is the

only functional option in this distribution)

DIRECT: Direct or diffuse incident radiation (1=direct, 0=diffuse)

APRX\_TYP: Two-Stream Approximation Type:

1 = Eddington

2 = Quadrature

3 = Hemispheric Mean

NOTE: Delta-Eddington Approximation is probably

best for the visible spectrum, but can provide

negative albedo in the near-IR under diffuse

light conditions. Hence, the Hemispheric Mean

approximation is recommended for general use.

DELTA: 1=Use Delta approximation (Joseph, 1976), 0=don't use

coszen: cosine of solar zenith angle (only applies when DIRECT=1)

R\_sfc: broadband albedo of underlying surface

(user can also define a spectrally-dependent albedo below)

dz: array of snow layer thicknesses [meters]. Top layer has index 1.

The length of this array defines number of snow layers

rho\_snw: array of snow layer densities [kg m-3]. Must have same length as dz

rds\_snw: array of snow layer effective grain radii [microns]. Must have same length as dz

rds\_coated: radius of snow grains in each layer if water - coated grains are used (ignored if values are zero)

nbr\_aer: number of aerosol species in snowpack

mss\_cnc\_sot1: mass mixing ratio of black carbon species 1 (uncoated BC)

(units of parts per billion, ng g-1)

mss\_cnc\_sot2: mass mixing ratio of black carbon species 2 (sulfate-coated BC)

(units of parts per billion, ng g-1)

mss\_cnc\_dst1: mass mixing ratio of dust species 1 (radii of 0.05-0.5um)

(units of parts per billion, ng g-1)

mss\_cnc\_dst2: mass mixing ratio of dust species 2 (radii of 0.5-1.25um)

(units of parts per billion, ng g-1)

mss\_cnc\_dst3: mass mixing ratio of dust species 3 (radii of 1.25-2.5um)

(units of parts per billion, ng g-1)

mss\_cnc\_dst4: mass mixing ratio of dust species 4 (radii of 2.5-5.0um)

(units of parts per billion, ng g-1)

mss\_cnc\_ash1: mass mixing ratio of volcanic ash species 1

(units of parts per billion, ng g-1)

mss\_cnc\_bio1: mass mixing ratio of biological impurity species 1

(units of parts per billion, ng g-1)

mss\_cnc\_bio2: mass mixing ratio of biological impurity species 1

(units of parts per billion, ng g-1)

mss\_cnc\_bio3: mass mixing ratio of biological impurity species 1

(units of parts per billion, ng g-1)

mss\_cnc\_bio4: mass mixing ratio of biological impurity species 1

(units of parts per billion, ng g-1)

mss\_cnc\_bio5: mass mixing ratio of biological impurity species 1

(units of parts per billion, ng g-1)

mss\_cnc\_bio6: mass mixing ratio of biological impurity species 1

(units of parts per billion, ng g-1)

mss\_cnc\_bio7: mass mixing ratio of biological impurity species 1

(units of parts per billion, ng g-1)

mss\_cnc\_water1: mass mixing ratio of water type 1

(units of parts per billion, ng g-1)

fl\_sot1: name of file containing optical properties for BC species 1

fl\_sot2: name of file containing optical properties for BC species 2

fl\_dst1: name of file containing optical properties for dust species 1

fl\_dst2: name of file containing optical properties for dust species 2

fl\_dst3: name of file containing optical properties for dust species 3

fl\_dst4: name of file containing optical properties for dust species 4

fl\_ash1: name of file containing optical properties for ash species 1

fl\_bio1: name of file containing optical properties for bio species 1

fl\_bio2: name of file containing optical properties for bio species 2

fl\_bio3: name of file containing optical properties for bio species 3

fl\_bio4: name of file containing optical properties for bio species 4

fl\_bio5: name of file containing optical properties for bio species 5

fl\_bio6: name of file containing optical properties for bio species 6

fl\_bio7: name of file containing optical properties for bio species 7

fl\_water1: name of file containing optical properties for water type 1

To run the model, the user must add values for each of these parameters. For convenient experiments with impurity concentrations, the following grid format was created:

for x = [1 e6, 1.5e6, 2e6] % for reference: 1e6 = 1mg/g (1000000 ppb or 1000 ppm)

% PARTICLE MASS MIXING RATIOS (units: ng(species)/g(ice), or ppb)

mss\_cnc\_sot1(1:nbr\_lyr) = [0,0,0,0,0]; % uncoated black carbon

mss\_cnc\_sot2(1:nbr\_lyr) = [0,0,0,0,0]; % coated black carbon

mss\_cnc\_dst1(1:nbr\_lyr) = [0,0,0,0,0]; % dust species 1

mss\_cnc\_dst2(1:nbr\_lyr) = [0,0,0,0,0]; % dust species 2

mss\_cnc\_dst3(1:nbr\_lyr) = [0,0,0,0,0]; % dust species 3

mss\_cnc\_dst4(1:nbr\_lyr) = [0,0,0,0,0]; % dust species 4

mss\_cnc\_ash1(1:nbr\_lyr) = [0,0,0,0,0]; % volcanic ash species 1

mss\_cnc\_bio1(1:nbr\_lyr) = [0,0,0,0,0]; % Biological impurity species 1

mss\_cnc\_bio2(1:nbr\_lyr) = [0,0,0,0,0]; % Biological impurity species 2

mss\_cnc\_bio3(1:nbr\_lyr) = [0,0,0,0,0]; % Biological impurity species 3

mss\_cnc\_bio4(1:nbr\_lyr) = [0,0,0,0,0]; % Biological impurity species 4

mss\_cnc\_bio5(1:nbr\_lyr) = [0,0,0,0,0]; % Biological impurity species 5

mss\_cnc\_bio6(1:nbr\_lyr) = [0,0,0,0,0]; % Biological impurity species 6

mss\_cnc\_bio7(1:nbr\_lyr) = [0,0,0,0,0]; % Biological impurity species 7

mss\_cnc\_water1(1:nbr\_lyr) = [0,0,0,0,0]; % Water, 2 mm spheres

The array ( x = […] ) can be updated to include a range of impurity concentrations for the model to cycle through. Other values can also be added to the grid in place of zero for each impurity. Once these parameters values have been filled, the model can be run simply by pressing F5 or clicking the green ‘run’ arrow in the toolbar. The result is a plot of spectral albedo and a numeric value for broadband albedo printed in the command window.

**PLOTTING FIELD SPECTRA ALONGSIDE SNICAR SPECTRA**

This describes how to add a field spectrum to the SNICAR plot so that simulated and measured albedo can be compared. This is actually very simple, but since I spent an hour getting muddled with it in February, it is worth explaining here. Once SNICAR has been run using the driver routine, a plot is produced with at least one albedo spectrum displayed. For the 2017 TC paper and for other purposes it is useful to overlay real field spectra on the same plot. This is achieved in three simple steps: 1) load the field spectra into the workspace, 2) create an appropriately sized wavelength range for plotting the field spectra against, 3) plot the spectra on the existing axes. These will be explained in detail below:

1. The field spectra are saved in several formats – the most convenient to use is the .csv’s. The appropriate csv file can simply be drag and dropped from its file to the workspace in Matlab.
2. The wavelength range should be the same length as the field spectrum. The field spectrum was measured at 1nm spectral resolution between 350 and 2500 nm. It should be presented in microns. Therefore, the wavelength range to use is:

Wvlx = (0.35:0.001:2.5)

Length(wvlx) = 2151

1. Plotting the field spectrum is then straightforwards. Make sure the SNICAR plot is still open and if needs be, ‘hold on’ (shouldn’t be necessary). Then, simply plot wvlx against field spectrum.

**6 Usage Guidelines**

We welcome collaborations and contributions from other developers. Please direct queries to Joseph Cook (University of Sheffield) at the following email address: [joe.cook@sheffield.ac.uk](mailto:joe.cook@sheffield.ac.uk). If any publications use our model, please cite the paper Cook et al., (2017) Bioalbedo: a new physical model and critique of empirical methods for characterizing biological albedo reduction on ice and snow, *in review*, and also Flanner et al. (2007) (the source of the original SNICAR code).

This software is in active development. We provide free and open access to this software on the understanding that the authors assume no responsibility for downstream usage and, while we welcome collaboration, we are not obliged to provide training or support for users. We hope that this code and data will be useful and encourage collaboration, but we provide it without warranty or guarantee, nor the implied warranty of merchantability or fitness for any particular purpose. A particularly useful development would be a translation of the SNICAR model into Python.