SPARTACUS-Surface User Guide

Robin J. Hogan
European Centre for Medium Range Weather Forecasts, Reading, UK

Document version 0.7.3 (October 2021) applicable to *SPARTACUS-Surface* version 0.7.3*

1 Introduction

SPARTACUS-Surface is a Fortran-2003 software library for computing the 3D interaction of solar (or *shortwave*) and thermal-infrared (or *longwave*) radiation with complex surface canopies, especially forests and urban areas. It uses a multi-layer description of the canopy but with a statistical description of the horizontal distribution of trees and buildings. This greatly reduces the variables needed to describe the canopy, and makes the scheme fast enough to use in weather and climate models.

The detailed theoretical basis of the library is provided in two papers: Hogan et al. (2018) developed the shortwave forest solver, and Hogan (2019b) extended this to include buildings and longwave radiative transfer (these works developed two prototype codes in Matlab: *SPARTACUS-Vegetation* and *SPARTACUS-Urban*, both available from the SPARTACUS web site). The resulting algorithm combines three key ideas from earlier papers in the atmospheric radiative transfer literature:

- To represent horizontal variations in vegetation leaf density (or equivalently, extinction coefficient), each layer in a vegetation canopy is divided horizontally into three regions: clear-air (unvegetated) and two vegetated regions of equal fractional cover but different extinction coefficient. This approach was proposed by Shonk and Hogan (2008), who showed (in the context of cloudy radiative transfer) that the radiative effect of the full distribution of extinction coefficient could be approximated well given an appropriate choice for the extinction coefficients of the two regions.

- Three-dimensional radiative effects are treated rigorously by using the *Speedy Algorithm for Radiative Transfer through Cloud Sides* (SPARTACUS) of Hogan et al. (2016), but replacing clouds with trees and buildings. Since it is reasonable to treat trees and buildings as randomly distributed in the horizontal plane (see also Hogan, 2019a), the rate of exchange of radiation between the clear and vegetated parts of a layer may be assumed to be proportional to the length of the interface between them, and likewise for the rate of interception of radiation by building walls.

- The *Discrete Ordinate Method* is used to approximate the zenithal distribution of diffuse radiation, with the coupled ordinary differential equations solved by Eigen decomposition similarly to Stamnes et al. (1988). This is more robust than the matrix-exponential method used by Hogan et al. (2016), and more accurate since *SPARTACUS-Surface* allows the diffuse radiation field to be described by more than just two streams.

Section 2 describes how to compile and use the offline version of *SPARTACUS-Surface*, which is essentially a Unix program that reads a configuration file and a netCDF file containing a description of a number of surfaces, and outputs a netCDF file containing the computed radiation properties. Sections 3–7 describe how to configure and run the offline software, and the contents of the input and output data files. Section 8 outlines how to incorporate *SPARTACUS-Surface* into a larger Fortran program, such as an atmospheric model.

 If you have any queries about *SPARTACUS-Surface* that are not answered by this document then please email me at r.j.hogan@ecmwf.int.

2 Compiling the package

The offline version of *SPARTACUS-Surface* is designed to be used on a Unix-like platform. You will need a Fortran compiler that supports the 2003 standard, such as `gfortran`. As a prerequisite, you will need to install the netCDF library, including the Fortran interface (packages to install on a Linux system are typically called `libnetcdff-dev` or `libnetcdff-devel`). If you have a recent netCDF version then the command

```
nc-config --fc
```

should return the Fortran compiler for which the netCDF library was compiled. To run some of the tests, you will also need to install the NCO utilities for manipulating netCDF Files.

First unpack the package and enter the subdirectory as follows:

```
tar xvfz spartacus_surface-0.8.tar.gz
cd spartacus_surface-0.8
```

On a non-GNU platform you may need to untar and unzip the package using the `tar` and `gunzip` commands separately. The `README` file contains concise instructions on compilation and testing, while the `COPYING` file provides the license conditions. The subdirectories are as follows:

`radsurf` The *SPARTACUS-Surface* souce code for canopy radiative transfer

`radtool` Mathematical support routines for radiative transfer

`utilities` Source code for useful utilities, such as reading netCDF files

`driver` The source code for the offline driver program `spartacus_surface`

`mod` Where Fortran module files are written

`lib` Where the static libraries are written

`bin` Where the executable `spartacus_surface` is written

`test` Test cases including Matlab code to plot the outputs

`doc` The source for this document

Compilation on different platforms using different compilers is facilitated by the various `Makefile_include.<prof>` files in the top-level directory: if you type

```
make
```

or

```
make PROFILE=gfortran
```

the code will be compiled using the `gfortran` compiler via the Makefile variables set in the `Makefile_include.gfortran` file. Using instead `PROFILE=pgi` will use the `Makefile_include.pgi` file to attempt to compile with the PGI compiler, while `PROFILE=intel` selects the Intel compiler. If everything goes to plan this should create the executable `bin/spartacus_surface` and various static libraries in the `lib` directory.

One common reason the code doesn't compile out of the box is that it can't find the netCDF library files. The *SPARTACUS-Surface* Makefile uses the `nf-config` script that comes with recent versions of the netCDF library to create the Makefile variables `NETCDF_INCLUDE` and `NETCDF_LIB`. If `nf-config` is not available on your system, or it fails to correctly locate the netCDF library files, then the cleanest way to fix this is to create a `Makefile_include.local` file (starting from one of the existing `Makefile_include.*` files) that defines `NETCDF_INCLUDE` and `NETCDF_LIB` explicity to contain arguments for the compile and link operations, respectively. Suppose you installed netCDF in `/path/to/netcdf` and you use the `gfortran` compiler then your file might contain:

```
include Makefile_include.gfortran
NETCDF = /path/to/netcdf
NETCDF_INCLUDE = -I$(NETCDF)/include
NETCDF_LIB = -L$(NETCDF)/lib -lnetcdff -lnetcdf -Wl,-rpath,$(NETCDF)/lib
```

You should then be able to build the code with

```
make PROFILE=local
```

To compile with debugging options turned on (no optimization, bounds checking and initializing real numbers with not-a-number), type

```
make PROFILE=gfortran DEBUG=1
```

Finer tuning may be achieved by overriding the optimization and debugging flags used in `Makefile` explicitly, for example

```
make PROFILE=gfortran OPTFLAGS="-O1" DEBUGFLAGS="-g1 -pg"
```

Remember that if you change the compile settings you will probably want to recompile everything, in which case you first need to remove all compiled files with

```
make clean
```

3 Running the offline scheme

To test the code, type

```
make test
```

which runs `make` in each of the subdirectories of the `test` directory. The `README` files in these directories provide more information on what they are doing, and some Matlab scripts are provided to visualize the outputs.

You will see in the output of the tests the command line in each invocation of *SPARTACUS-Surface*, which is of the form

```
spartacus_surface config.nam input.nc output.nc
```

where `spartacus_surface` needs to be the full path to the *SPARTACUS-Surface* executable, `config.nam` is a Fortran namelist file configuring the code, `input.nc` contains the input atmospheric profiles and `output.nc` contains the output irradiance (flux) profiles. The namelist file contains a `radsurf` namelist that configures the *SPARTACUS-Surface* scheme itself; the parameters available are described in section 4. The file also contains a `radsurf_config` namelist that configures aspects of the offline package, described in section 5. Only the `radsurf` namelist is used when *SPARTACUS-Surface* is incorporated into an atmospheric model.

The input netCDF file contains numerous floating-point variables listed in Table 1. The dimensions are shown in the order that they are listed by the `ncdump` utility, with the first dimension varying slowest in the file (opposite to the Fortran convention). Most variables are stored as a function of column and layer (dimensions named `col` and `layer` in Table 1, although the actual dimension names are ignored by *SPARTACUS-Surface*). The `layer_int` dimension corresponds to interfaces between layers, plus the top-of-canopy and surface, and so must be one more than `layer`. Note that both `layer` and `layer_int` should increase upwards from the surface.

The optional `sw` and `lw` dimensions allow for shortwave and longwave optical properties of leaves and facets to be specified in user-defined spectral intervals. If these dimensions are omitted for these variables then constant optical properties will be assumed across the longwave and shortwave spectra.

Some variables can be omitted in which case default values will be used or these fields will be constructed according to `radsurf_config` namelist parameters (section 5).

Table 1: Main variables contained in the input netCDF file to *SPARTACUS-Surface*. All are floating-point numbers except for `surface_type` and `nlayer`, which contain integers.

Variable	Dimensions	Description
`cos_solar_zenith_angle`	`col`	Cosine of solar zenith angle
`surface_type`	`col`	Surface type: (0) flat, (1) forest, (2) urban, (3) vegetated urban, (4) simple urban, and (5) infinite street
`nlayer`	`col`	Number of active layers
`height`	`col, layer_int`	Height of layer interfaces (m)
`veg_fraction`	`col, layer`	Vegetation fraction
`veg_scale`	`col, layer`	Vegetation horizontal scale (m)
`veg_extinction`	`col, layer`	Wavelength-independent vegetation extinction coefficient (m^{-1})
`veg_fsd`	`col, layer`	Fractional standard deviation of vegetation extinction
`veg_contact_fraction`	`col, layer`	Fraction of building walls in contact with vegetation; note that in version 0.7.2 and earlier this was instead defined as the fraction of vegetation edge in contact with buildings
`building_fraction`	`col, layer`	Building fraction
`building_scale`	`col, layer`	Building horizontal scale (m)
`clear_air_temperature`	`col, layer`	Temperature of clear (unvegetated) part of layer (K)
`veg_temperature`	`col, layer`	Temperature of leaves (K)
`veg_air_temperature`	`col, layer`	Temperature of air in vegetated part of layer (K)
`air_temperature`	`col, layer`	Alternative way to specify `clear_air_temperature` and `veg_air_temperature` if the same (K)
`ground_temperature`	`col`	Ground temperature (K)
`roof_temperature`	`col, layer`	Temperature of the roofs at the top of the layer (K)
`wall_temperature`	`col, layer`	Wall temperature (K)
`ground_sw_albedo`	`col, sw`	Shortwave albedo of ground
`ground_sw_albedo_direct`	`col, sw`	Shortwave albedo of ground to direct beam (if different)
`ground_lw_emissivity`	`col, lw`	Longwave emissivity of ground
`veg_sw_ssa`	`col, layer, sw`	Shortwave single-scattering albedo of the leaves
`veg_lw_ssa`	`col, layer, lw`	Longwave single-scattering albedo of the leaves
`roof_sw_albedo`	`col, layer, sw`	Shortwave albedo of roofs
`roof_sw_albedo_direct`	`col, layer, sw`	Shortwave albedo of roofs to direct beam (if different)
`roof_lw_emissivity`	`col, layer, lw`	Longwave emissivity of roofs
`wall_sw_albedo`	`col, layer, sw`	Shortwave albedo of walls
`wall_sw_specular_fraction`	`col, layer, sw`	Fraction of wall reflection that is specular
`wall_lw_emissivity`	`col, layer, lw`	Longwave emissivity of walls
`sky_temperature`	`col`	Equivalent emitting temperature of sky (K)
`top_flux_dn_sw`	`col, sw`	Top-of-canopy downwelling shortwave flux ($W\ m^{-2}$)
`top_flux_dn_direct_sw`	`col, sw`	Top-of-canopy downwelling direct shortwave flux ($W\ m^{-2}$)

Input fields should be provided in order of increasing height, and the output data use the same convention. The `surface_type` variable selects how the column is to be treated, as depicted in Fig. 1. Types 1–3 use the SPARTACUS method as described by Hogan (2019b) but differ according to whether trees, buildings or both are represented. Types 4 and 5 use a simplified description of unvegetated urban areas in which all buildings are assumed to have the same height, and the Harman et al. (2004) method is used to represent multiple scattering within the urban canopy by solving a 2×2 matrix problem. Type 4 makes the same assumption about the horizontal distribution of buildings as SPARTACUS, which is that the wall-to-wall separation distances follow an exponential distribution (Hogan, 2019a). Type 5 makes the 'infinite street' assumption: streets are assumed to be of equal width and infinite in length. While the latter assumption is commonly used in urban exchange models, it was found by Hogan (2019a) to be a poorer fit to the building distributions in real cities.

The output netCDF file contains the typical set of broadband fluxes and absorption rates listed in Table 2. If you need them spectrally resolved (using the input spectral discretization) then set the `radsurf_config` namelist

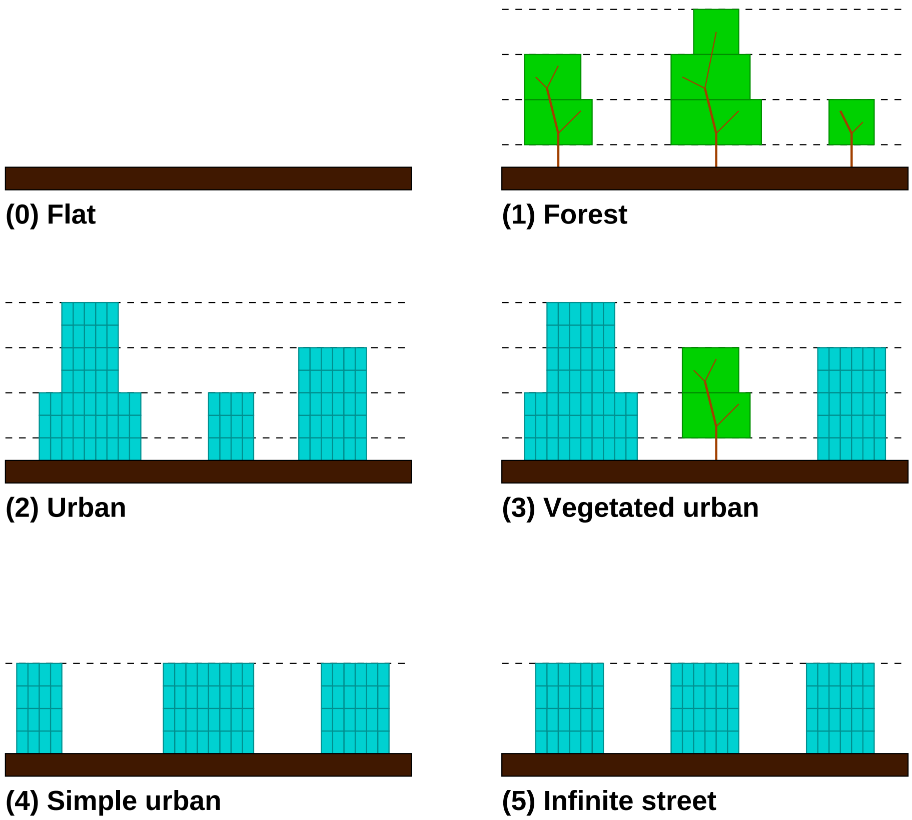

Figure 1: Schematic of the six surface types represented by the `surface_type` variable provided in the *SPARTACUS-Surface* input file (see Table 1).

variable `do_spectral` to `true` and the same variables will be output but with the prefix `spectral_`.

Table 2: Variables contained in the output netCDF file from *SPARTACUS-Surface*. All fluxes (or irradiances) and absorption rates have units of W m^{-2}, but note that this is power per unit area of the *entire domain*, not per unit area of a specific facet type. 'Net' fluxes are defined as the flux into a facet type (or downward) minus the flux out of a facet type (or upward).

Variable	Dimensions	Description
`height`	`col, layer_int`	Height of layer interfaces above ground (m)
`ground_flux_dn_sw`	`col`	Downwelling shortwave flux into the ground
`ground_flux_dn_direct_sw`	`col`	Direct downwelling shortwave flux into the ground
`ground_flux_net_sw`	`col`	Net shortwave flux into the ground
`ground_flux_dn_direct_sw`	`col`	Direct downwelling shortwave flux into the ground
`ground_flux_vertical_diffuse_sw`	`col`	Diffuse shortwave flux into a vertical surface at ground level
`top_flux_dn_sw`	`col`	Top-of-canopy downwelling shortwave flux
`top_flux_net_sw`	`col`	Top-of-canopy net shortwave flux
`top_flux_dn_direct_sw`	`col`	Top-of-canopy direct downwelling shortwave flux
`roof_flux_in_sw`	`col, layer`	Shortwave flux into roofs
`roof_flux_in_direct_sw`	`col, layer`	Direct shortwave flux into roofs
`roof_flux_net_sw`	`col, layer`	Net shortwave flux into roofs
`wall_flux_in_sw`	`col, layer`	Shortwave flux into walls
`wall_flux_in_direct_sw`	`col, layer`	Direct shortwave flux into walls
`wall_flux_net_sw`	`col, layer`	Net shortwave flux into walls
`clear_air_absorption_sw`	`col, layer`	Shortwave absorption rate in clear-air part of layer

veg_absorption_sw	col, layer	Shortwave absorption rate by leaves
veg_absorption_direct_sw	col, layer	Direct shortwave absorption rate by leaves
veg_air_absorption_sw	col, layer	Shortwave absorption rate by air in vegetated part of layer
ground_flux_dn_lw	col	Downwelling longwave flux into the ground
ground_flux_net_lw	col	Net longwave flux into the ground
ground_flux_vertical_lw	col	Longwave flux into a vertical surface at ground level
top_flux_dn_lw	col	Top-of-canopy donwelling longwave flux
top_flux_net_lw	col	Top-of-canopy net longwave flux
roof_flux_in_lw	col, layer	Longwave flux into roofs
roof_flux_net_lw	col, layer	Net longwave flux into roofs
wall_flux_in_lw	col, layer	Longwave flux into walls
wall_flux_net_lw	col, layer	Net flux into walls
clear_air_absorption_lw	col, layer	Net longwave absorption rate in clear-air part of layer
veg_absorption_lw	col, layer	Net longwave absorption rate by leaves
veg_air_absorption_lw	col, layer	Net longwave absorption rate by air in vegetated part of layer
ground_sunlit_fraction	col	Fraction of the ground area in direct sunlight
roof_sunlit_fraction	col, layer	Fraction of the roof area in direct sunlight
wall_sunlit_fraction	col, layer	Fraction of the wall area in direct sunlight
veg_sunlit_fraction	col, layer	Fraction of the one-sided leaf area that is in direct sunlight

The `ground_flux_vertical_*` variables are diffuse fluxes into a vertical plane at ground level, and are useful for computing thermal comfort indices and mean radiant temperature. If you want them at other heights in the urban canopy then they may be estimated from the wall incoming fluxes `wall_flux_in_*`. For example, the diffuse flux into a vertical plane averaged over layer i in the urban canopy may be computed from

$$V_i = \frac{2}{\pi}\frac{W_i}{L_i \Delta z_i}, \tag{1}$$

where W_i is the reported diffuse flux into walls in layer i (i.e. `wall_flux_in_lw` in the longwave and `wall_flux_in_sw - wall_flux_in_direct_sw` in the shortwave), L_i is the normalized wall perimeter length and Δz_i is the thickness of layer i. The factor of $2/\pi$ converts from fluxes into a wall that may be oriented in any azimuthal direction to fluxes into a vertical plane oriented in one direction.

Some urban models predict temperatures separately for sunlit and shaded roofs, walls and streets. Similarly, to compute photosynthesis rates more accurately, some vegetation models need to know the fraction of leaf area that is sunlit. To support such applications, Table 2 lists four `*_sunlit_fraction` variables quantifying the fraction of these surfaces that are in direct sunlight. Note that because walls are assumed to be vertical surfaces, the maximum wall fraction that can be sunlit is $1/2$. In the case of vegetation, the sunlit fraction refers to the fraction of one-sided leaf area, so can approach 1 in thin layers at the top of a vegetation canopy. These variables are accompanied by direct fluxes `ground_dn_direct_sw`, `roof_in_direct_sw`, `wall_in_direct_sw` and `veg_absorption_direct_sw`. Therefore sufficient information is provided to treat sunlit and shaded surfaces separately, assuming that diffuse radiation evenly illuminates all surfaces in a layer.

If you set namelist parameter `do_save_flux_profile` to `true`, then the flux-profile variables listed in Table 3 will also be output, which are averaged over both the clear and vegetated regions. Note that these variables are output at the top and base of each layer. In forests, the fluxes at the top of one layer will be equal to the fluxes at the base of the layer above, whereas in urban areas, since the fluxes are per unit horizontal area of the entire domain, these fluxes will not match when the building fraction decreases with height.

Table 3: Additional variables contained in the output netCDF file from *SPARTACUS-Surface* if the namelist parameter `do_save_flux_profile` is set to `true`. All fluxes have units of W m^{-2}, but note that this is power per unit area of the *entire domain*, not per unit area of the clear/vegetated part of the layer.

Variable	Dimensions	Description
`flux_dn_layer_top_sw`	`col, layer`	Downwelling shortwave flux in the clear/vegetated regions at the top of each layer
`flux_dn_direct_layer_top_sw`	`col, layer`	Direct downwelling shortwave flux in the clear/vegetated regions at the top of each layer
`flux_up_layer_top_sw`	`col, layer`	Upwelling shortwave flux in the clear/vegetated regions at the top of each layer
`flux_dn_layer_base_sw`	`col, layer`	Downwelling shortwave flux in the clear/vegetated regions at the base of each layer
`flux_dn_direct_layer_base_sw`	`col, layer`	Direct downwelling shortwave flux in the clear/vegetated regions at the base of each layer
`flux_up_layer_base_sw`	`col, layer`	Upwelling shortwave flux in the clear/vegetated regions at the base of each layer
`flux_dn_layer_top_lw`	`col, layer`	Downwelling longwave flux in the clear/vegetated regions at the top of each layer
`flux_up_layer_top_lw`	`col, layer`	Upwelling longwave flux in the clear/vegetated regions at the top of each layer
`flux_dn_layer_base_lw`	`col, layer`	Downwelling longwave flux in the clear/vegetated regions at the base of each layer
`flux_up_layer_base_lw`	`col, layer`	Upwelling longwave flux in the clear/vegetated regions at the base of each layer

4 Configuring the *SPARTACUS-Surface* algorithm

The detailed settings of *SPARTACUS-Surface* are configured using the `radsurf` namelist in the namelist file provided as the first command-line argument to the `spartacus_surface` executable. The available namelist parameters are listed in Table 4.

Table 4: Options for the `radsurf` namelist that configures *SPARTACUS-Surface* algorithm. The type of each parameter can be inferred from its name: logicals begin with `do_` or `use_`, integers start with `i_` or `n_`, strings end with `_name`, and all other parameters are real numbers.

Parameter	**Default**, other values	Description
General		
`do_sw`	**`true`**	Compute shortwave fluxes?
`do_lw`	**`true`**	Compute longwave fluxes?
`do_vegetation`	**`true`**	Will vegetation be represented?
`do_urban`	**`true`**	Will urban areas be represented?
`use_sw_direct_albedo`	**`false`**	Specify ground and roof albedos separately for direct solar radiation?
`min_vegetation_fraction`	$\mathbf{10^{-6}}$	Minimum area fraction below which a region is removed completely
`do_save_flux_profile`	**`false`**	Save the flux profile variables listed in Table 3
Options specific to forest tiles		
`n_vegetation_region_forest`	**1**, 2	Number of regions used to describe vegetation (2 needed for heterogeneity)

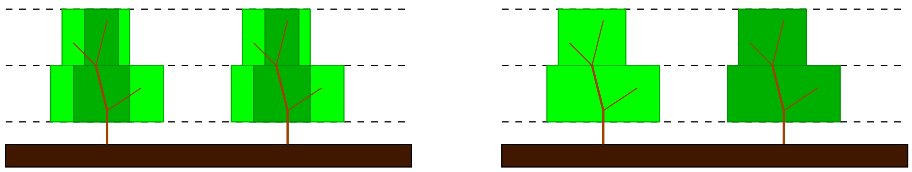

Figure 2: Schematic illustrating how the `vegetation_isolation_factor_*` parameters in Table 4 describe the contact between the clear-air region and the two vegetated regions in each layer.

`n_stream_sw_forest`	**4**	Streams per hemisphere to describe diffuse shortwave radiation
`n_stream_sw_forest`	**4**	Streams per hemisphere to describe longwave radiation
`use_symmetric_vegetation_scale_forest`	**true**	Compute vegetation perimeter length using Eq. 20 of Hogan et al. (2018)? Otherwise Eq. 19
`vegetation_isolation_factor_forest`	**0.0**, 0.0–1.0	In forests dense vegetation region is (0.0) embedded within sparse region or (1.0) in physically isolated regions, or in between
`vegetation_isolation_factor_urban`	**0.0**, 0.0–1.0	In urban areas dense vegetation region is (0.0) embedded within sparse region or (1.0) in physically isolated regions, or in between
Options specific to urban tiles		
`min_building_fraction`	$\mathbf{10^{-6}}$	Minimum building area fraction below which a building is ignored
`n_vegetation_region_urban`	**1**, 2	Number of regions used to describe vegetation (2 needed for heterogeneity)
`n_stream_sw_urban`	**4**	Streams per hemisphere to describe diffuse shortwave radiation
`n_stream_sw_urban`	**4**	Streams per hemisphere to describe longwave radiation
`use_symmetric_vegetation_scale_urban`	**true**	Compute vegetation perimeter length using Eq. 20 of Hogan et al. (2018)? Otherwise Eq. 19
`vegetation_isolation_factor_urban`	**0.0**, 0.0–1.0	Dense vegetation region is (0.0) embedded within sparse region or (1.0) in physically isolated regions

The number of streams can be any positive integer, but note that this is the number of quadrature points that the radiation field is divided into *per hemisphere*, so a value of 4 corresponds to an ‘8-stream scheme’ in the convention that all possible directions are considered. Larger values give more accuracy at greater computational cost, but as shown by Hogan (2019b), there is little change in the results above a value of 4.

The other parameters in Table 4 allow the treatment of vegetation to be fine tuned. If the number of vegetation regions is 1 then the trees are represented as in Fig. 1, with a single vegetation extinction coefficient per layer. In order to represent horizontal heterogeneity within tree crowns, a value of 2 should be selected, resulting in the representation in Fig. 2. The `vegetation_isolation_factor_*` parameters describe the contact between the clear region and the two vegetated regions in each layer, with the extremes being 0.0 in which the denser vegetation region is completely embedded in the sparser region (the assumption used by Hogan et al., 2018), and 1.0 in which the dense and sparse regions form unconnected tree crowns.

5 Configuring the offline package

In addition to the namelist parameters described in section 4 an additional set of parameters are available in the `radsurf_config` namelist that are specific to the offline version of *SPARTACUS-Surface* and are listed in Table 5. In general if these parameters are present in the namelist then they will override the corresponding variable provided in the input file.

Table 5: Options for the `radsurf_config` namelist that configures additional aspects of the offline radiation scheme. All entries must be scalars. If an override parameter is present then usually it need not be included in the input file.

Parameter	Description
Execution control	
`nrepeat`	Number of times to repeat, for benchmarking
`istartcol`	Start at specified input column (1 based)
`iendcol`	End at specified input column (1 based)
`iverbose`	Verbosity in offline setup (default 2)
`do_parallel`	Use OpenMP parallelism? (default `true`)
`nblocksize`	Number of columns per block when using OpenMP
`do_conservation_check`	Check final energy balance for each column
Override input variables	
`cos_solar_zenith_angle`	Override cosine of solar zenith angle
`isurfacetype`	Override value of surface type (0–5)
`ground_sw_albedo`	Override shortwave albedo of ground
`roof_sw_albedo`	Override shortwave albedo of roofs
`wall_sw_albedo`	Override shortwave albedo of walls
`ground_lw_emissivity`	Override longwave emissivity of ground
`roof_lw_emissivity`	Override longwave emissivity of roofs
`wall_lw_emissivity`	Override longwave emissivity of walls
`vegetation_fraction`	Override vegetation fraction
`vegetation_extinction`	Override vegetation extinction coefficient (m^{-1})
`vegetation_extinction_scaling`	Scale vegetation extinction coefficient
`vegetation_fsd`	Override vegetation fractional standard deviation of extinction
`vegetation_sw_ssa`	Override vegetation shortwave single scattering albedo
`vegetation_lw_ssa`	Override vegetation longwave single scattering albedo
`top_flux_dn_sw`	Override top-of-canopy downwelling shortwave flux ($W\ m^{-2}$)
`top_flux_dn_direct_sw`	Override top-of-canopy downwelling direct shortwave flux ($W\ m^{-2}$)
`top_flux_dn_lw`	Override top-of-canopy downwelling longwave flux ($W\ m^{-2}$)

6 Interpretation of geometry input variables

6.1 Forests

SPARTACUS-Surface assumes that the rate of lateral exchange of radiation between the clear-air and vegetated regions is proportional to the normalized length of the perimeter separating the vegetation and clear regions, L_{av}, i.e. the perimeter length divided by the horizontal area of the domain. This variable has units of m^{-1} and in practice is a strong function of vegetation fraction, so it is more convenient for models to parameterize the horizontal size of typical tree crowns, as expressed by the `vegetation_scale` input variable in Table 1. The `use_symmetric_vegetation_scale_*` parameters determine how this scale is used to compute L_{av} in the 'forest' and 'vegetated urban' surface types (shown in Fig. 1). If a symmetric vegetation scale is selected then Eq. 20 of Hogan et al. (2018) is used:

$$L_{av} = \frac{4av}{S_v}, \tag{2}$$

where a is the clear-air fraction, v is the vegetation fraction and S_v is the vegetation scale. In the case of a forest, there are no other regions so $a = 1 - v$ and as the vegetation fraction approaches one, the normalized perimeter approaches zero, indicating that the tree crowns effectively merge. If `use_symmetric_vegetation_scale_*` is `false` then Eq. 19 of Hogan et al. (2018) is used:

$$L_{av} = \frac{4v}{D_v}, \tag{3}$$

where this time `vegetation_scale` is interpreted as the effective crown diameter, D_v. This has the property that L_{av} approaches a constant value as the vegetation fraction approaches one, which could be thought of as the property of *crown shyness* exhibited by some forest canopies.

If you have two vegetated regions (`n_vegetation_region_*` in Table 4 is 2) then the normalized perimeter length between vegetation and clear-air, L_{av}, will be partitioned between the two vegetated regions in proportion to the relevant `vegetation_isolation_factor_*` parameter f:

$$L_{a1} = L_{av}(1 - f/2); \tag{4}$$

$$L_{a2} = L_{av} f/2, \tag{5}$$

where L_{a1} and L_{a2} are the normalized perimeter lengths between clear-air and each of the two vegetation regions. Figure 2 illustrates the limits of $f = 0$ and $f = 1$. This still leaves the length of the interface between the two vegetated regions, L_{12}, to be determined, which is used internally to compute the rate of lateral radiation exchange between these two regions. It is computed in *SPARTACUS-Surface* as follows. If the symmetric vegetation scale is selected then the horizontal scale of the vegetation inhomogeneities is assumed to be the same as the scale of the tree crowns themselves. Therefore, to compute the length of the interface between the two vegetated regions we use (2) but replace the vegetation fraction v with the fraction of just one of the two vegetation regions, $v/2$, such that

$$L_{12} = (1 - f)\frac{4(v/2)(1 - v/2)}{S_v} = (1 - f)\frac{v(2 - v)}{S_v}. \tag{6}$$

If the non-symmetric vegetation scale, D_v, is to be used then the following formula is applied instead

$$L_{12} = (1 - f)\frac{4v}{2^{1/2} D_v}. \tag{7}$$

The factor of $2^{1/2}$ on the denominator was used by Hogan et al. (2018) and is consistent with a tree model depicted in their Fig. 1 in which crowns are quasi-circular in horizontal cross-section and the optically thicker of the two region lies within a quasi-circular core surrounded by the optically thinner region.

If you are trying to simulate a 3D forest scene for which you know the exact location and sizes of the trees (e.g. from one of the RAMI intercomparisons) then you should analyze the scene to compute the vertical profile of L_{av}, then invert either (2) or (3) to obtain the profile of vegetation scale to use as input to the algorithm. The two formulas for L_{12} above do not allow for both L_{av} and L_{12} to be specified independently, even if you are analysing a 3D scene for which information on L_{12} might be known, but it should be noted that L_{av} has a much larger impact on radiative fluxes in the canopy than L_{12}.

6.2 Urban areas

The `building_scale` input variable in Table 1 is always interpreted as an effective building diameter, i.e. the code uses the equivalent of (3) to convert to the normalized building perimeter length in each layer, L_b, given the building fraction b:

$$L_b = \frac{4b}{D_b}. \tag{8}$$

If there is no vegetation then all the building perimeter length is in contact with clear air, so $L_{ab} = L_b$. If vegetation is present then the building perimeter L_b is divided between clear-air and vegetation according to the `veg_contact_fraction` variable, c:

$$L_{ab} = (1 - c)L_b; \tag{9}$$

$$L_{vb} = cL_b. \tag{10}$$

In the offline version of *SPARTACUS-Surface*, if `veg_contact_fraction` is not provided in the input file then it is assumed that trees are randomly located in the space between buildings, and therefore that the probability of a building wall being in contact with vegetation is equal to the fraction of this space containing vegetation, i.e.:

$$c = v/(a+v). \tag{11}$$

Note that the building, clear-air and vegetation fractions sum to one: $b+a+v = 1$. If *SPARTACUS-Surface* is embedded in a larger model then `veg_contact_fraction` must be provided explicitly, in which case it is recommended that (11) is used. Please note that in *SPARTACUS-Surface* version 0.7.2 and earlier, `veg_contact_fraction` was defined as the fraction of vegetation perimeter in contact with the walls, rather than the current definition which is the fraction of building perimeter in contact with vegetation.

The length of the interface between vegetation and clear-air is treated slightly differently for vegetated urban areas than forests, due to the fact that the building fraction b is removed from consideration. If a symmetric vegetation scale is selected then we apply (2) to the non-building part of the domain only (with fraction $a+v$):

$$\frac{L_{av}}{a+v} = \frac{4}{S_v}\frac{a}{a+v}\frac{v}{a+v}, \tag{12}$$

leading to

$$L_{av} = \frac{4}{S_v}\frac{av}{a+v}. \tag{13}$$

If a non-symmetric vegetation scale is selected then (3) is applied as before. If there are two vegetation regions in an urban area then the normalized perimeter length between clear-air and each of the two vegetation regions uses (4) and (5), as before. If the symmetric vegetation scale is selected then we take a similar approach and (6) becomes

$$L_{12} = (1-f)\frac{4(v/2)(1-v/2-b)}{(a+b)S_v}, \tag{14}$$

while for a non-symmetric vegetation scale, (7) is applied as before.

7 Checking the configuration

When `spartacus_surface` is run, it outputs to the screen a summary of the configuration options, with the level of detail controled by the `radsurf_config` namelist parameter `iverbose`. This can be used to check that *SPARTACUS-Surface* has been configured as intended. The following is an example from the default test in the `test/simple` directory, in the case of `iverbose=2`:

```
------------------ OFFLINE SPARTACUS-SURFACE RADIATION SCHEME ------------------
Copyright (C) 2019-2020 European Centre for Medium-Range Weather Forecasts
Contact: Robin Hogan (r.j.hogan@ecmwf.int)
Floating-point precision: double
General settings:
  Represent vegetation ON                  (do_vegetation=T)
  Represent urban areas ON                 (do_urban=T)
  Do shortwave (SW) calculations ON        (do_sw=T)
  Do longwave (LW) calculations ON         (do_sw=T)
  Number of SW spectral intervals = 1      (nsw)
  Number of LW spectral intervals = 1      (nlw)
  Minimum vegetation fraction = .100E-05   (min_vegetation_fraction)
Settings for forests:
  Number of vegetation regions = 2         (n_vegetation_region_forest)
  Use symmetric vegetation scale ON        (use_symmetric_vegetation_scale_forest=T)
  Vegetation isolation factor = 0.00       (vegetation_isolation_factor_forest)
  SW diffuse streams per hemisphere = 2    (n_stream_sw_forest)
  LW streams per hemisphere = 2            (n_stream_lw_forest)
Settings for urban areas:
  Number of vegetation regions = 2         (n_vegetation_region_urban)
  Use symmetric vegetation scale ON        (use_symmetric_vegetation_scale_urban=T)
  Vegetation isolation factor = 0.00       (vegetation_isolation_factor_urban)
```

```
  SW diffuse streams per hemisphere = 2   (n_stream_sw_urban)
  LW streams per hemisphere = 2           (n_stream_lw_urban)
Reading NetCDF file test_surfaces_in.nc
  Overriding cosine of the solar zenith angle with  0.500
  Overriding vegetation extinction with  0.250
  Setting temperature of clear-air and air in vegetation to air_temperature
  Setting vegetation temperature equal to air temperature
  Assuming roof albedo to direct albedo is the same as to diffuse
  Assuming wall reflection is Lambertian (no specular component)
    1: Flat
    2: Forest,            2 layers, 2 diffuse streams per hemisphere, 3 regions
    3: Unvegetated urban, 2 layers, 2 diffuse streams per hemisphere, 1 region
    4: Vegetated urban,   2 layers, 2 diffuse streams per hemisphere, 3 regions
Direct shortwave budget
Layer   Ground      Air     Wall     Roof      Veg  Air-veg      Top   Residual
    1  320.000    0.000    0.000    0.000    0.000    0.000  320.000  0.000E+00
    2   87.441    0.000    0.000    0.000  293.893    0.000  381.334  0.568E-13
    3   51.015    0.000  185.652  119.081    0.000    0.000  355.748 -0.568E-13
    4   25.686    0.000  163.389  119.057   51.620    0.000  359.752  0.000E+00
Diffuse shortwave budget
Layer   Ground      Air     Wall     Roof      Veg  Air-veg      Top   Residual
    1   80.000    0.000    0.000    0.000    0.000    0.000   80.000  0.000E+00
    2   27.565    0.000    0.000    0.000   67.146    0.000   94.710  0.000E+00
    3   20.203    0.000   37.465   30.846    0.000    0.000   88.514  0.000E+00
    4   13.199    0.000   33.583   30.840   12.105    0.000   89.726 -0.142E-13
Internal longwave budget
Layer   Ground      Air     Wall     Roof      Veg  Air-veg      Top   Residual
    1 -328.035    0.000    0.000    0.000    0.000    0.000 -328.035  0.000E+00
    2 -327.647    0.031    0.000    0.000  216.819    0.011 -110.785  0.847E-05
    3  -80.093   -0.158 -138.486 -125.417    0.000    0.000 -344.211  0.569E-01
    4  -55.546   -0.151 -130.312 -125.423  -32.158   -0.002 -343.627  0.351E-01
Incoming longwave budget
Layer   Ground      Air     Wall     Roof      Veg  Air-veg      Top   Residual
    1  263.855    0.000    0.000    0.000    0.000    0.000  263.855  0.000E+00
    2   89.108    0.029    0.000    0.000  198.716    0.010  287.868 -0.416E-02
    3   64.432    0.157  111.365  100.876    0.000    0.000  276.877 -0.462E-01
    4   41.886    0.146  101.133  100.871   34.728    0.002  278.796 -0.287E-01
Writing NetCDF file test_surfaces_out.nc
--------------------------------------------------------------------------------
```

8 Incorporating *SPARTACUS-Surface* into another program

SPARTACUS-Surface is primarily a software library that is designed to be called from within a larger program such as an atmospheric model. The library is written entirely using Fortran modules, and you will need all the Fortran source files in the `utilities`, `radtool` and `radsurf` directories. The `utilities/radiation_io.F90` file provides the `nulout` unit for logging messages and the `radiation_abort` routine for exiting if an error occurs. This file will likely need to be rewritten for your model, for example to ensure appropriate clean-up after an anomalous abort.

SPARTACUS-Surface adopts a strict policy of no global variables and no variables in modules (which are really a type of global variable). Therefore, information is passed to and from routines only via the arguments to those routines. This includes configuration information, which is stored in a `config_type` object from the `radsurf_config` module. As part of the initialization stage of the atmospheric model, such an object should be created, and will later be passed to the routine that performs the radiative transfer. A configuration object is created as follows:

```fortran
use radsurf_config, only : config_type ! Read module defining configuration type
type(config_type) :: config            ! Create a configuration object
! ...optionally set some default values - see radsurf/radsurf_config.F90...
call config%read(namelist_file_name)   ! Read configuration information from a namelist file
call config%consolidate()              ! Perform any additional configuration steps needed
```

Performing radiative transfer on a set of `ncol` surface canopy profiles is carried out in two parts. In the first part, the geometric and spectral properties of the canopies are used to compute (1) the top-of-canopy properties presented to the atmosphere above such as spectral albedo and upward longwave emission, and (2) fluxes within the canopy that are normalized with respect to downwelling shortwave and longwave fluxes at the top-of-canopy. It is envisaged that after this step, the top-of-canopy properties presented to the atmosphere are used as boundary conditions for a full atmospheric radiative transfer calculation. One of the outputs of such a calculation is the downwelling spectral shortwave and longwave fluxes at top-of-canopy. The second (much simpler) part of the interface to *SPARTACUS-Surface* involves scaling the normalized fluxes computed in the first step to obtain the absolute fluxes within the canopy, including net fluxes into ground, roofs, walls and vegetation. These fluxes and heating rates can then be used by a canopy energy balance model.

The first part involves calling the `radsurf` routine, which takes as arguments a number of objects. The input description of the canopy is in the form of three Fortran derived types: the `canopy_properties_type` describes the wavelength-independent properties of the canopy, and the `sw_spectral_properties_type` and `lw_spectral_properties_type` describe the shortwave and longwave spectral properties. Each object describes `ncol` surface 'columns' to be treated independently; each column could correspond to an atmospheric model column, or alternatively may represent multiply 'tiles' underlying one or more atmospheric columns.

The arrays in the `canopy_properties_type` could have been dimensioned `nmaxlay`×`ncol`, where `nmaxlay` is the maximum number of layers in any of the individual columns. However, it is recognised that in a global model many or even most of the columns would be treated as 'flat' (type 0 in Fig. 1) and many layers would be unused. Therefore the arrays in these objects use a 'packed' representation, explained by considering some of the elements of `canopy_properties_type`:

```
! Integers
ncol                ! Number of columns
ntotlay             ! Total number of layers

! Allocatable integer vectors of length "ncol"
nlay                ! Number of layers in column (can be 0)
istartlay           ! Index to first layer of the column
i_representation    ! Surface type (0-5)

! Allocatable real vectors of length "ncol"
cos_sza, ground_temperature

! Allocatable real vectors of length "ntotlay"
roof_temperature, wall_temperature, building_fraction, veg_fraction...
```

It can be seen that the variables describing properties as a function of height (e.g. wall temperature) are vectors of dimension `ntotlay`, expressing the total number of layers in this block of columns. The integer vectors `nlay` and `istartlay` enable the range of elements corresponding to the layers of a particular column to be identified. The `sw_spectral_properties_type` and `lw_spectral_properties_type` are packed similarly, except that each array has an additional `nspec` dimension expressing the number of shortwave or longwave spectral intervals.

Preparation of these three objects can be done as follows:

```
! Read modules defining the three derived types
use radsurf_canopy_properties,      only : canopy_properties_type
use radsurf_sw_spectral_properties, only : sw_spectral_properties_type
use radsurf_lw_spectral_properties, only : lw_spectral_properties_type

! Declare instances of these types
type(canopy_properties_type)      :: canopy_props
type(sw_spectral_properties_type) :: sw_spectral_props
type(lw_spectral_properties_type) :: lw_spectral_props

! Allocate canopy properties given existing configuration object "config", number of columns
! "ncol", total number of layers "ntotlay" and number of shortwave and longwave spectral
! intervals "nsw" and "nlw":
call       canopy_props%allocate(config, ncol, ntotlay)
call sw_spectral_props%allocate(config, ncol, ntotlay, nsw)
```

```
call lw_spectral_props%allocate(config, ncol, ntotlay, nlw)
```

Subsequent code would then populate the arrays within these three objects using data from the host model (consult the files `radsurf_canopy_properties.F90`, `radsurf_sw_spectral_properties.F90` and `radsurf_lw_spectral_properties.F90` for precise contents of these arrays). Note that the offline driver program `driver/spartacus_surface_driver.F90` does not use these `allocate` type-bound procedures, but rather populates the arrays within the objects directly using the contents of the input netCDF file.

We also need to prepare objects to hold the output from the `radsurf` routine:

```
! Read modules defining the relevant derived types
use radsurf_boundary_conds_out,     only : boundary_conds_out_type
use radsurf_canopy_flux,            only : canopy_flux_type

! Declare an object holding the top-of-canopy boundary conditions presented to the atmosphere
! above
type(boundary_conds_out_type) :: bc_out
! Declare canopy flux components: the first three contain the fluxes and net absorption rates
! within the canopy normalized by the shortwave-diffuse, shortwave-direct and longwave
! downwelling flux at top-of-canopy. The fourth contains the same but purely due to internal
! longwave emission within the canopy
type(canopy_flux_type)        :: sw_norm_diff, sw_norm_dir, lw_norm, lw_internal

! Allocate these objects, noting that the shortwave objects use internal arrays to also store
! direct fluxes, not needed in the longwave
call       bc_out%allocate(ncol, nsw, nlw)
call sw_norm_diff%allocate(config, ncol, ntotlay, nsw, use_direct=.true.)
call  sw_norm_dir%allocate(config, ncol, ntotlay, nsw, use_direct=.true.)
call  lw_internal%allocate(config, ncol, ntotlay, nlw, use_direct=.false.)
call      lw_norm%allocate(config, ncol, ntotlay, nlw, use_direct=.false.)

! Optionally set all fluxes to zero
call  sw_norm_dir%zero_all()
call sw_norm_diff%zero_all()
call  lw_internal%zero_all()
call      lw_norm%zero_all()
```

We are now in a position to call the `radsurf` routine:

```
! Read module defining the radsurf routine
use radsurf_interface, only : radsurf

! ...prepare the objects to pass to radsurf...

! Call the SPARTACUS-Surface radiation scheme
call radsurf(config, canopy_props, sw_spectral_props, lw_spectral_props, & ! Inputs
     &       bc_out, &                                                     ! Outputs
     &       istartcol, iendcol, &                                         ! Optional inputs
     &       sw_norm_dir, sw_norm_diff, lw_internal, lw_norm)              ! Optional outputs
```

Optionally, a call to `radsurf` may specify the range of columns to process via `istartcol` and `iendcol`. This is useful for OpenMP parallelization: if the objects contain a large number of columns then `radsurf` can be called simultaneously by mutliple threads, each thread being instructed to work on a different range of columns. This is done by the offline *SPARTACUS-Surface* driver. The final four canopy flux objects are marked as 'optional' because the `config` object may have specified to only perform shortwave or longwave radiative transfer, in which case only two such output objects would be needed.

The `boundary_conds_out_type` is fairly small, containing four arrays. The arrays `sw_albedo` and `sw_albedo_dir` are the shortwave spectral albedo at top-of-canopy presented to incoming diffuse and direct solar radiation, and are dimensioned `nsw`×`ncol`. The arrays `lw_emissivity` and `lw_emission` are the top-of-canopy longwave spectral emissivity and upward emission (in W m^{-2}), dimensioned `nlw`×`ncol`.

After the atmospheric radiative transfer calculation has completed using these arrays as boundary conditions, we need to use the downwelling fluxes at top-of-canopy to scale the normalized canopy fluxes.

```
! Declare objects to contain total (unnormalized) canopy fluxes
```

```
type(canopy_flux_type) :: lw_flux, sw_flux

! Allocate these objects
call sw_flux%allocate(config, ncol, ntotlay, nsw, use_direct=.true.)
call lw_flux%allocate(config, ncol, ntotlay, nlw, use_direct=.false.)

! Suppose the atmospheric radiative transfer scheme has provided three arrays of spectral
! downwelling radiation at top of canopy, top_flux_dn_diffuse_sw(nsw,ncol),
! top_flux_dn_direct_sw(nsw,ncol) and top_flux_dn_lw(nlw,ncol), we can use them to scale
! the normalized shortwave and longwave canopy fluxes, also taking as input the number of
! layers stored in each column, canopy_props%nlay
call sw_norm_diff%scale(canopy_props%nlay, top_flux_dn_diffuse_sw)
call  sw_norm_dir%scale(canopy_props%nlay, top_flux_dn_direct_sw)
call      lw_norm%scale(canopy_props%nlay, top_flux_dn_lw)

! Finally, sum the two contributions in each spectral region to obtain total (unnormalized)
! canopy fluxes
call sw_flux%sum(sw_norm_dir, sw_norm_diff)
call lw_flux%sum(lw_internal, lw_norm)
```

The contents of `sw_flux` and `lw_flux` are then available to use in a canopy energy balance scheme. See `radsurf_radsurf_canopy_flux.F90` for the precise contents of these objects. Note that at this stage they are still in the same spectral intervals as used by the *SPARTACUS-Surface* calculation. In the shortwave this is useful as it enables the photosynthetically active part of the spectral absorption by vegetation to be computed.

Further development is still needed on the spectral aspect of SPARTACUS-Surface*. It is expected that in the shortwave, the radiation calculations will all be performed in the user-specified spectral intervals, and atmospheric extinction will either be ignored or it will be left to the user to provide atmospheric extinction coefficients, especially due to aerosols. It ought to be reasonable to neglect gas absorption and Rayleigh scattering since the Rayleigh optical depth is small through the limited depth of a surface canopy, and in the parts of the near-infrared spectrum where the gaseous absorpion is large, little solar radiation is likely to penetrate down through the atmosphere to the top of the canopy.*

In the longwave, there will need to be a mechanism to treat gaseous absorption given its importance (Hogan, 2019b). This could be implemented by mapping the user-specified longwave spectral intervals on to a larger number of intervals suitable for longwave radiative transfer and running SPARTACUS-Surface *at this spectral resolution. The larger number of intervals could then be passed to the atmospheric longwave radiative transfer. For the canopy energy balance model, only the broadband longwave fluxes are likely to be of interest, which could be computed by simply summing the fluxes along the* `nlw` *dimension.*

9 License and copyright

References

Harman, I. N., M. J. Best and S. E. Belcher, 2004: Radiative exchange in an urban street canyon. *Boundary-Layer Meteorol.,* **110,** 301–316.

Hogan, R. J., 2019a: An exponential model of urban geometry for use in radiative transfer applications. *Boundary-Layer Meteorol.,* **170,** 357–472.

Hogan, R. J., 2019b: Flexible treatment of radiative transfer in complex urban canopies for use in weather and climate models. *Boundary-Layer Meteorol.,* **173,** 53-78.

Hogan, R. J., T. Quaife and R. Braghiere, 2018: Fast matrix treatment of 3-D radiative transfer in vegetation canopies: SPARTACUS-Vegetation 1.1. *Geosci. Model Dev.,* **11,** 339-350.

Hogan, R. J., S. A. K. Schäfer, C. Klinger, J.-C. Chiu and B. Mayer, 2016: Representing 3D cloud-radiation effects in two-stream schemes: 2. Matrix formulation and broadband evaluation. *J. Geophys. Res.,* **121,** 8583–8599.

Shonk, J. K. P., and R. J. Hogan, 2008: Tripleclouds: an efficient method for representing horizontal cloud inhomogeneity in 1D radiation schemes by using three regions at each height. *J. Climate,* **21,** 2352–2370.

Stamnes K., S. C. Tsay, W. Wiscombe and K. Jayaweera, 1988: Numerically stable algorithm for discrete-ordinate-method radiative transfer in multiple scattering and emitting layered media. *Appl. Opt.,* **27,** 2502–2509.