Satellite Data Simulator Unit (SDSU) ver. 2 User's Guide

Release 2.1

revised for ver. 2.1.2

Hirohiko Masunaga[†]

May 2010

[†]Hydrospheric Atmospheric Research Center, Nagoya University

Contents

1 Introduction

1.1 Overview

The Satellite Data Simulator Unit version 2 (SDSU-v2) is a Fortran 90 package to compute synthetic satellite data from user-provided geophysical parameters such as cloud-resolving model (CRM) output. The SDSU is designed to simulate microwave brightness temperature, radar reflectivity, radar path-integrated attenuation (PIA), visible and near-infrared radiances, and thermal infrared brightness temperature. The existing and prospective satellite sensors applicable include (but are not limited to):

1. Microwave radiometers and sounders*
    - Special Sensor Microwave/Imager (SSM/I) and Special Sensor Microwave Imager/Sounder (SSMIS)
    - Tropical Rainfall Measuring Mission (TRMM) Microwave Imager (TMI)
    - Advanced Microwave Scanning Radiometer (AMSR) and AMSR-E
    - Global Precipitation Measurement Mission (GPM) Microwave Imager (GMI)
    - Coriolis WindSat
    - Advanced Microwave Sounding Unit (AMSU) and Microwave Humidity Sounder (MHS)
2. Radars
    - TRMM Precipitation Radar (PR)
    - CloudSat Cloud Profiling Radar (CPR)
    - GPM Dual-frequency Precipitation Radar (DPR)
3. Visible and infrared (IR) imagers
    - Advanced Very High Resolution Radiometer (AVHRR)
    - TRMM Visible/Infrared Scanner (VIRS)
    - Moderate Resolution Imaging Spectroradiometer (MODIS)
    - Visible/IR sensors onboard operational geostationary satellites such as GMS (MTSAT), GOES, and Meteosat.

The SDSU-v2 package also includes a beam convolution routine to adjust the spatial resolution to a sensor field of view (FOV), a Mie lookup table (LUT) generator to improve computational efficiency, sample CRM datasets (full package only), and the user's guide (this document).

*A minor modification to the radiative transfer code will be needed to simulate mixed polarization channels typical of cross-track scanning sounders

1.2 What's new in the version 2?

Since the earliest version was released, it has been felt that there were some key issues with the SDSU that had yet to be resolved. First, the SDSU version 1 was coded in a somewhat ancient style of Fortran 77. A main task for the SDSU upgrade was to rewrite the entire code, except some ancillary subroutines, in the Fortran 90 fashion. This revision helped refine the old SDSU code. The pre-fixed array dimensions such as `mxgridx` and `mxgridy`, previously needed to assure static memory allocation, are no longer required. Memory space assigned to the geophysical and radiative parameters is now dynamically allocated with `ngridx`, `ngridy`, and `nlyr` provided from the input data. The F77 include files (`*.inc`) to declare primary variables are now replaced by F90 modules, eliminating lengthy argument lists from the top of major subroutines.

Another problem (which may not be regarded as any problem to many users, though) of earlier SDSU versions is that microphysical assumptions such as particle size distribution (PSD)* and particle density were not modifiable unless the user manually makes changes to low-level subroutines. A discrepancy in PSD, however, could dramatically change the results for simulations sensitive to hydrometeor size (e.g., radar reflectivity and visible optical thickness). It is therefore important, in such cases, to specify a PSD model consistent with the microphysical scheme implemented in the user's CRM when the SDSU is run to compare CRM output with satellite observations. An intriguing application utilizing this fact is to do sensitivity tests with different PSD models to see what microphysical scheme best reproduces satellite measurements (e.g., Masunaga et al., 2008). The SDSU-v2 includes the user interface to explicitly characterize the PSD model of each hydrometeor species simulated.

The users of the SDSU version 1 may have noticed that the visible/IR simulator is somehow "isolated" from the other two simulator components. The Mie LUT option did not work for visible/IR simulations, making the simulations extremely time consuming. To resolve this problem, the visible/IR simulator has been completely renewed in the SDSU-v2. First, the visible/IR simulator, previously based on RSTAR5c, has been upgraded to its latest version or the RSTAR6b, where the gas absorption table has been thoroughly updated (Sekiguchi and Nakajima, 2008). Second, the Mie calculation code built in the RSTAR was replaced by a new code created from the existing SDSU subroutines running with the microwave radiometer and radar simulators. This upgrade facilitates the installation of user-provided PSDs into the visible/IR simulator with or without the Mie LUTs. The program to generate Mie LUTs, hereafter called the Mie LUT generator, has been also revised so it automatically reflects changes made to the PSD settings. The LUT option works for the visible/IR simulator as well as the microwave radiometer and radar simulators in the SDSU-v2. Computational efficiency dramatically improves with the LUT option activated, although the visible/IR simulator is still considerably slower than the other two simulators.

Other changes from the SDSU ver. 1 to the ver. 2 are summarized as follows.

- The height coordinate was a one-dimensional variable in the SDSU version 1, but is now allowed to vary horizontally so that pressure-coordinated grids are readily incorporated.
- The subroutines calculating the hydrometeor radiative properties and gas absorption tables are now stored in separate directories (`src/simulator/hydromet/` and `src/simulator/gas/`).

*The term PSD is hereafter used for all hydrometeor species including rain and liquid cloud, for which PSD is conventionally called drop/droplet size distribution or DSD.

- The beam convolution program and subroutines are stored in a separate directory (`src/beamconv/`). Beam convolution is applicable to simulated data either online with or separately from the main SDSU routine.
- In the SDSU version 1, the Mie LUT files needed to be tagged manually for separating different channel frequencies. In the SDSU-v2, the Mie LUTs are named automatically after the specified channel frequency (or wavelength) and PSD model. The main simulator unit analyzes information given by the user and reconstructs the LUT tag to track down the proper LUT.
- The Mie LUT storage (`Mie-table/`) is divided into sub-directories separated by different PSD models. The melting particle LUTs are stored in their own directory (`melt-p/`)
- Some variables and files have been renamed for clarity.

1.3 Unpacking the SDSU-v2

The SDSU-v2 package is available for download from the SDSU WWW site

`http://precip.hyarc.nagoya-u.ac.jp/sdsu/index.html`

for registered users. Registration only requests you to provide your name and email address*. The user can choose either the full package or source-only package. The full package contains all the SDSU routines and sample CRM dataset, while the source-only package includes the source code files without the sample CRM and sample output.

The SDSU-v2 package, once expanded by `gunzip` and `tar`, consists of the directories illustrated in Figure 1. In the source tree `src/`, the main simulator unit (see section 2 for detailed descriptions) is stored in `simulator/`, the Mie LUT generator (section 4) in `MieLUT/`, and the beam convolution routine (section 5) in `beamconv/`. Input geophysical parameters are read in from `CRM/` and the simulated data (synthesized satellite data) are written out into `outputs/`. The storage for the Mie LUTs is designated by `Mie-table/`.

1.4 Disclaimer

The SDSU is developed and maintained by Hiro Masunaga (Hydrospheric Atmospheric Research Center, Nagoya University). The SDSU is not guaranteed to perform always as intended, depending on users' computer environment and other factors, so please use it at your own risk. Users are prohibited from redistributing any part of the SDSU package without permission. The latest version of the SDSU-v2 as of May 21, 2010 is the version 2.1.2. See `RELEASE_NOTES` in the SDSU-v2 top directory for the upgrade history.

*Registrants will be notified of any issues, if found, that are worth attention of general users. Provided information will never be used for any other purposes.

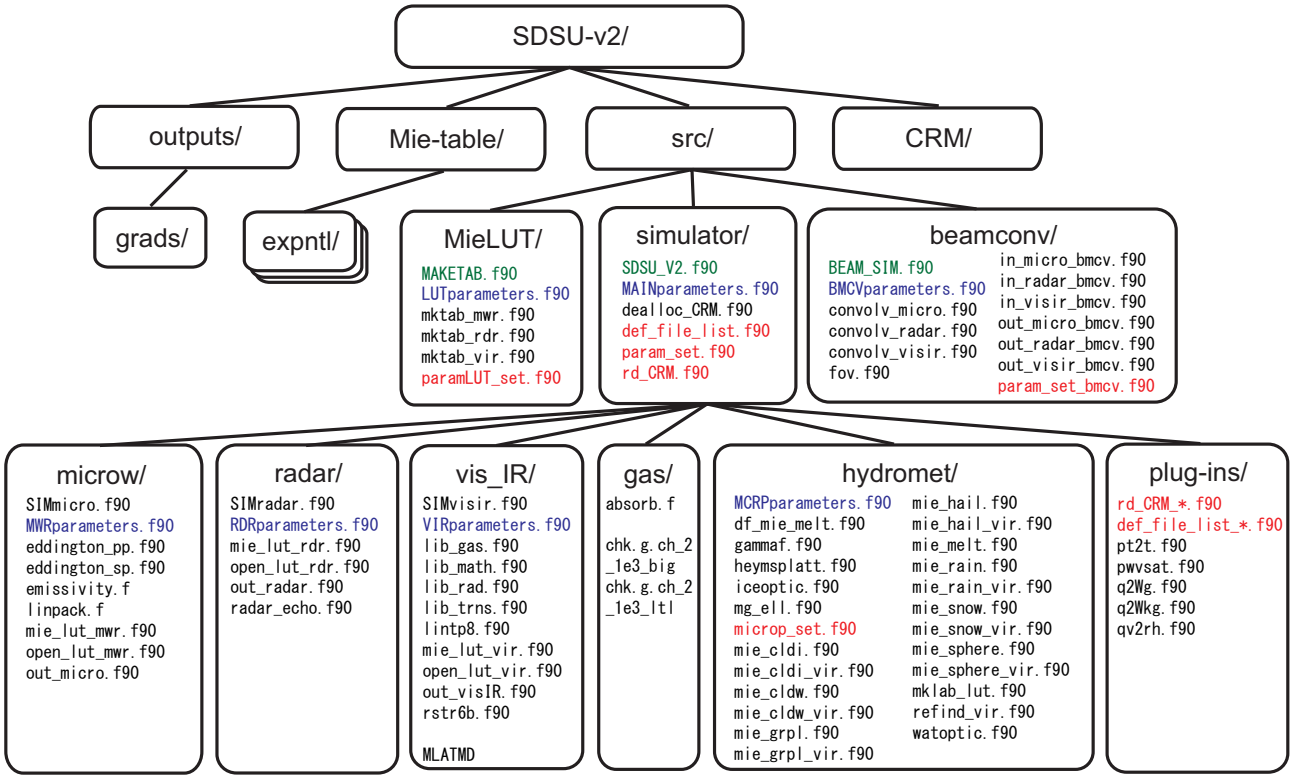

Figure 1: SDSU directory tree. Main programs are highlighted in green, F90 modules in blue, and user interface routines in red.

2 Main Simulator Unit

This section is devoted to the thorough documentation of the main simulator unit. Instructions are given for running the code (section 2.1.1), parameter setting regarding the individual simulator components (section 2.1.2), input geophysical parameter settings (section 2.1.3), and microphysical model settings (section 2.1.4). The output parameters and formats are described in section 2.2 and additional notes on the simulator components are given in section 2.3.

2.1 User Interface

2.1.1 To run the SDSU

To run the SDSU, go to the directory `src/simulator/` and first make sure that Fortran 90 compiler and compiling options are properly specified in `makefile` so they work with your compiler. If you do not have the Intel Fortran compiler (*ifort*) installed in your computer, you will need to modify 3 lines defining the compiler setup (lines 67-69) in `makefile`. The Fortran compiler is defined by `COMPL`, and compiling options are given by `FLAGS`. The I/O binary record length must be specified in terms of byte (not in multiples of 4 bytes). Paths to the F90 modules are given by `FINCL`.

Once the compiler setup is done, run the `makefile`,

```
% make sdsu
```

and you will obtain the executable `sdsu.x`. If you have input geophysical parameters ready, go ahead to run the SDSU.

```
% sdsu.x
```

Without any customization, the SDSU will read in the sample CRM profiles (contained in the full package) and write out a set of microwave radiometer, radar, and visible/IR simulations. However, this is probably not exactly what you really want to do. Instructions are provided below to customize the SDSU so it serves your own purposes.

2.1.2 General simulator settings (`param_set`)

Running options including satellite sensor specifications can be customized by editing `param_set.f90`. The parameters reviewed in this subsection are all defined there.

Customizing the microwave radiometer simulator The microwave radiometer simulator is controlled by the parameters listed in the table below.

Microwave radiometer simulator options (`param_set.f90`)

Parameter	Parameter type	Description
`microw`	logical	The simulator is run when true.
`mxfreq_microw`	integer	number of frequency channels
`freq_microw`	real(1:mxfreq_microw)	channel frequencies [GHz]
`inc_angle_mwr`	real	viewing zenith angle [°]
`slant_path`	logical	The slant path option is on when true.

The SDSU computes a pair of brightness temperatures, polarized vertically and horizontally, for each frequency specified. The viewing zenith angle here and for other simulators should be the value measured at the Earth's surface.

The slant path option introduces a "pseudo" three-dimensional effect to the plane-parallel approximation. Radiative transfer calculation is performed along a slant path of "observed" microwave radiance if this option is turned on. This is done by first tracing the layers intersected by a given microwave ray and then applying a plane-parallel calculation to the identified layers. The microwave ray is assumed to be confined within the y-z plain. When the slant path option is off, plane-parallel calculations are carried out for each vertical column individually, that is, in the way occasionally called the independent-pixel approximation.

By default `freq_microw` and `inc_angle_mwr` are taken from the TMI instrumental design.

Customizing the radar simulator The radar simulator is controlled by the parameters listed below.

Radar simulator options (`param_set.f90`)

Parameter	Parameter type	Description
`radar`	logical	The simulator is run when true.
`mxfreq_radar`	integer	number of frequency channels
`freq_radar`	real(1:mxfreq_radar)	channel frequencies [GHz]
`inc_angle_rdr`	real	viewing zenith angle [°]

By default `freq_radar` follows the GMI DPR instrumental design.

Customizing the visible/IR simulator The visible/IR simulator is controlled by the parameters listed below.

Visible/IR simulator options (`param_set.f90`)

Parameter	Parameter type	Description
`visIR`	logical	The simulator is run when true.
`mxwavel`	integer	number of wavelength channels
`wavel`	real(1:mxwavel)	channel wavelengths [μm]
`tbform`	logical(1:mxwavel)	converted to T_b [K] when true
`znth_slr`	real	solar zenith angle [°]
`znth_obs`	real	viewing zenith angle [°]
`azmth`	real	azimuthal angle [°] between the Sun and observer
`ngausp`	integer(1:mxwavel)	number of Gaussian quadrature points
`lab_endian`	character*3	"ltl" or "big" depending on the CPU

The computed result is written out either in raw radiance or in the brightness temperature form depending on the user's choice. Radiance is internally converted to brightness temperature using the Planck function for channels with `tbform=.true.`, and otherwise it remains unconverted. Typically `tbform=.true.` is good for thermal infrared channels.

The parameter `ngausp` specifies the number of Gaussian quadrature points used for the discrete-ordinate radiative transfer engine. The two-stream approximation, for example, is realized when `ngausp`=1. The smaller `ngausp` is, the faster the calculation is at the expense of a loss of computational accuracy. Different channels may be tuned with different `ngausp`s. It is recommended to set `ngausp` to be 6 or higher for shortwave simulations and 2 or higher for longwave to assure reasonable accuracy.

The last parameter `lab_endian` should be adjusted to your CPU endian: `lab_endian` = "`ltl`" ("`big`") for little endian (big endian) CPUs. This label is needed for the simulator to access the proper gas absorption table.

By default `wavel` corresponds to the TRMM VIRS instrumental design.

Other running options There are a few more options to set in `param_set.f90`.

Other running options (`param_set.f90`)

Parameter	Parameter type	Applicable to[‡]	Description
`background`	character*5	M/V	background type ("water","land", or "mixed")
`lut`	logical	M/R/V	The Mie LUTs are used when true.
`meltp`	logical	M/R	Melting particles are considered when true.
`beamconv`	logical	M/R/V	Beam convolution is applied online when true.

The background type ("water", "land", or "mixed") separates ocean and land surfaces and affects the surface emissivity estimate for passive sensor simulations. While the water surface emissivity is reliable for most applications, the land emissivity is grossly simplified in the current version of the SDSU. The background type is assumed to be uniform across the computational domain if "water" or "land" is specified. The surface type is allowed to vary spatially when `background='mixed'` with an additional 2D parameter `sfc_type` defined to specify the local surface type (see "surface parameter" section later). When `background='water'` or `'land'`, `sfc_type` is ignored and may be left undefined.

The Mie LUT option, activated when `lut=.true.`, greatly accelerates simulations and is highly recommended to utilize. The LUTs must be created in advance of running the simulator unit. See section 4 for instructions to use the Mie LUT generator.

When the melting particle option `meltp` is turned on, the simulator generates particles made of water-and-ice mixture in the melting layer (a layer immediately below the 0°C level). These melting particles give rise to the bright band in simulated radar echoes and some increase in microwave brightness temperature at low frequencies. The melting particle option is ignored by the visible/IR simulator because no melting layer effect is expected for shortwave and longwave radiances.

Set `beamconv` to be ".true." if the user wishes to obtain simulated data with beam convolution applied in addition to those at the original input-model resolution. Alternatively, beam convolution may be applied off-line by running afterward the stand-alone beam convolution program in `src/beamconv/`. See section 5 for details in the beam convolution routine.

[‡]Acronyms are: microwave radiometer simulator (M), radar simulator (R), visible/infrared simulator (V), and beam convolution routine (B)

2.1.3 Input geophysical parameters (rd_CRM)

The directory CRM/ serves as the storage for input geophysical parameters. The SDSU full package comes with sample cloud-resolving model profiles from the Goddard Cumulus Ensemble (GCE) model (Tao and Simpson, 1993), provided by courtesy of Wei-Kuo Tao (NASA GSFC). This sample database contains four snapshots taken from a simulated tropical squall line observed during the GARP Atlantic Tropical Experiments (GATE). The input geophysical parameters are read in by rd_CRM.f90. The input file names are defined in def_file_list.f90.

This subsection describes the geophysical parameters required for the simulator input. The input data files may be named and organized in arbitrary format as far as the whole set of the required parameters are supplied and rd_CRM.f90 and def_file_list.f90 are revised accordingly.

Input file list The list of input file names is defined in def_file_list.f90. The list may include an arbitrary number of CRM snapshots but a single file must not contain two snapshots or more. The directory path to the input files is designated by CRMdir specified in MAINparameters.f90.

List of input file names (def_file_list.f90)

Parameter	Parameter type	Applicable to‡	Description
nmodel	integer	M/R/V/B	number of files/snapshots
CRM_file_list	character*90(1:nmodel)	M/R/V/B	file names

The latest version of SDSU (v2.1.2 or later) also offers an alternative option that the input file list is read in from command line arguments, instead of being specified in def_file_list.f90. This option may be turned on by adding the following line to def_file_list.f90,

```
nmodel = 1 ; CRM_file_list = '2BREADIN'
```

Provide your input file list when running the SDSU, e.g.,

```
% sdsu.x FILE1 FILE2 FILE3 ...
```

The number of input files is unlimited. It will be automatically detected and override nmodel.

Dimension sizes The three spatial dimension sizes of input data are defined by ngridx, ngridy, and nlyr. Horizontal grid intervals deltax and deltay, which must be uniform across the model domain, are used only by the beam convolution routine.

Input parameter dimension sizes (rd_CRM.f90)

Parameter	Parameter type	Applicable to‡	Description
ngridx	integer	M/R/V	horizontal (x) dimension size
ngridy	integer	M/R/V	horizontal (y) dimension size
nlyr	integer	M/R/V	vertical dimension size
deltax	real	B	horizontal (x) grid interval [km]
deltay	real	B	horizontal (y) grid interval [km]

Note that ngridx, ngridy, and nlyr must be ingested at the very beginning of the input data file since these parameters are used for memory allocation to the remaining parameters.

Surface parameters The required surface parameters are listed in the table below. The skin temperature is the ocean/land surface temperature used to evaluate the surface emissivity for passive sensor simulations. The convective/stratiform separation, `ics`, is 1 for convective rain and is 2 for stratiform rain. (`ics` is not actually a "surface" parameter but is classified here for convenience because it is a two-dimensional variable as are other surface parameters). When the melting particle option is activated, melting particles are assumed to be present for stratiform rain while absent for convective. The near-surface wind affects the ocean surface emissivity. The soil moisture fraction and ground albedo, ranging from 0 to 1, control the land surface emissivity where the background type is specified to be land. A Lambert surface is assumed to derive the land surface emissivity. The local surface type or `sfc_type`, 0 for open ocean and 1 for land surface, defines spatially variable surface types when `background='mixed'`. Note that when `background='water'` or `'land'`, the background condition is forced to be ocean or land everywhere across the domain regardless of `sfc_type`. By default `background` is "water" and `sfc_type` is undefined. There is a slot reserved for surface rain rate (`rainrate_sfc`) for the potential convenience of users but is not used by any SDSU component.

Surface parameters (`rd_CRM.f90`)

Parameter	Parameter type	Applicable to‡	Description
`tskin`	real(1:ngridx,1:ngridy)	M/V	skin temperature [K]
`ics`	integer(1:ngridx,1:ngridy)	M/R	1 (2) for convective (stratiform)
`wind_sfc`	real(1:ngridx,1:ngridy)	M/V	near-surface (10-m) wind [m/s]
`soilH2O`	real(1:ngridx,1:ngridy)	M	soil moisture fraction
`albd_visir`	real(1:ngridx,1:ngridy,1:mxwavel)	V	visible/infrared ground albedo
`sfc_type`	integer(1:ngridx,1:ngridy)	M/V	0 (1) for ocean (land) surface
`rainrate_sfc`	real(1:ngridx,1:ngridy)	-	surface rain rate (optional)

Atmospheric parameters In the SDSU, there are two categories of atmospheric parameters in terms of vertical configuration: one defined at layer interfaces and another averaged within a layer. The first category includes pressure, temperature, relative humidity, and height coordinate. Subroutines to compute air temperature from potential temperature and estimate relative humidity from water vapor mixing ratio are available in the SDSU (see section 2.1.5). Vertical wind (`wind_lev`) is, like surface rain rate, a dummy parameter not ingested by the SDSU.

Atmospheric parameters at layer interfaces (`rd_CRM.f90`)

Parameter	Parameter type	Applicable to‡	Description
`hght_lev`	real(1:ngridx,1:ngridy,0:nlyr)	M/R/V	height [km]
`pres_lev`	real(1:ngridx,1:ngridy,0:nlyr)	M/R/V	pressure [hPa]
`temp_lev`	real(1:ngridx,1:ngridy,0:nlyr)	M/R/V	air temperature [K]
`rlhm_lev`	real(1:ngridx,1:ngridy,0:nlyr)	M/R/V	relative humidity [%]
`wind_lev`	real	-	vertical wind [m/s] (optional)

The 0-th layer interface corresponds to the surface.

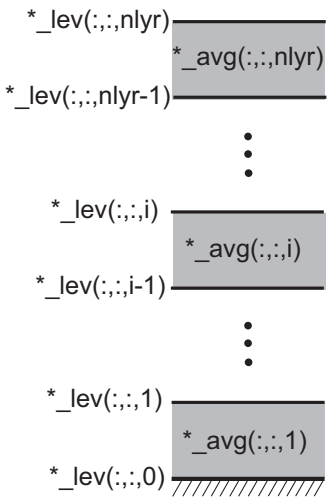

Figure 2: Vertical configurations of atmospheric parameters at layer interfaces (*_lev) and layer-averaged parameters (*_avg).

Hydrometeor properties are defined as layer-averaged variables, for which the i-th slot in the vertical (third) dimension represents the domain bound between the i-th and $(i-1)$-th layer interfaces (Figure 2). The SDSU allows six hydrometeor species to exist: liquid and ice clouds, rain, snow, graupel, and hail. If any of these parameters is unavailable in the input dataset, fill in the absent parameter(s) with zeros. The SDSU requires hydrometeor water content, or mixing ratio multiplied by dry-air density (ρq), instead of hydrometeor mixing ratio (q). A subroutine to convert q to ρq is provided (see section 2.1.5).

Layer-averaged atmospheric parameters (hydrometeor water contents) (`rd_CRM.f90`)

Parameter	Parameter type	Applicable to‡	Description
`Wcldi_avg`	real(1:ngridx,1:ngridy,1:nlyr)	M/R/V	cloud ice content [g m^{-3}]
`Wcldw_avg`	real(1:ngridx,1:ngridy,1:nlyr)	M/R/V	cloud water content [g m^{-3}]
`Wrain_avg`	real(1:ngridx,1:ngridy,1:nlyr)	M/R/V	rain water content [g m^{-3}]
`Wsnow_avg`	real(1:ngridx,1:ngridy,1:nlyr)	M/R/V	snow ice content [g m^{-3}]
`Wgrpl_avg`	real(1:ngridx,1:ngridy,1:nlyr)	M/R/V	graupel ice content [g m^{-3}]
`Whail_avg`	real(1:ngridx,1:ngridy,1:nlyr)	M/R/V	hail ice content [g m^{-3}]

In the SDSU-v2, the hydrometeor number concentration may be explicitly specified to characterize the hydrometeor PSDs. This allows to define a PSD consistent with two-moment bulk microphysical schemes. The number concentration is given by the parameters listed below. These parameters are required only when the selected PSD model is a function of the number concentration.

Layer-averaged atmospheric parameters (hydrometeor number concentrations) (`rd_CRM.f90`)

Parameter	Parameter type	Applicable to‡	Description
Ncldi_avg	real(1:ngridx,1:ngridy,1:nlyr)	M/R/V	ice cloud number concentration [m^{-3}]
Ncldw_avg	real(1:ngridx,1:ngridy,1:nlyr)	M/R/V	liquid cloud number concentration [m^{-3}]
Nrain_avg	real(1:ngridx,1:ngridy,1:nlyr)	M/R/V	rain number concentration [m^{-3}]
Nsnow_avg	real(1:ngridx,1:ngridy,1:nlyr)	M/R/V	snow number concentration [m^{-3}]
Ngrpl_avg	real(1:ngridx,1:ngridy,1:nlyr)	M/R/V	graupel number concentration [m^{-3}]
Nhail_avg	real(1:ngridx,1:ngridy,1:nlyr)	M/R/V	hail number concentration [m^{-3}]

2.1.4 Microphysical model settings (microp_set)

Hydrometeor microphysical properties such as PSD and particle density can be customized by editing microp_set.f90 in src/simulator/hydromet/. The PSD function type and PSD parameters including particle density are controlled using a F90 data structure psd_params constituted of the parameters shown below.

F90 data structure psd_params (MCRPparameters.f90)

Parameter	Parameter type	Description
name	character*8	PSD model name
functype	character*6	PSD function type identifier
nparams	integer	number of PSD parameters
paramset	real(1:nparams)	the whole set of PSD parameters
normaliz	logical	External normalization is required when true.
indpndtT	logical	The PSD parameters are independent of temperature when .true.
twomomnt	logical	Set to be .true. for two-moment PSD models

The PSD model name (name) is an arbitrary character string of 8 letters or shorter (e.g., 'cldiHP84'). This parameter is used to label the Mie LUT files (see section 4) and must be a unique index to each PSD model. The function type identifier functype must match one of the existing PSD models in the library. Rules for nparams and paramset are established individually for each PSD model (see the table below). The user can ignore normaliz, indpndtT, and twomomnt except when they create a new PSD model to add to the library (section 3.3). At present, 9 function types are available.

SDSU PSD library (psd_func in MCRPparameters.f90)

PSD function type	functype	nparams	paramset
Exponential PSD	'expntl'	2	#1. particle density [kg m^{-3}] #2. intercept parameter [m^{-4}]
Gamma PSD for given μ and n_w	'gammwN'	3	#1. particle density [kg m^{-3}] #2. normalized intercept parameter n_w [m^{-4}] #3. exponent μ

PSD function type	`functype`	`nparams`	`paramset`
Gamma PSD for given μ and D_0	`'gammwD'`	3	#1. particle density [kg m^{-3}] #2. median volume diameter D_0 [mm] #3. exponent μ
Log-normal PSD	`'lgnorm'`	3	#1. particle density [kg m^{-3}] #2. mode diameter D_m [mm] #3. dispersion in logarithm σ
Exponential PSD with a power-law mass spectrum ($m(D) = aD^b$)	`'expaDb'`	4	#1. N/A #2. intercept parameter [m^{-4}] #3. a in [kg m^{-b}] #4. b
Modified gamma PSD	`'modGam'`	3	#1. particle density [kg m^{-3}] #2. mode diameter D_m [mm] #3. exponent μ
Heymsfield and Platt (1984) for cloud ice PSD	`'HyPl84'`	2	#1 N/A #2. running option (0 or 1) =0 for the equal diameter scheme =1 for the equal mass scheme
2-moment exponential PSD	`'exp2mo'`	1	#1. particle density [kg m^{-3}]
2-moment lognormal PSD	`'lgn2mo'`	2	#1. particle density [kg m^{-3}] #2. dispersion in logarithm σ

See section 3 for the complete description of the PSD library.

Microphysical parameters are defined for each hydrometeor species of ice cloud (`cldi`), liquid cloud (`cldw`), rain (`rain`), snow (`snow`), graupel (`grpl`), and hail (`hail`). An example from `microp_set.f90` is

```
cldw%name = 'cldw'
cldw%functype = 'lgnorm' ; cldw%nparams = 3
cldw%paramset(1:cldw%nparams) = (/ densliq, 2.e-2, 0.35 /)
```

where the cloud water PSD is defined to be a log-normal function with the mode diameter of 20 μm and the dispersion of 0.35. Liquid and ice water density, `densliq`=1.0$\times 10^3$ kg m^{-3} and `densice`=0.917$\times 10^3$ kg m^{-3}, respectively, are pre-defined constants and available for the user. Another example is

```
snow%name = 'snow'
snow%functype = 'expntl' ; snow%nparams = 2
snow%paramset(1:snow%nparams) = (/  0.1e+3, 1.0e+8 /)
```

where the snow particle density is 0.1×10^3 kg m^{-3} and the snow PSD is exponential with the intercept parameter of 1.0×10^8 m^{-4}.

2.1.5 SDSU plug-in library (`src/plug-ins`)

A directory to store "plug-in" routines is included in the SDSU package since ver. 2.1.0. The files named as `def_file_list_*.f90` and `rd_CRM_*.f90` are the template subroutines that replace `def_file_list.f90` and `rd_CRM.f90` for input data in different formats. See the header of `rd_CRM_*.f90` for instructions to activate these routines. Subroutines currently available in the plug-in library are listed in the table below.

Plug-in routines (`src/plug-ins/`)

Files	Description
`pt2p.f90`	computes T in [K] from θ and p
`pwvsat.f90`	computes saturation vapor pressure in [hPa]
`q2Wg.f90`	computes ρq in [g/m^3] from q
`q2Wkg.f90`	computes ρq in [kg/m^3] from q
`qv2rh.f90`	computes relative humidity from vapor mixing ratio
`def_file_list_CReSS.f90`	`def_file_list.f90` for CReSS*
`rd_CRM_CReSS.f90`	`rd_CRM.f90` for CReSS

The plug-in library will be expanded whenever new subroutines become available in future versions of the SDSU.

2.2 Output Parameters and Format

The output parameters and file format are described in this section. Simulated data are all output into `outputs/`. The naming rule for output files is simply to suffix an extension ".org" to the input file name. The extension is replaced by ".fov" for the simulated output with beam convolution applied. ASCII files "*.info" contain a short note recording the running and microphysical options adopted.

2.2.1 Microwave radiometer simulator

Computed brightness temperature is written out in plain binary format by `out_micro.f90`.

Output parameter for the microwave radiometer simulator (`out_micro.f90`)

Parameter	Parameter type	Description
`Tb_out`	real(1:ngridx,1:ngridy,1:mxfreq_microw,0:1)	microwave brightness temperature [K]

In the output file, `Tb_out` is looped fourfold over x dimension, y dimension, polarization (the horizontal first), and channel frequency in this order from the inner to outer loops. For those familiar with the GrADS, sample control files ("*_mwr.ctl") to plot simulated microwave brightness temperature are available in `outputs/grads/`.

*Cloud Resolving Storm Simulator (CReSS) is developed and distributed at Hydrospheric Atmospheric Research Center, Nagoya University, Japan.

2.2.2 Radar simulator

Computed radar reflectivity and PIA are written out in plain binary format by `out_radar.f90`.

Output parameter for the radar simulator (`out_radar.f90`)

Parameter	Parameter type	Description
`dBZ_out`	real(1:ngridx,1:ngridy,1:nlyr,1:mxfreq_radar)	radar reflectivity [dBZ]
`PIA_out`	real(1:ngridx,1:ngridy,1:mxfreq_radar)	2-way path-integrated attenuation [dB]

In the output file, `dBZ_out` is looped fourfold over x dimension, y dimension, vertical dimension, and channel frequency in this order from the inner to outer loops. Similarly, `PIA_out` is looped threefold with the vertical dimension skipped. Sample GrADS control files ("*_rdr.ctl") to plot simulated radar echo and PIA are available in `outputs/grads/`.

2.2.3 Visible/infrared simulator

Computed visible/infrared radiance is written out in plain binary format by `out_visIR.f90`.

Output parameter for the visible/infrared simulator (`out_visIR.f90`)

Parameter	Parameter type	Description
`Ivis_out`	real(1:ngridx,1:ngridy,1:mxwavel)	radiance [W m^{-2} μm^{-1} str^{-1}] where `tbform`=.false. or brightness temperature [K] where `tbform`=.true.

In the output file, `Ivis_out` is looped threefold over x dimension, y dimension, and channel wavelength in this order from the inner to outer loops. Sample GrADS control files ("*_vir.ctl") to plot simulated radiance are available in `outputs/grads/`.

2.3 Brief Notes on the Simulators

2.3.1 Microwave radiometer simulator

The subroutines relevant to microwave radiometer simulations are stored in `microw/` (see Figure 1). The main simulator routine is `SIMmicro.f90`. Microwave brightness temperature is computed by a plane-parallel Eddington radiative transfer scheme developed by Kummerow (1993). The "slant path" option slightly relaxes the one-dimensional constraint as described in section 2.1.2. The radiative transfer code with the gas absorption and surface emissivity subroutines was provided by courtesy of Chris Kummerow (Colorado State University).

The microwave land surface emissivity is synthesized simply by a weighted average of the water emissivity and a fixed value of 0.9 where the soil moisture fraction serves as the averaging weight. Care must be taken when `background`='land' since the actual land surface emissivity is highly variable in space and time and can be a source of large errors in simulated brightness temperature.

2.3.2 Radar simulator

The subroutines relevant to radar simulations are stored in `radar/` (see Figure 1). The main simulator routine is `SIMradar.f90`. The radar equation with the attenuation term included is solved along each

vertical column to obtain radar reflectivity and PIA. Three-dimensional effects are not considered in the radar simulator: increasing the viewing zenith angle only multiples a cos θ factor and does not allow radar paths to intersect neighboring columns. The surface reflection is not included in the simulation, so that surface clutters are not reproduced. See Masunaga and Kummerow (2005) for details in the computational procedure.

2.3.3 Visible/infrared simulator

The subroutines relevant to visible/IR simulations are stored in `vis_IR/` (see Figure 1). The main simulator routine is `SIMvisir.f90`. The radiative transfer scheme adopted here is based on a discrete-ordinate method (Nakajima and Tanaka, 1986) augmented by some technical improvements (Nakajima and Tanaka, 1988; Stamnes et al., 1988; Sekiguchi and Nakajima, 2008). No three-dimensional effect is considered beyond the so-called independent pixel approximation, that is, photons are not supposed to travel across neighboring columns no matter how large the viewing zenith angle is. Core libraries in the visible/IR simulator are based on the Remote sensing System for Transfer of Atmospheric Radiation (RSTAR) 6B (Nakajima et al., 2003), provided by courtesy of Terry Nakajima (University of Tokyo), Takashi Y. Nakajima (Tokai University), and Miho Sekiguchi (Tokyo University of Marine Science and Technology). The original RSTAR package is available from

`http://www.ccsr.u-tokyo.ac.jp/~clastr/`

2.3.4 Hydrometeor radiative properties

The radiative properties of hydrometeors are computed based on the Mie theory*. Rain drops and liquid cloud droplets are simply assumed to be homogeneous water spheres. The dielectric function of frozen hydrometeors is calculated based on the Maxwell-Garnett model with an ice matrix and air inclusions, where the volume fraction of the inclusions is estimated from a given particle density.

The Maxwell-Garnett model is also used to calculate the dielectric function of melting particles, which are assumed to have a water matrix with ice inclusions. Melting particles substitute for snow and rain within a layer below the 0°C level or the melting layer, where the melting particle PSD is constructed by a weighted average of the snow and rain PSDs. The averaging weight, or ice-to-water ratio, is assumed to linearly decrease downward across the melting layer so that the melting particle water content is continuous with the snow ice content at the melting layer top and with the rain water content at its bottom.

The original subroutines to calculate the microwave extinction and scattering coefficients and asymmetry parameter were provided by courtesy of Peter Bauer (ECMWF) and Bill Olson (NASA GSFC). Relevant publications are Bauer (2001), Olson et al. (2001), and Bauer et al. (2006) among others. These codes have been expanded by H. M. to compute the scattering phase function for use by visible/IR simulations.

*As we all know, the Mie approximation breaks down for frozen hydrometeors, which are generally far from spherical. It is left for a future improvement of the SDSU to incorporate the radiative properties of non-spherical particles.

3 Particle Size Distribution (PSD) Library

One of the largest upgrades from earlier versions of the SDSU is that particle size distributions (PSDs) are allowed to be specified by users for individual hydrometeor species in the SDSU-v2. Users can either choose one from built-in models in the PSD library such as exponential and gamma distributions, or create their own PSDs. This new feature opens ways to a variety of applications such as evaluating the validity of the microphysical scheme implemented in CRMs and doing experiments to test the sensitivity of satellite measurements to microphysics.

The current contents of the SDSU PSD library are reviewed in this section. See section 2.1.4 for the user interface to refer to the PSD library.

3.1 How does it work?

The PSD library is in the form of a Fortran function `psd_func` contained in `MCRPparameters.f90`. The Mie calculation routines (`mie_????.f90` and `mie_????_vir.f90`) call `psd_func` while integrating numerically the radiative properties over particle diameter, D. The user-specified PSD parameters are passed on to `psd_func` through the F90 data structure denoted by `cldi`, `cldw`, `rain`, `snow`, `grpl`, or `hail`. The "select case" statement tree in `psd_func` is searched for the PSD model matching the user-specified `functype`. This "select case" tree serves as the PSD library of the SDSU-v2.

3.2 PSD library inventory

The SDSU PSD library currently consists of the following PSD models.

3.2.1 Exponential PSD (`'expntl'`)

The exponential PSD,

$$n(D)dD = n_0 \exp(-\lambda D)dD, \tag{1}$$

where the slope parameter, λ, is determined by

$$\lambda = \left(\frac{\pi n_0 \rho_p}{W}\right)^{1/4}, \tag{2}$$

is specified by the intercept parameter, n_0, hydrometer water content, W, and particle density, ρ_p.

3.2.2 Gamma PSD (`'gammwN'`/`'gammwD'`)

The gamma PSD,

$$n(D)dD = n_w \frac{6}{(3.67)^4} \frac{(3.67+\mu)^{\mu+4}}{\Gamma(\mu+4)} \left(\frac{D}{D_0}\right)^{\mu} \exp\left[-(3.67+\mu)\frac{D}{D_0}\right] dD, \tag{3}$$

with Eq. (4) or (5) below is specified by W, ρ_p, the exponent μ, and either the normalized intercept parameter, n_w, or the median volume diameter, D_0. The normalized intercept parameter is equivalent to n_0 of the exponential PSD that has the same water content and D_0. See section 7.1.4 of Bringi and Chandrasekar (2001) for mathematical background.

When n_w is given by the user (`functype='gammwN'`), D_0 is determined by

$$D_0 = 3.67 \left(\frac{W}{\pi n_w \rho_p} \right)^{1/4}. \tag{4}$$

On the other hand, n_w is determined by

$$n_w = \frac{\pi \rho_p}{W} \left(\frac{3.67}{D_0} \right)^4, \tag{5}$$

when D_0 is provided by the user (`functype='gammwD'`).

3.2.3 Log-normal PSD (`'lgnorm'`)

The log-normal PSD,

$$n(D)dD = \frac{6W}{\pi\sqrt{2\pi}\rho_p \sigma D_m^3} \exp\left(-\frac{9}{2}\sigma^2\right) \exp\left[-\frac{(\ln D - \ln D_m)^2}{2\sigma^2}\right] \frac{dD}{D}, \tag{6}$$

is specified by W, ρ_p, the mode diameter D_m, and the dispersion σ.

3.2.4 Modified gamma PSD (`'modGam'`)

The modified gamma PSD,

$$n(D)dD = \frac{6}{\pi} \frac{W}{\rho_p D_m^3} \frac{\mu^{\mu+4}}{\Gamma(\mu+4)} \frac{1}{D_m} \left(\frac{D}{D_m}\right)^{\mu} \exp\left(-\mu \frac{D}{D_m}\right) dD, \tag{7}$$

is specified by W, ρ_p, D_m, and μ. Liou (1992) argued that the modified gamma function with a mode radius of 4 μm ($D_m = 8\mu$m) and $\mu = 6$ reasonably represents a liquid cloud DSD as observed for fair weather cumulus.

3.2.5 Exponential PSD with a power-law mass spectrum (`'expaDb'`)

The PSD is assumed here to be exponential (Eq. [1]) but, instead of a constant ρ_p, the mass of a hydrometeor particle having the size D, $m(D)$, is defined as,

$$m(D) = aD^b, \tag{8}$$

or

$$\rho_p = \frac{6}{\pi} a D^{b-3}. \tag{9}$$

The slope parameter in this case is estimated to be

$$\lambda = \left[\frac{a n_0 \Gamma(b+1)}{W} \right]^{\frac{1}{b+1}}. \tag{10}$$

The PSD is specified by W, a, and b.

3.2.6 Heymsfield and Platt (1984) ice cloud parameterization (`'HyPl84'`)

The PSD model here is designed for ice clouds on the basis of the parameterization proposed by Heymsfield and Platt (1984) and is specified by W and local air temperature (ρ_p is computed internally). Two options are available, namely the "equal diameter" and "equal mass" schemes, depending on how to translate the particle non-sphericity into the Mie routines. In the equal diameter scheme, the ice particle mass is assumed to be distributed in a spherical volume with the diameter equal to the maximum crystal dimension. The equal mass scheme, on the other hand, describes an ice particle as a pure ice sphere with the same mass as elongated crystal.

The original Heymsfield and Platt (1984) parameterization is applicable only to ice crystal sizes larger than 20 μm. Smaller ice crystals, however, have a non-negligible contribution to the visible and infrared scattering properties. To overcome this limitation, the Heymsfield and Platt (1984) parameterization is extrapolated to smaller sizes using the exponential size distribution with $\lambda = 520$ cm^{-1} (Mitchell et al., 1996).

3.2.7 Two-moment exponential PSD (`'exp2mo'`)

A simplest two-moment bulk microphysical scheme is the exponential PSD (Eq. [1]) with the prognostic hydrometeor number concentration in [m^{-3}],

$$N_c = \int_0^\infty n_0 \exp(-\lambda D) dD = n_0/\lambda. \tag{11}$$

For given W and ρ_p, λ and n_0 are determined by

$$\lambda = \left(\frac{\pi N_c \rho_p}{W} \right)^{1/3} \tag{12}$$

and

$$n_0 = \lambda N_c. \tag{13}$$

3.2.8 Two-moment log-normal PSD (`'lgn2mo'`)

A two-moment scheme with the log-normal PSD (Eq. [3]) is defined with the mode diameter D_m internally calculated from a given N_c, i.e.,

$$D_m = \left(\frac{6W}{\pi \rho_p N_c} \right)^{1/3} \exp\left(-\frac{3}{2}\sigma^2 \right). \tag{14}$$

The PSD parameters to be specified are W, N_c, ρ_p, and the dispersion σ.

3.3 Build your own PSD model

If the desired PSD model is not found in the SDSU PSD library, the user can create a new PSD model. Instructions are provided in this section for registering a user-defined PSD model with the library.

The PSD library is highly flexible to accept arbitrary PSD models as a function of hydrometeor water content [g m^{-3}], temperature [K], hydrometeor number concentration [m^{-3}], and particle diameter [mm]. Any of these independent parameters may be omitted if not necessary. To add your own

PSD model to the library, create a new "case" entry in `psd_func` contained in `MCRPparameters.f90`. Here is an example:

```
!
! ** Gamma disribution with specified particle density, mu,
! ** and normalized intercept parameter.
!
          case( 'gammwN' ) ;&
          hydr%normaliz = .false. ;&
          hydr%indpndtT = .true. ;&
          hydr%twomomnt = .false. ;&
          density  = hydr%paramset(1) ;&
          nw       = hydr%paramset(2) ;&
          mu       = hydr%paramset(3) ;&
          D0       = 3.67*(lwc*1.e-3/(pi*density*nw))**0.25 * 1.e3  ;& ! [mm]
          psd_func = nw * 6*(3.67+mu)**(mu+4.)/(3.67**4*gammaf(mu+4.)) * &
                     (D/D0)**mu * exp(-(3.67+mu)*D/D0)
```

where

- The argument 'gammwN' is the function type identifier, which is a 6-letter character string designating the PSD model.
- The option `normaliz` controls the PSD normalization. The Mie subroutines internally normalize the PSD so that its third moment fits the given hydrometeor water content if `normaliz=.true.` This procedure is skipped when `normaliz=.false.` If your PSD is defined as normalized, set `normaliz=.false.` and save some computational time.
- Set `indpndtT` to be ".true." if your PSD parameters are independent of temperature. This option is aimed at accelerating the Mie LUT generator. When `indpndtT=.true.`, the loop over temperature in the visible/infrared component of the Mie LUT generator is skipped and replaced by copies of the same data sequence, because the water/ice refractive index at visible and infrared wavelengths are virtually independent of temperature.
- Make sure that `twomomnt` is ".true." for a two-moment PSD model and otherwise `twomomnt=.false.`.
- The user has their choice to determine what and how many PSD parameters (`paramset(1:nparams)`) define their PSD model. There is, however, one rule to follow: the first entry of the PSD parameters (`paramset(1)`) is reserved for the particle density ρ_p. If ρ_p is not specified by the user and computed internally, `paramset(1)` must be left blank in `microp_set.f90` so it is overwritten by `psd_func`. If you need 10 or more PSD parameters, change the definition of `npmax` in `MCRPparameters.f90` to a number large enough to accommodate `nparams`.
- Define your PSD following "`psd_func=`". If your PSD is not just a single analytic function, an external function module or subroutine may be employed as done in the `'HyPl84'` model, although it could considerably increase the computational time.

- If you make Mie LUTs based on your PSD model, create a directory of the same name as `functype` in `Mie-table/` (e.g., `Mie-table/gammwN/`).

Run-time access to the PSD library is controlled by `microp_set.f90` as described in section 2.1.4. Make sure that `functype`, `nparams`, and `paramset` are specified consistently with the PSD design in the library.

4 Mie Lookup Tables (LUTs)

The LUTs of the hydrometer radiative properties, hereafter called the Mie LUTs, are optionally available in order to improve the computational efficiency of the simulator unit. The Mie LUTs need to be created in advance off-line from the simulator unit. The Mie LUT generator, installed in `src/MieLUT/`, is introduced in this section.

4.1 User Interface

4.1.1 To run the Mie LUT generator

To run the Mie LUT generator, go to the directory `src/MieLUT/` and edit 3 lines defining the compiler and compiling options (lines 37-39) in `makefile` (see also section 2.1.1).

Once the compiler options and the customizations described next are all set, run the `makefile`,

```
% make mktab
```

and you will obtain the executable `mktab.x`. Run the Mie LUT generator by

```
% mktab.x
```

4.1.2 Microphysical model settings

The Mie LUT generator shares the PSD library and its run-time controller, `psd_func` and `microp_set` both in `src/simulator/hydromet/`, with the main simulator unit. See section 2.1.4 and section 3 for the usage and contents of the SDSU PSD library. Any changes made to these subroutines are reflected to the Mie LUT generator when compiled. The user may need to run `make clean` first when the shared subroutines are modified and compiled for the main simulator before compiled again for the Mie LUT generator.

4.1.3 Mie LUT generator settings (`paramLUT_set`)

LUT dimensions and channel frequencies/wavelengths can be customized by editing `paramLUT_set.f90`. The parameters reviewed in this subsection are all defined there.

LUT dimensions The Mie LUTs are formed by the three independent parameters of hydrometeor water content, temperature, and hydrometer number concentration. Hydrometer number concentration is required only for two-moment PSD models. The Mie LUT generator refers to the logical parameter `twomomnt` defined in the PSD library to find whether a given PSD model is single- or two-moment. The number concentration dimension is skipped for single-moment PSD models. Parameters `npts_*` and `pts_*(:)` define the LUT dimension size and a vector of reference points, respectively.

LUT dimension parameters (`paramLUT_set.f90`)

Parameter	Parameter type	Description
`npts_hydro`	integer	water content dimension size
`npts_temp`	integer	temperature dimension size
`npts_ncnc`	integer	number concentration dimension size
`pts_hydro`	real(1:npts_hydro)	hydrometeor water content [g m^{-3}]
`pts_temp_w`	real(1:npts_temp)	temperature for liquid hydrometeors [K]
`pts_temp_i`	real(1:npts_temp)	temperature for ice hydrometeors [K]
`pts_ncncc`	real(1:npts_ncnc)	cloud number concentration
`pts_ncncp`	real(1:npts_ncnc)	precipitation number concentration

By default `npts_hydro`, `npts_temp`, and `npts_ncnc` are 37, 12, and 30, respectively, and the reference points are defined as

```
pts_hydro = (/ 0.001, 0.002, 0.003, 0.004, 0.005, 0.006, &
               0.007, 0.008, 0.009, 0.01, 0.02, 0.03, 0.04, &
               0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.2, 0.3, &
               0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0, &
               4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0 /)
pts_temp_w = (/ 255., 260., 265., 270., 275., 280., &
                285., 290., 295., 300., 305., 310. /)
pts_temp_i = (/ 180., 190., 200., 210., 220., 230., &
                240., 250., 260., 270., 275., 280. /)
pts_ncncc = (/ 1.e+3, 3.e+3, 1.e+4, 2.e+4, 4.e+4, 6.e+4, 8.e+4, 1.e+5, &
               2.e+5, 4.e+5, 6.e+5, 8.e+5, 1.e+6, 2.e+6, 4.e+6, 6.e+6, &
               8.e+6, 1.e+7, 2.e+7, 4.e+7, 6.e+7, 8.e+7, 1.e+8, 2.e+8, &
               4.e+8, 6.e+8, 8.e+8, 1.e+9, 3.e+9, 1.e+10 /)
pts_ncncp = (/ 1.e+0, 3.e+0, 1.e+1, 2.e+1, 4.e+1, 6.e+1, 8.e+1, 1.e+2, &
               2.e+2, 4.e+2, 6.e+2, 8.e+2, 1.e+3, 2.e+3, 4.e+3, 6.e+3, &
               8.e+3, 1.e+4, 2.e+4, 4.e+4, 6.e+4, 8.e+4, 1.e+5, 2.e+5, &
               4.e+5, 6.e+5, 8.e+5, 1.e+6, 3.e+6, 1.e+7 /)
```

For single-moment PSD models, `npts_ncnc` is automatically reset to 1 and the number concentration vectors are ignored when the LUT is generated. The user can modify the LUT format by editing these parameters. The pre-set LUT dimensions above would work for most applications, although the range of number concentration might need to be modified depending on the two-moment microphysical scheme adopted.

Customizing the microwave radiometer simulator The Mie LUTs for microwave radiometer simulations are controlled by the parameters listed below.

Microwave radiometer simulator options (`paramLUT_set.f90`)

Parameter	Parameter type	Description
`microw`	logical	The Mie LUTs are created when true.
`nfreqMW`	integer	number of frequency channels
`freq_rdmtr`	real(1:nfreqMW)	channel frequencies [GHz]

By default `freq_rdmtr` is taken from the TMI instrumental design.

Customizing the radar simulator The Mie LUTs for radar simulations are controlled by the parameters listed below.

Radar simulator options (`paramLUT_set.f90`)

Parameter	Parameter type	Description
`radar`	logical	The Mie LUTs are created when true.
`nfreqRD`	integer	number of frequency channels
`freq_radar`	real(1:nfreqRD)	channel frequencies [GHz]

By default `freq_radar` follows the GMI DPR instrumental design.

Customizing the visible/IR simulator The Mie LUTs for visible/IR simulations controlled by the parameters listed below.

Visible/IR simulator options (`paramLUT_set.f90`)

Parameter	Parameter type	Description
`visIR`	logical	The Mie LUTs are created when true.
`nwvlnVI`	integer	number of wavelength channels
`wvln_visir`	real(1:nwvlnVI)	channel wavelengths [μm]

By default `wvln_visir` corresponds to the TRMM VIRS instrumental design.

4.2 LUT Nomenclature

The Mie LUT directories and files obey the following naming rules.

Directory tree The Mie LUTs are stored in `Mie-tables/(functype)/` separately by the PSD models. The sub-directory `(functype)` is named after the PSD function type identifier defined in the PSD library (see section 3) with the exception that `melt-p/` is reserved for the melting particle LUTs.

Individual LUT files Each Mie LUT file is named as "(*PSD model name*)-(*sensor index*)(*channel tag*).dat". The *PSD model name* is defined by the user in `microp_set.f90` in terms of hydrometeor species and PSD model-specific information (section 2.1.4). See the table below for *sensor index* and *channel tag*.

LUT file nomenclature

Simulator	*sensor index*	*channel tag*
Microwave radiometer	`mwr`	3-digit frequency [GHz] + '`G`'
Radar	`rdr`	3-digit frequency [GHz] + '`G`'
Visible/IR	`vir`	3-digit wavelength [nm] + '`N`' for $\lambda < 1$ μm
		3-digit wavelength [$10^{-1}\mu$m] + '`M`' for $\lambda \geq 1$ μm

For melting particles the naming rule is slightly modified to "(*snow PSD model name*)_(*rain PSD model name*)-(*sensor index*)(*channel tag*).dat". The adopted PSD parameters are recorded in ASCII files with the extension of `.info`. The file names are generated uniformly by the Mie LUT generator and by the simulator unit. The LUT nomenclature can be modified by editing `mktab_*.f90` in `src/MieLUT/`, `open_lut*.f90` in `src/simulator/*/`, and `mklab_lut.f90` in `src/simulator/hydromet/`.

5 Beam Convolution

The main simulator unit (section 2) has an option to write out the synthetic satellite data smoothed over a sensor field of view (FOV), convoluted with the Gaussian antenna pattern. This procedure, called beam convolution or antenna pattern convolution, is a critical step to compare the simulated and observed satellite data particularly for sensors having a large FOV size such as microwave radiometers. The SDSU package also contains a separate beam convolution code running off-line from the main simulator unit. This program allows the user to apply beam convolution later on, making it possible, for example, to test the simulated data with multiple sensors having different FOV sizes without running the main simulator unit every time for each sensor. The beam convolution routine is documented in this section.

5.1 User Interface

5.1.1 To run the beam convolution routine

To run the beam convolution routine, go to the directory `src/beamconv/` and edit 3 lines defining the compiler and compiling options (lines 29-31) in `makefile` (see also section 2.1.1).

Once the compiler options and the customizations described next are all set, run the `makefile`,

```
% make bmcv
```

and you will obtain the executable `bmcv.x`. Run the beam convolution routine by

```
% bmcv.x
```

5.1.2 General settings (`param_set_bmcv`)

The spatial dimension sizes `ngridx`, `ngridy`, and `nlyr` and the grid intervals `delta_x` and `delta_y` are read in from the `*.info` files in `outputs/` written out by the main simulator unit. The beam convolution routine next ingests the FOV sizes provided by the user (`param_set_bmcv.f90` in `src/beamconv/`).

FOV sizes (`param_set_bmcv.f90`)

Parameter	Parameter type	Description
`fov_ct_microw`	real(1:mxfreq_microw)	microwave radiometer crosstrack FOV FWHM [km]
`fov_dt_microw`	real(1:mxfreq_microw)	microwave radiometer downtrack FOV FWHM [km]
`fov_ct_radar`	real(1:mxfreq_radar)	radar crosstrack FOV FWHM [km]
`fov_dt_radar`	real(1:mxfreq_radar)	radar downtrack FOV FWHM [km]
`fov_ct_visir`	real(1:mxfreq_visir)	visible/IR imager crosstrack FOV FWHM [km]
`fov_dt_visir`	real(1:mxfreq_visir)	visible/IR imager downtrack FOV FWHM [km]

The beam pattern is approximated by a two-dimensional Gaussian function characterized by the crosstrack and downtrack FOV sizes or the full widths at half maximum (FWHM). The crosstrack and downtrack directions are defined as the x and y directions, respectively, so that the microwave slant path is parallel to the downtrack direction when `slant_path=.true.` (section 2.1.2). The present

beam convolution routine only smoothes the data and does not alter the spatial sampling (or grid dimensions) of the original data.

The output files are written out into `outputs/` and have the extension of `.fov`. See section 2.2 for the output file format.

6 Citations

It would be appreciated to acknowledge any of these references that are relevant to your work when it appears in publications. The complete list of references is found in the end of this document. See section 2.3 for more information.

SDSU Citations

Simulator	Citations
Microwave radiometer simulator	Kummerow (1993)
Radar simulator	Masunaga and Kummerow (2005)
Visible/IR simulator	Nakajima et al. (2003)

A short aritcle on the SDSU is to appear in the In Box section of Bulletin of the American Meteorological Society. Preprint in PDF is available from the SDSU web site.

7 Contact

Please send your feedback to Hiro Masunaga at masunaga@hyarc.nagoya-u.ac.jp. Latest updates, whenever available, are posted on the SDSU www site.

8 Acknowledgments

I am deeply grateful to Chris Kummerow (Colorado State Univ.), Terry Nakajima (Univ. of Tokyo), Peter Bauer (ECMWF), Bill Olson (NASA GSFC), Wei-Kuo Tao (NASA GSFC), Toshi Matsui (NASA GSFC), Takashi Y. Nakajima (Tokai Univ.), and Miho Sekiguchi (Tokyo Univ. of Marine Sci. Tech.) for their support to the SDSU. Their contributions constitute crucial elements of the SDSU. I am also indebted to the users who sent me feedbacks for upgrading the SDSU.

References

Bauer, P., 2001: Microwave radiative transfer simulation in clouds: Including a melting layer in cloud model bulk hydrometeor distributions. *Atmos. Res.*, **57**, 9–30.

Bauer, P., E. Moreau, F. Chevallier, and U. O'Keefe, 2006: Multiple-scattering microwave radiative transfer for data assimilation applications. *Quart. J. Roy. Meteor. Soc.*, **132**, 1259–1281.

Bringi, V. N. and V. Chandrasekar, 2001: *Polarimetric Doppler weather radar*. Cambridge Univ. Press, 636 pp.

Heymsfield, A. J. and C. M. R. Platt, 1984: A parameterization of the particle size spectrum of ice clouds in terms of the ambient temperature and the ice water content. *J. Climate*, **10**, 647–661.

Kummerow, C. D., 1993: On the accuracy of the Eddington approximation for radiative transfer in the microwave frequencies. *J. Geophys. Res.*, **98**, 2757–2765.

Liou, K.-N., 1992: *Radiation and cloud processes in the atmosphere: theory, observation and modeling*. Oxford Univ. Press, 487 pp.

Masunaga, H. and C. D. Kummerow, 2005: Combined radar and radiometer analysis of precipitation profiles for a parametric retrieval algorithm. *J. Atmos. Oceanic Technol.*, **22**, 909–929.

Masunaga, H., M. Satoh, and H. Miura, 2008: A joint satellite and global cloud-resolving model analysis of a Madden-Julian Oscillation event: Model diagnosis. *J. Geophys. Res.*, **113**, D17210, doi:10.1029/2008JD009986.

Mitchell, D. L., S. K. Chai, Y. Liu, A. H. Heymsfield, and Y. Dong, 1996: Modeling cirrus clouds. part I: treatment of bimodal size spectra and case study analysis. *J. Atmos. Sci.*, **53**, 2952–2966.

Nakajima, T. and M. Tanaka, 1986: Matrix formulations for the transfer of solar radiation in a plane-parallel scattering atmosphere. *J. Quant. Spec. Rad. Trans.*, **35**, 13–21.

Nakajima, T. and M. Tanaka, 1988: Algorithms for radiative intensity calculations in moderately thick atmospheres using a truncation approximation. *J. Quant. Spec. Rad. Trans.*, **40**, 51–69.

Nakajima, T. Y., H. Murakami, M. Hori, T. Nakajima, T. Aoki, T. Oishi, and A. Tanaka, 2003: Efficient use of an improved radiative transfer code to simulate near-global distributions of satellite-measured radiances. *Appl. Optics*, **42**, 3460–3471.

Olson, W. W., P. Bauer, N. F. Viltard, D. E. Johnson, W.-K. Tao, R. Meneghini, and L. Liao, 2001: A melting-layer model for passive/active microwave remote sensing applications. part i: Model formulation and comparison with observations. *J. Appl. Meteor.*, **40**, 1145–1163.

Sekiguchi, M. and T. Nakajima, 2008: A k-distribution-based radiation code and its computational optimization for an atmospheric general circulation model. *J. Quant. Spec. Rad. Trans.*, **109**, 2779–2793.

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.

Tao, W.-K. and J. Simpson, 1993: Goddard Cumulus Ensemble Model. part I: model description. *Terr. Atmos. Oceanic Sci.*, **4**, 35–72.