System for Atmospheric Modeling

SAM

Version 6.11.3

User's Guide

Marat Khairoutdinov

Marat.Khairoutdinov@stonybrook.edu

Updated: Dec 4, 2018

Introduction

The System for Atmospheric Modeling (SAM) is a non-hydrostatic anelastic model that can be used to simulate cloudy atmospheres in a wide range of scales, from boundary-layer turbulence to hurricanes. It can be configured as a Large-Eddy Simulation (LES) model to study cloudy or cloud-free boundary layers, or as a Cloud-Resolving Model (CRM) to study deep convective clouds and meso-scale cloud systems. The SAM has been successfully ported to many different computing platforms including massively parallel supercomputers. This document provides a brief, but, hopefully, sufficient guide for a new user to compile, set-up a case, and run SAM. The details of theoretical formulation are given by Khairoutdinov and Randall (2003, *J. Atmos. Sci*., pp. 607-625). The SAM's Fortran code, especially its dynamical core, mostly follows the paper, although there have been a few changes, bug fixes, and extensions introduced to the code over the years, most are not yet formally documented.

Files and Directories

A listing of the SAM root directory may look as the following (your model distribution may be slightly different but the essential files should be similar):

```
fen 13:56 ~/SAM6.9.0 > ls
ARM0003/   Build*        DYCOMS_RF01/  GATE_IDEAL/  Makefile  SRC/        UTIL/
ARM9707/   CaseName      DYCOMS_RF02/  GCSSARM/     RUNDATA/  TEST/
ASTEX209/  Changes_log/  GABLS/        KWAJEX/      SCRIPTS/  TOGA/
BOMEX/     DOC/          GATE/         LBA/         SMOKE/    TOGA_LONG/
```

The directory SRC contains all the source-files of the model. The directories ARM0003, ARM9707, BOMEX, GATE, ASTEX209, TOGA, etc. are so-called case-directories. Each case-directory contains files with initial and forcing data necessary to run a particular simulation case. A user can easily create new case-directories. SAM executable is usually built in and run from the root directory with the output usually written elsewhere using symbolic links. The file CaseName sets the name of the current (run-time) case-directory.

The file Build is the C-shell script used to build the SAM executable and create additional output directories as well as symbolic links. It sets up several environmental variables and calls a suitable version of the make-utility to compile

the code. Note that despite presence of a Makefile, the make or gmake utility should not be directly used to compile SAM.

The UTIL directory contains the source files to build utilities that convert the model output files into the netcdf files among other things. Unlike SAM, the make or gmake utilities should be used to compile many conversion utilities found in the UTIL/SRC directory.

The SCRIPTS directory contains some useful C-shell scripts and NCL (NCAR Command Language) scripts that are supplied by the author as a courtesy, but currently with no explicit documentation. Note that all SAM's output produces files that can easily be converted to netcdf format using the conversion utilities in UTIL directory.

Building executable

In order to build the SAM executable, the following steps are taken:

Step 1: Editing Build script

If SAM is compiled for the first time on a particular computer, you need to edit the Build script to set a few environmental variables required for the compilation process.

First, the variable SAM_SCR should be defined. The SAM executable is always run from in the SAM root directory, which is usually located in a user's home directory where it cannot be accidently erased by a scratch-disk sweeper. However, user home directories usually have quite limited disk space. Typically, on supercomputer systems, a user has access to a much larger 'scratch' disk space where the output is typically stored. The SAM_SCR variable points to that remote directory where all the output files generated by the model will be written. For example, the following line defines a directory with the same name as the SAM root directory in the scratch-disk directory /scratch/marat:

```
setenv SAM_SCR /scratch/marat/SAM6.9.1
```

It is always a good practice to use different output directories for different SAM home directories. When running on a personal computer or UNIX-cluster, the root directory can reside on a personal disk with sufficiently large space to write the

model output. In that case, a user can set the SAM_SCR variable to point to the SAM root directory. Generally, an explicit path like ./ could be used; to be safe, the following line can be used to generate a full path (note the difference between Unix command-execution back-quote ` and strings quote ‘):

setenv SAM_SCR `pwd`

When the Build script is executed, the directory defined by the SAM_SCR variable is created, unless, of course, it already exists. Several subdirectories are also created. These subdirectories are OBJ, OUT_STAT, OUT_2D, OUT_3D, OUT_MOMENTS, OUT_MOVIES, and RESTART. In the case when the SAM_SCR variable points to a directory different from the SAM root directory, the corresponding symbolic links to the above subdirectories are created in the root directory.

The RESTART directory is used to store the files necessary for the model continuation of a previous run or a new branch run using earlier run. The OBJ is the directory where all the object files and dependency files are written. The OUT_* are the directories where the model output files are written. Each output directory contains specific groups of files:

OUT_STAT	time-height horizontally averaged statistics;
OUT_2D	2D x-y fields;
OUT_3D	3D snapshots of various fields;
OUT_MOMENTS	3D fields of various statistics on degraded grid;
OUT_MOVIES	files used for visualization and movies.

The variable ADV_DIR defines at the compile time which monotonic and positive definite scheme is used for advection of all scalars in the model. There are currently two choices, the original SAM’s advection scheme based on Smolarkiewicz’ MPDATA scheme with monotonic corrector, and a recently developed by one of the SAM users flux-corrected transport scheme ULTIMATE-MACHO based on 5^{th}-order advection scheme (*Yamaguchi et al. 2011, Mon. Wea. Rev., pp 3248–3264*). The later scheme is expected to be less numerically diffusive than MPDATA, although there has been relatively little experience running SAM with that scheme since it became available starting with version 6.9. The computation expense of using either scheme is quite similar.

The variable RAD_DIR defines a specific radiation package that will be used at compile time. Currently, two radiation packages are supplied with SAM. These are

the CAM3 radiation (directory SRC/RAD_CAM) and RRTM (directory SRC/RAD_RRTM) radiation packages. Both packages are exported from the NCAR Community Climate System Model (CCSM).

The variable MICRO_DIR defines a specific cloud-microphysics package that will be used at compile time. There are currently three choices for the cloud microphysics: 1) the original SAM's single-moment microphysics (directory SRC/ MICRO_SAM1MOM), 2) double-moment microphysics (Morrison et al. 2005; directory SRC/MICRO_M2005), directory SRC/MICRO_THOM (WRF's Thompson microphysics), and 4) simple drizzle microphysics (directory MICRO_DRIZZLE), which is a simplified (drizzle only) version of Khairoutdinov and Kogan (2000) microphysics. Note that a user can easily develop his/her own or port some existing microphysics package using a microphysics-template directory MICRO_TEMPLATE. The choice of microphysics can significantly affect the wall-clock computer time because of different number of the microphysics prognostic variables. Transporting the microphysics variables using positive definite monotonic schemes can constitute main cost of running the model. The standard SAM1MOM microphysics is the least expensive option. The M2005 microphysics is the most comprehensive.

The variable SGS_DIR defines a subgrid-scale (unresolved) model used by SAM. The original SAM's SGS closure based on 1.5-order closure (prognostic SGS turbulent kinetic energy (TKE) and the stationary form of it called the Smagorinsky scheme is in the SRC/SGS_TKE directory. Hopefully, in the future there will be more choices for the SGS model. Note that any user can easily develop his/her own or port some existing SGS package to SAM. The procedure is described in the template directory SGS_TEMPLATE.

Finally, the GNU-make utility command (gmake or gnumake) should be specified. It is also possible to use any make-utility other than GNU-make as long as it supports the VPATH environmental variable.

Step 2: Editing Makefile

The Makefile supplied with the model has already been setup to compile on several platforms. If your platform is already supported, you would still need to check the corresponding subsection of the Makefile to see if the compiler names, options and library paths are correct.

If you don't see your platform in the Makefile, you would have to create your own subsection. Use other subsections as templates. To find out the name of your system that should be used in ifeq ($(PLATFORM),... statement, execute the *uname* command at the command prompt (if using UNIX-like system).

The Makefile doesn't have any source-file names and dependencies hardcoded (imaging, people still do that even for well-known operational models!). Instead, all the file dependencies needed by the make are generated automatically using the scripts written in perl, which I shamelessly borrowed from the CCSM distribution and adopted for SAM (CCSM programmers are amazing!). Therefore, compilation process requires that your system has the Perl scripting enabled. By default, it is assumed that the Perl executable perl is located in the /usr/bin directory. If a directory other than /usr/bin is used in your system, you would need to edit the first lines of each perl-script to set the correct path. The scripts are located in SRC/SCRIPTS directory. The automatic generation of file-dependencies means that you can add your own source files to SAM and they will be compiled and linked automatically!

If you successfully added a new platform subsection that compiles the model, please email your Makefile to the author, so that your platform's Makefile additions are included in the future releases of the model, which would make life of future SAM users a bit easier.

Step 3: Editing SRC/domain.f90

Before compiling SAM by running the Build script, you should set the dimensions of the computational grid set in the file SRC/domain.f90. The variables *nx_gl*, *ny_gl*, and *nz_gl* specify the domain sizes in x, y, and vertical z directions, respectively. When developing SAM, I consciously avoided dynamical allocation of arrays (I forgot the exact reason why, probably to aid compiler optimization of the code), so each SAM executable can run on a predefined grid only.

Because of particular Fast-Fourier-Transform subroutines, the horizontal domain dimensions *nx_gl* and *ny_gl* can only be the factors of 2, 3, and 5. For example, nx_gl=240 is allowed (240=2*2*2*2*3*5), but nx_gl=168 is not because of of the factors is 7 (168=2*2*2*7*3). The performance of the model may strongly depend on the choice of the domain dimensions because of issues like cache memory sizes, etc. It is, therefore, a good idea to test the model efficiency using several dimensions in a narrow range. Usually, the power-of-two dimensions, like 64, 128, 512, are the safest bet. If you set the *YES3D* variable to 0, the model will run as a 2-

D model (x-z domain). In that case, you still need to set *ny_gl* and *nsubdomains_y* explicitly to 1. Don't also forget to use high levels of compiler optimization, although carefully as compilers are notorious for inserting incorrectly optimized code. If the model crashes for unexplained reason or produces strange looking results, one of the first thing to do before plunging into debugging mode is to reduce optimization level or switch it off altogether to see if the model run normal. If it does, then compiler optimization bug is the likely culprit (it happens more often than you think).

The SAM uses the Message-Passing Interface (MPI) protocol to run on parallel computers. If your system has multiple processors, and MPI library is available, the model is easily set up to run on multiple processors. Currently, there is no OpenMP support for multithreading, so SAM is a pure MPI model. For parallel runs, the model domain is divided into equally sized subdomains of the same height as the global domain, one subdomain per processor. The number of subdomains in x and y directions are given by the *nsubdomains_x* and *nsubdomains_y*, respectively, so that the total number of processors equals to *nsubdomains_x*nsubdomains_y*.

The rules for choosing the number of subdomains and, hence, the total number of processors are fairly simple. The global domain dimensions *nx_gl* and *ny_gl* should be divisible without remainder by *nsubdomains_x* and *nsubdomains_y*, respectively. If either *nx_gl* or *ny_gl* is not divisible without remainder by the total number of processors, then the number of pressure levels nz_gl should be divisible by the total number of processors. If both *nx_gl* and *ny_gl* are divisible without remainder by the total number of processors, then an arbitrary number of vertical levels can be used. Generally, the number of grid points in a horizontal direction is larger than the number of the vertical levels. Therefore, the maximum number of processors that can be used for a grid of given size depends mostly on horizontal dimensions. For example, for 512 x 512 x 64 grid in x, y, and z, the maximum number of processors that can be used is 512 (e.g., *nsubdomains_x=16, nsubdomains_y=32*), while for 256x512x64 grid it is 256. For the 2-D version of SAM (don't forget to set YES3D=0 in that case), the number of levels should always be divisible by the total number of processors; for example, for 64 levels, you can use, 4, 8, 32, and 64 processors, bit not 20.

If *nsubdomains_x* and *nsubdomains_y* are *both* set to 1, the compiled code can be run as a serial job that is using a single processor (back to 80s!). The MPI is not used in this case, but the MPI libraries would still be linked as the model doesn't use the C-preprocessor code-brackets and, therefore, there is no direct way to remove all the MPI related code during compilation. If a single-processor set-up is

used and there are no MPI libraries installed, you could avoid that compilation problem by replacing task_util_MPI.f90 with task_util_NOMPI.f90 (both files found in SRC directory). (The automated dependency-builder script for the make procedure looks only at the files that end with .f, .f90,.f95, and .c. Therefore, to avoid compilation of task_util_MPI.f90 together with task_util_NOMPI.f90, you should change its extension .f90 to, for instance .f9000, like task_util_MPI.f9000, while task_util_NOMPI.f9000 should be renamed to task_util_NOMPI.f90.)

Finally, domain.f90 also sets the number of tracer arrays *ntracers*. If you don't want to transport any tracers around, just leave ntracers = 0 .

Step 4: Compiling SAM

After the first three steps are completed, it is time to compile the code. Just run the edited Build script by executing

> Build

or, if failed and using csh or tcsh, by

> csh Build

Upon successful outcome, your base SAM directory would look similar to the following:

```
fen 14:44 ~/SAM6.9.0 > ls
ARM0003/   Changes_log/  GATE_IDEAL/  OUT_2D/       RUNDATA/                    TOGA/
ARM9707/   DOC/          GCSSARM/     OUT_3D/       SAM_RAD_CAM_MICRO_SAM1MOM*  TOGA_LONG/
ASTEX209/  DYCOMS_RF01/  KWAJEX/      OUT_MOMENTS/  SCRIPTS/                    UTIL/
BOMEX/     DYCOMS_RF02/  LBA/         OUT_MOVIES/   SMOKE/
Build*     GABLS/        Makefile     OUT_STAT/     SRC/
CaseName   GATE/         OBJ/         RESTART/      TEST/
```

The long name of the executable is to provide an easy way of making sure that the desirable advection scheme, radiation and microphysics packages are used at the run-time.

By default, SAM is compiled as a single-precision (4-byte real); however, it can be safely run in double precision mode (8-byte real) by specifying compiler option to set the default size of real to 8 bytes. Regardless of the choice, all the output files will still be written using real 4-byte numbers.

The executable should be run using exactly the number of processors implied by the SRC/domain.inc file. For example, if *nsubdomains_x* = 2 and *nsubdomains_y* = 4, then the number of the requested processors should always be equal to 8. On a system that uses mpirun to launch parallel jobs, the command would be similar to

> mpirun -np 8 SAM_RAD_CAM_MICRO_SAM1MOM

To run in the background rather than interactively and using a file for the printout (e.g., with the name out) instead of a terminal, you could use a slightly expanded command:

> mpirun -np 8 SAM_RAD_CAM_MICRO_SAM1MOM < dev/null > out &

There are other ways of launching parallel jobs, for example batch systems. However, these issues are system dependent and, therefore, are beyond the scope of this document.

We are almost done. But before you run the model, you need to specify the CASE!

Setting up a Case

To choose a case, you need to edit the *CaseName* file, which simply contains the name of the corresponding case-directory. Each case-directory should always contain the following files in ASCII format:

prm - the file containing the namelists for the model parameters. The namelist variables are listed further down the text.

snd – the file containing the soundings that are used to specify initial sounding, and, in the absence of the lsf file, the soundings that the model solution can be nudged to. The snd file contains soundings for several time points. The variables are height and/or pressure, potential temperature, water vapor mixing ratio, and zonal and meridional velocity components. The model will use linear interpolation in time and in the vertical to produce the initial sounding; therefore, minimum two time samples are needed. If only one sounding is available, still, the snd file should contain at least two identical soundings separated in time for the model to initialize correctly (see the BOMEX case, for example).

While the files above are absolutely essential to run SAM, the following files are optional in the sense that, in principle, the model can be configured to run without them:

grd - specifies the vertical grid unless dz_const=.true., in which case the constant dz vertical grid step is set by the namelist PARAMETERS. The model uses Arakawa –C staggered grid. The file contains the height of scalar mid-levels (in meters), not interface levels, starting from the levels near the surface. The surface is the first interface level; hence, the first mid-level scalar level is between the surface and second interface levels. Note that grd file specifies mid-levels only. The interface levels will be computed automatically. If you want to specify vertical grid that consists of the layers of the same thickness, say, 10 m, then the grd file would contain the following heights:

5.
15.
25.

You don't have to specify all levels implied by the nz_gl. though. The algorithm will continue building the grid levels above the last one specified in the grd file assuming the grid spacing between two last specified levels to be the spacing between all the grid levels above.

lsf - large-scale forcing file. It specifies the large-scale temperature and vapor tendencies, as well as large-scale wind that simulated mean wind may be nudged (relaxed) to. To use that file, the prm file should set dolargescale=.true.

sfc – contains the time evolution of prescribed SST, and surface fluxes. To use that file, the prm file should set dosfcforcing=.true.

rad - prescribed radiation heating rates. To use that file, the prm file should set doradforcing=.true.

Important note: starting from SAM 6.7, there is an alternative way of setting the case, that is using a standard NCAR SCAM's (CAM single-column model) input file in netcdf format. See doscamiopdata namelist parameter. Also, note that starting from version 10, lst data file has been moved to RUNDATA directory and should not be edited.

NAMELIST VARIABLES

File **prm** contains the namelist variables that control the model execution and also set some physical parameters. The parameters in PARAMETERS namelist as well as their default values can be found in SRC/params.f90 and SRC/grid.f90 files. There are also separate namelists in microphysics modules.

Namelist PARAMETERS

Run controls

caseid - the run identifying name. All the names of output files will contain this string. Put as much information as possible into this string for easy identification of output files later.
Example: caseid ='128x128x112_80m_40m_2s_LES'

nrestart - run type
nrestart = 0 - initial run
nrestart = 1 - restart (continue) previous run; the namelist parameters will be ignored except for nstop, nprint, nstat, and nstatfrq, and 2D and 3D output parameters
nrestart = 2 - branch restart; the run will restart using restart files from another run. You need to specify both the case_restart and caseid_restart. The namelist parameters and forcing fields will be overwritten. Basically, this is a way to start a new run from already spun-up model run.

Freshly written restart files will overwrite older restart-files. So, if your restart files are corrupted for any reason when being written, you won't be able to restart the run, it will be lost and have to be started the from the very beginning. For very expensive long runs, it is advisable to back-up the restart files before each restart. Also, if it is a new run (nrestart=0), the corresponding .stat file and other output files in OUT_2D and OUT_3D directories should not exist. If they do, the model will crash trying to overwrite those files. This is done to avoid accidental overwrite of the older (potentially valuable) output files if accidently the caseid string has not been changed for a new run.

caseid_restart – the branch-root identification caseid for branch restart
case_restart – the branch-root identification case (case directory name) for branch restart. Note that both case_restart and caseid_restart should be specified

restart_sep – if .true., a separate restart file for each MPI task will be written. By default (restart_sep=.false.), the serial restart files are written (as if he model is run on a serial computer rather than parallel). If very large domain is chosen to run using massive number of processors, and the size of the serial restart files could be too for a file system or mass-storage systems. Also, writing one large restart file from the MPI task 0 when many processors are used can be quite a communication bottleneck in the overall performance and hence should be avoided. Writing a separate restart file for each MPI task is the solution.

nrestart_skip – skip writing restart files nrestart_skip times. By default (nrestart_skip=0), the restart files will be written every time the horizontally mean statistics file is updated (see nstat parameter); however, it may be inefficient (slow) if the statistics file (written to OUT_STAT directory) is written often and the domain is large. Therefore, this parameters allows one to skip writing the restart files. For example, nrestart_skip=11 will write the restart every 12^{th} time the statistics file is written. At the end of the run, the restart will be always written regardless of the value of this parameter.

nstop - maximum number of time steps to complete the case. This is a required parameter. Note that nstop is not written to the restart file, so it can be changed before each restart. Therefore, this parameter should always be present in the namelist.

nelapse – optional parameter to set the elapsed time (in time steps) to run before stopping (optional parameter). Convenient for automatic job resubmission on computers that have CPU time limit. After computing nelapse time steps, the model stops gracefully after saving the restart files.

nelapsemin - the run should terminate at the end of nelapsemin minutes of runtime. Note that the run will only terminate after the writing of statistics (and 2D output if save2Davg==.true.). This is useful if there is a timelimit on your queue.

day0 - The initial calendar day of the run. This time will be used for interpolation of the initial sounding read from snd file. Also, the current model time during the run will be computed using day0. Remember that the solar radiation depends on the

current calendar day. If solar radiation is prescribed, day0 can be set to 0 as the absolute value of the calendar day becomes in that case unimportant.

ncycle_max – number of maximum divisions of main timestep (see **dt**)

Output controls

The following set of namelist parameters control the way the output files are written. Remember that they all will be written into the appropriate OUT_* directory.

In addition to the horizontally averaged statistics, the model can optionally output the snapshots of 3-D fields (2-D fields in the case of 2-D model), as well as the snapshots of 2-D x-y fields (1-D x fields in the case of 2-D model). For the list of the variables written in these files check the SRC/write_fields3D.f90 and SRC/write_fields2D.f90 files.

By default, to save storage space, since some of the files can be quite bulky, the data is written in compressed form using 2-byte integers. However, there is an option to write using floating point format, which would take 4 bytes per each element of the output. In either case, the datasets can be converted into the NETCDF files using the conversion utilities found in UTIL directory.

nprint - frequency of short printouts (in time steps) performed during the execution. By default, the printing is done to the standard output. The printouts contain minimum and maximum values of various fields among some other information. The important number to watch here is the max and min values of the mass divergence (variable div), which is theoretically zero for anelastic model. As long as the absolute maximum of the divergence doesn't exceed 1e-7, there is no reason for concern. However, if that number is larger, or significantly larger than, say, 1.e-4, something is wrong and the model should not be running. Check the correctness of the domain dimensions to see if they have legal size. Also, sometimes, certain set of compiler options can produce erroneous code, try to compile with other smaller optimization or with no optimization at all, and see if the divergence in a new run is within the acceptable limits.

nstat - number of time steps used for averaging the statistics output. Note that the restart files will be written every nstat steps unless nrestart_skip is set.

nstatfrq - number of samples collected over the nstat steps.

Example: you use 10 sec time step, and the statistics should be computed and averaged over one hour intervals, collecting samples separated by 2 minutes. Then, you will need to set
nstat = 360
nstatfrq = 30

doSAMconditionals – if .true., core (suffix: COR) and downdraft core (suffix: CDN) averages will be outputted to *.stat file. Note that the number of fields can be quite large and would increase the stat-file size considerably.

dosatupdnconditionals – if .true., cloudy updraft (suffix: SUP), cloudy downdraft (suffix: SDN) and cloud-free (suffix: ENV for environment) averages will also be output tp *.stat file. The threshold for cloud is 10^(-5) kg/kg, so that there may be small amounts of cloud water and ice in the environmental air. Note that the number of fields can be quite large and would increase the stat-file size considerably.

doisccp – if .true., ISCCP satellite simulator will be on and a *.isccp file containing the isccp histograms (if dosimfilesout=.true.) will be written in OUT_STAT directory as well as several bulk statistics into the *.stat file. The file can be converted into netcdf format with isccptonc utility found in UTIL.

domodis – if .true., MODIS satellite simulator will be on and a *.modis file containing the isccp histograms (if dosimfilesout=.true.) will be written in OUT_STAT directory as well as several bulk statistics into the *.stat file

domisr – if .true., MISR satellite simulator will be on and a *.misr file containing the isccp histograms (if dosimfilesout=.true.) will be written in OUT_STAT directory as well as several bulk statistics into the *.stat file

dosimfilesout – if .true., the satellite simulators' histograms will be outputted into separate files written in OUT_STAT directory.

output_sep – if .true., all the 2D and 3D output will be written separately for each MPI process, that is separately for each subdomain. This is done for speed when massively parallel computers are used to avoid communication bottleneck. Each 2D and 3D file for each subdomain will be appended with the underscore and the rank (index) of the subdomain that wrote it. So, if running on 2048 processors, 2048 times as many files will be written. The files should be glued together after the run using some a suitable utility found in UTIL directory (for example, com3D_sep2one for *.com3D files). The SCRIPTS/FILES directory contains some

csh scripts that help to automate the process when many files need to be glued together.

The following parameters control writing 2-D information (1-D in case of 2-D model). Those are horizontal fields (x,y). Note that by default the time samples will be appended to each other in the same file. In the case of compressed data, the file extension will be .2Dcom, in the case of binary output - .2Dbin. By default, the compressed 2-byte output will be used. You could postprocess these files into netcdf format using 2Dcom2nc and 2Dbin2nc utilities, respectively, found in UTIL directory.

nsave2D: sampling period of 2D fields in model steps. The fields will be averaged over that sampling period if save2Davg=.true.; otherwise, the saved fields will be snapshots at the end of each sampling period

nsave2Dstart: the time step to start sampling 2D data data. If this time step is larger than the nstop parameter, 2-D output files will never be written.

nsave2Dend: the time step to finish sampling 2-D data. No 2D data will be sampled after the time step larger than nsave2Dend.

save2Dbin – if .true., the output will be saved as floating-point binary.

save2Dsep - if .true., each 2-D horizonatal snapshot or sampling period average will be saved into a separate file.

save2Davg - if .true., each 2-D field will be averaged over sampling period. Otherwise, the snapshots collected at the very end of the sampling period will be written.

dogzip2D if true, the output file will be further compressed using gzip utility (can be very slow for very big files; the model will be standing by waiting for the compression to finish)

The following parameters control writing the snapshots of the 3-D fields. They are similar to 2-D parameters with the exception that the 3-D data will always be

written as a snapshot, one per file, so no time averaging is done. The files will contain the time-step stamp in their name. In the case of 2-byte compressed data, the file extension will be .com3D, in the case of binary output - .bin3D. The files can be postprocessed into netcdf format using com3D2nc (com2D2nc) and bin3D2nc (bin2D2nc) utilities, respectively, found in UTIL directory. By default, the 2-byte compression will be used.

nsave3D: frequency of writing 3D snapshots in time steps.

nsave3Dstart: the time step to start writing 3D snapshots. If this timestep is larger than the nstop parameter, no 3-D output will be written.

nsave3Dend: the time step to finish writing 3-D snapshots. No data will be written after the time step larger than nsave3Dend.

save3Dbin – if .true., the output will be saved in the binary format.

save3Dsep - if .true., each 3-D snapshot will be saved into separate file. Always true for 3-D model output; however, for 2D runs, the x-z snapshots will be saved to a single file unless using this parameter.

dogzip3D if true the output file will be further compressed using gzip utility (can be very slow for very big files)

The following parameters control saving the 3-D moment-statistics fields. The fields will be averaged to a coarser grid as specified by the navgmom_x and navgmom_y variables set in domain.f90. See Changes_log/README.UUmods file for more details.

The data will represent snapshots of fields, so no time averaging is done. The files will contain the time step stamp in their name. In the case of compressed data, the file extension will be .com3D, in the case of binary output - .bin3D. If the 2-D snapshots are appended into one file, the extensions will be .com2D and bin2D respectively. You could postprocess the files into the NETCDF file using com3D2nc (com2D2nc) and bin3D2nc (bin2D2nc) utilities, respectively, found in UTIL directory.

nstatmom: frequency of writing (steps).

nstatmomstart: the time step to start sampling moment statistics. If this timestep is larger than the nstop parameter, no stat-moment output file will be written.

nstatmomend: the time step to finish saving stat-moment data. No data will be written after the time step larger than nstatmomend.

savemombin – if .true., the output will not be compressed, but rather will be saved in the binary format.

savemomsep - each 3-D snapshot will be saved into separate file. Always true for 3-D model output; however, for 2D runs, the x-z snapshots can be saved to a single file using this parameter.

The following parameters control saving 1-byte data used for animations (movies). All movie files for separately for several 2-D (x-y) fields are written into OUT_MOVIES directory. The code can be found in movies.f90. The compression uses the field limits set by namelist MOVIES. Note that each processor writes its own movie file, which then need to be glued together. There is a utility called glue_movie_raw in UTIL directory that can glue together all these files and produces a single RAW format *.raw file for each 2-D field. One can convert the *.raw file into animated gif-movie using Image Magick convert command like:

convert -depth 8 -size nx_gl x ny_gl file.raw file.gif

where nx_gl and ny_gl are the domain sizes in x and y. You could also convert the raw file into video file in some format rather than to animated-gif, but it is beyond the scope of this document and vast internet resources on the subject exist.

The movie creation is controlled by the following namelist parameters:

nmovie: frequency of saving (time steps).

nmoviestart: the time step to start saving moment statistics. If this timestep is large than the nstop parameter, no stat-moment output file will be written.

nmovieend: the time step to finish saving stat-moment data. No data will be appended after the time step larger than nmovieend.

Physics controls

dx - constant horizontal grid spacing (in m) in the x direction (West-East).
dy - constant horizontal grid spacing (in m) in the y direction (South-North).
dz – constant vertical grid spacing (in m). Works only in pair with the dz_constant=.true.; Otherwise, the vertical grid is set by the grd file.

dz_constant – if .true., the vertical grid step is constant and set by the variable dz; no grd file is then needed to run the case.

dt - time step (in seconds). When setting dt, try to set it such that the anticipated maximum advective Courant number is not above 0.5. The dynamical core uses Adams-Bashforth time scheme with variable time step, so in the event of high Courant number the time step will be automatically divided by 2, so the time step will be subcycled twice, which adds to the expense. If halving the time step is still not enough to keep the model linearly stable, the time step will be further divided by two. This process will be repeated until maximum set by **ncycle_max** parameter. At that point, the model is likely unstable, and the run will abort. It is also possible that rather than blaming the model, it is you who chose dt to be too large, so the model easily ends up dividing it more than ncycle_max times when strong updrafts or hurricane winds develop.

LES (retired starting from v 6.11.3) - if .true., the model is run as a traditional LES (Large-Eddy Simulation Mode), that is the surface fluxes will be computed assuming the horizontally average values of velocity and scalars, that is the same value of the surface flux is applied everywhere in the domain. Also, unlike CRM, in LES mode, the cloud is diagnosed as cloud condensate higher than 0. Also, the updraft core is diagnosed as being positively buoyant.

CRM (retired starting from v 6.11.3) - if .true., the model is run as a CRM (Cloud-Resolving Model), so that the surface fluxes computed using local values of wind and scalars. Also, unlike LES, in CRM model the cloud is diagnosed as cloud condensate higher than 1% of local supersaturation with respect to water. Also, the updraft core is diagnosed differently as vertical velocity exceeding 1 m/s regardless of buoyancy.

LES_S – if .true. collect cloud and core statistics typical for GCSS PBL cloud cases, that is ql>0 and updraft core is defined not using vertical velocity. If .false.,

use criteria typical for deep convection cases, such as cloud cell when qc+qi>0.01*qsat, and core requires w>1m/s. Default .true.

LAND – if .true., the surface is the land. The soilwetness parameter can also be set (it is swamp by default). Starting from SAM6.11, there is a comprehensive land model, SLM, which is activated by setting SLM=,true. flag (see a corresponding section of this Guide fro more details.) The value of prescribed effective radius over the land is about 8 mkm.

SLM – flag to switch on the Simplified Land Model (SLM). Note that LAND has also to be set to .true.

soil_wetness – parameter that controls the relative 'wetness' of the soil with 0 meaning bare soil (no moisture) and 1 completely saturated soil (swamp). Used only when SLM=.false.

z0 – surface roughness length for the land (m). Used only when SLM=.false.

OCEAN – if .true., the surface is the ocean. The value of prescribed effective radius over the ocean is about 14 mkm.

dowallx – if .true., solid walls will be used as domain boundaries in x direction rather than periodical boundaries.

dowally – if .true., solid walls will be used as domain boundaries in y direction rather than periodical boundaries. Useful when varying Coriolis parameter in meridional direction (in y) is used as periodical conditions in that case don't make sense (see dofplane and dobetaplane).

rundatadir – path to the directory that contains the internal data necessary for running SAM; by default, these datasets are found in RUNDATA directory in SAM's root directory.

dosgs – if .true., do subgrid-scale (SGS) parameterization.

doscalar – if .true., transport a passive scalar in the place of prognostic SGS TKE (works only if **dosmagor**=.true.)

dodamping – if .true., damp gravity waves at the domain top

doupperbound – if .true., maintain the temp and vapor gradient at the domain top

docloud – if .true., allow cloud formation

doprecip – if .true., allow precipitation

longitude0 - longituide (degrees); negative for west and positive for east from the Greenwich meridian

latitude0 - latitude (degrees); positive for North, negative for South.

docoriolis – if .true., the Coriolis force is applied

docorioulisz – if .true, the vertical Coriolis patameter is also applied

dofplane – if .true., the Coriolis parameter is constant everywhere (f-plane approximation); otherwise, it will vary in y direction. The center of the domain is set by latitude0. In the case when dofplane=.false. and docoriolis=.true., the dowally will be automatically set to .true., as periodic boundary conditions in y direction won't make sense. Don't forget to set docoriolis flag to .true. to have the Coriolis force on.

fcor - Coriolis parameter (1/s) in the case when dofplain=.true.. Don't forget to set docoriolis flag to .true. to have the Coriolis force on.

dolongwave – if .true., compute longwave radiation

doshortwave – if .true., compute shortwave radiation

nrad - frequency (in time steps) of updating the radiation heating rates by computing radiative transfer. For example, nrad = 20 means that radiation will be called every 20 model time steps. Computations of radiative heating rates is expensive; therefore, they should not be updated every time step. Generally, for deep convection, 3-5 minutes frequency is adequate.

doperpetual – if .true., sun is perpetual with the total incoming radiation that matches the normal (moving) sun's input for the day defined by day0 parameter.

dosolarconstant used together with the doperpetual flag set to .true..

solar_constant – value of the sun solar input when doperpetual = .true. Generally, it is not equal to the actual solar constant as for perpertual run, there is averaging over the whole day.

zenith_angle – average value of the zenith angle when doperpetual = .true.

doseasons – if .true., the solar incoming radiation change according to current calendar day, that is there is seasonal and diurnal cycle. If .false., then solar diurnal cycle is always the same as during the initial day defined by the day0 parameter

doradforcing – if .true., prescribed radiation from the rad file will be applied

doradhomo – if .true., homogenization the radiative heating rates will be performed that the rates computed for each column separately will then be horizontally averaged and the average profile will be applied in each grid column.

doradlon – if .true., the solar zenith angle depends on the exact longitude of a grid point; otherwise, the solar angle is the same everywhere as if longitude = longitute0 at the center of the domain.

doradlat - if .true., the solar zenith angle depends on the exact latitude of a grid point; otherwise, the solar angle is the same everywhere as if latitude = latitude0 at the center of the domain.

nxco2 – factor of increase of CO2 relative the present as set by the radiation data files; for example, nxco2=2 means double-CO2 run.

dotracers – if .true., tracers will be transported if ntracers > 0 (set in domain.f90)

dosmoke – smoke-cloud case. Optically thick smoke is considered instead of water vapor, so docloud should be set to .false. Initialize smoke concentration in sounding file snd in place of q field. If doradsimple is .true., then rad_simple_smoke() subroutine will be called; it is an error however to call conventional radiation. Note that to work correctly, MICRO_SAM1MOM microphysics should be chosen at compilation. (SAM6.7.4 or later)

dosurface – if .true., compute surface fluxes

dosfchomo – if .true., the computed surface fluxes will be horizontally homogenized (averaged)

dolargescale – if .true., the large-scale forcing from the lsf file will be applied

dosfcforcing – if .true., the surface forcing will be read from sfc file

ocean_type integer parameter that controls the initialization of SST. If =0 (default) SST is constant everywhere; =1 or =2 - sinusoidal distribution in x, and y directions, respectively, with mean tabs_s and amplitude delta_sst as set in the simple_ocean.f90 (set_sst procedure). If you want to set your own SST distribution, or read it from file, you need to create a new entry in set_sst subroutine under ocean_type=4 or larger. Never rewrite the default code as that may lead to some unexpected results, just extend it.

doseawater - if .true., compute surface fluxes taking into account lowering of the saturation vapor pressure over salt water (by a factor 0.981). default=.false.

tabs_s – set SST in K; works when dosfcforcing=.false. and ocean_type is set to 0 (default). If ocean_type=1, or ocean_type=2, then tabs_s is the mean SST. With SLM active, use this parameter to specify water temperature.

delta_sst is the amplitude of sinusoidal SST about tabs_s when ocean type=1 or ocean type=2, so that the maximum SST is at the domain center, minimum value is tabs_s–delta_sst, maximum tabs_s+delta_sst.

dodynamicocean - if .true., the SST will be interactive using a simple mixed-layer model. To take advantage of this option, the dosfcforcing flag should be set to .false.. The ocean model is implemented in simple_ocean.f90 file, so you must edit that file to change the mixed-layer parameters. (See also ocean_type).

timesimpleocean – day to start using slab-ocean model

depth_slab_ocean – depth of the slab ocean in meters; works with dodynamicocean=.true.

Szero - mean ocean transport (away) for the slab ocean model (W/m2)

deltas - amplitude of the linear variation of ocean transport (away) along the x direction for the slab ocean model (W/m2), so the total transport is Szero + deltas*|2x/L – 1|, where L is domain width.

donudging_uv - if .true., nudge horizontal mean u and v to the specified in lsf (if dolargescale=.true.) or snd files

donudging_tq - if .true., nudge both horizontal mean t and q to specified in snd file

donudging_t - if .true., nudge horizontal mean t to specified in snd file

donudging_q - if .true., nudge horizontal mean q to specified in snd file

nudging_uv_z1, nudging_uv_z2 – nudging height boundaries (z2>z1) for uv, meters. Default – nudge at all levels.

nudging_t_z1, nudging_t_z2 – nudging height boundaries (z2>z1) for t, meters. Default – nudge at all levels.

nudging_q_z1, nudging_q_z2 – nudging height boundaries (z2>z1) for q, meters. Default – nudge at all levels.

tauls – nudging time-scale for uv and for tq if tautqls is not set

tautqls – nudging time-scale for tq

doensemble - if .true., run ensemble member (nensemble must then be specified) generated by slight perturbation of the initial sounding using the tqpert file in RUNDATA

nensemble – ensemble member index needed for run initialization

SFC_FLX_FXD – latent heat, sensible heat, and momentum (unless SFC_TAU_FLX=.false.) fluxes are prescribed (.true.) or computed.(.false.) If .true., the prescribed fluxes will be read from sfc file if dosfcforcing=.true., or set by fluxt0 (sensible flux) fluxq0 (latent) in W/m2, and tau0 in m2/s2.

SFC_TAU_FXD – momentum fluxes are prescribed (.true.) or computed.(.false.) If .true., the prescribed flux will be read from sfc file if dosfcforcing=.true., or set by tau0 namelist parameter otherwise in m2/s2.

tau0 – prescribed surface momentum flux (m2/s2) when SFC_TAU_FXD=.true. and dosfcforcing=.false.

fluxt0 – prescribed surface sensible heat flux (W/m2) when SFC_FLX_FXD=.true. and dosfcforcing=.false.

fluxq0 – prescribed surface latent heat flux (W/m2) when SFC_FLX_FXD=.true. and dosfcforcing=.false.

The following parameters control the case setup using CAM's single-column model's SCAM netcdf input file. Note that no snd, lsf, and sfc files are needed then.

doscamiopdata – if .true., then the case setup is done using a SCAM file

dozero_out_day0 – if .true., forces the initial calendar day be 0

iopfile – specify the path and name of the SCAP input file

Initialization of motion

There are two ways to initialize the fluid motion in a new run. One is by specifying some initial random noise in the boundary layer, and the other is to specify a 'warm bubble'. The first way is preferred when using the model to study evolution of turbulent statistics, the second when one wants to study evolution of an explosively developing single cloud (for example, supercell). The following namelist variables control initialization of the fluid motion.

perturb_type – type of perturbation that can be set to integer values (default 0 – initial white noise in temperature field near the surface). You can look at specific initializations for different values of this parameter in setperturb.f90 file. For example, value 2 would create a warm bubble. The bubble is controlled by the following parameters:

bubble_x0, bubble_y0, bubble_z0 – coordinate of the bubble center in meters;

bubble_radius_hor – horizontal radius in meters;

bubble_radius_ver – vertical radius in meters;

bubble_dtemp – bubble temperature perturbation in K with respect to the environment. Temperature perturbation is varying as cosine squared with maximum at the center and zero at the bubble edges.

bubble_dq – bubble water vapor perturbation in kg/kg with respect to the environment. Also changes as a cosine-squared function with zero perturbation at the bubble edges.

Tracers

Starting from version 6.4, one can add arbitrary number of tracers to be transported in the domain. The tracer physics can also be added. First, one needs to specify the number of tracers **ntracers** in domain.f90. A minimal tracer interface can be found in module tracers.f90 . What is guaranteed by the code is that the tracers will be advected and mixed around the domain automatically. The user only needs to supply initialization code, surface fluxes if different than zero, and physics code that describes the change in tracers due to some processes. The code also will output the horizontally averaged statistics with the names of tracers as TR01, TR02, etc. Again, only a generic interface is provided, and it is a user who should insert a specific code except for advection and SGS diffusion which is done automatically. In order to do tracers, don't forget to set **dotracers** = .true. in the namelist-file prm.

Namelist MICRO_M2005

The MICRO_M2005 microphysics package has its own namelist variables. The namelist and default values are described in SRC/MICRO_M2005/microphysics.f90 file.

doicemicro – if .true., use ice species (snow/cloud ice/graupel)

dograupel – if .true., graupel is used for falling ice rather than hail

dohail – if .true., graupel species has qualities of hail

osb_warm_rain – if .true., use Seifert & Beheng (2001) warm rain parameterization in place of Khairoutdinov and Kogan (2000)

dopredictNc – if .true., predict cloud drop number based on CCN concentration

dospecifyaerosol – if .true., specify two modes of (sulfate) aerosol (see aer_rm1, aer_rm2)

dosubgridw – if .true., use estimate of subgrid w in microphysics

doarcticicenucl – if .true., use arctic parameter values for ice nucleation

docloudedgeactivation – if .true., activate droplets at cloud edges as well as base

Nc0 – prescribed droplet number concentration (#/cm3)

ccnconst, **ccnexpnt** – CCN power activation spectrum (N=C S^k) parameters, #/cm3

aer_rm1, **aer_rm2** - two modes of aerosol spectrum used when dospecifyaerosol=T

aer_sig1, **aer_sig2** - geom standard deviation of aerosol size distribution

dofix_pgam – if .true., specify the gamma in gamma distribution for cloud droplets

pgam_fixed – gamma-parameter

dosnow_radiatively_active – self-explanatory

douse_reffc – if .true., compute cloud water effective radius for radiation

douse_reffi – if .true., compute ice-crystal effective size for radiation

Namelist MICRO_THOM

The MICRO_THOM (WRF-Thompson) microphysics package has its own namelist variables. The namelist and default values are described in SRC/MICRO_THOM/microphysics.f90 file.

doicemicro – if .true., use ice species (snow/cloud ice/graupel)

doaerosols – if .true., use Thompson-Eidhammer scheme with water- and ice-friendly aerosols.

doisotopes – option to enable computation of water isotopologues.

Nc0 – prescribed droplet number concentration (#/cm3)

fixed_mu_r, fixed_mu_i, fixed_mu_g – gamma exponents for rain, cloud ice, graupel.

dofix_mu_c, fixed_mu_c - option to specify pgam (exponent of cloud water's gamma distn)

do_output_process_rates - self-explanatory

dosnow_radiatively_active – self-explanatory

dorrtm_cloud_optics_from_effrad_legacyoption – self-explanatory

douse_reffc – if .true., compute cloud water effective radius for radiation

douse_reffi – if .true., compute ice-crystal effective size for radiation

Namelist SGS_TKE

dosmagor – if .true., do the Smagorinsky-type SGS closure rather than prognostic TKE 1.5 order closure. Default is .true.

Namelist MICRO_DRIZZLE

The MICRO_DRIZZLE microphysics used the drizzle water parameterization based on Khairoutdinov and Kogan (2000) microphysics. It is a warm-rain microphysics with no ice; therefore, it is only suitable for the shallow clouds,

preferably stratocumulus-topped boundary layers. There is only one parameter in the namelist MICRO_DRIZZLE:

Nc0 – prescribed droplet number concentration (#/cm3)

Namelist MOVIES

The movies *.raw files store several fields as compressed 1-byte data. For that, the maximum and minimum values should be defined. The default values are defined in movies.f90 file. The limits can be also defined using the namelist MOVIES:

u_min, u_max - surface velocity in x
v_min, v_max - surface velocity in x
cldtop_min, cldtop_max - cloud-top temperature
sst_min, sst_max - surface temperature
tasfc_min, tasfc_max - surface air temperature
qvsfc_min, qvsfc_max - surface vapor mixing ratio
prec_min, prec_max - surface precipitation (will convert to log scale)
cwp_min, cwp_max - cloud water path (will convert to log)
iwp_min, iwp_max - ice water path

Simplified Land Model (SLM)

The theoretical formulation of SLM follows the paper by Lee and Khairoutdinov, 2015: Simplified Land Model (SLM) for use in cloud resolving models: Formulation and evaluation. *J. Adv. Model. Earth Syst*., 07, doi: 10.1002/2014MS000419. SLM files can be found in the directory SRC/SLM.

The model uses one layer of vegetation over interactive soil. It supports 17 classes of land cover or landtypes as defined by the International Geosphere–Biosphere Programme (IGBP) classification. The landtypes are

0 – water
1 - evergreen needleleaf forest
2 - evergreen broadleaf forest
3 - deciduous needleleaf forest
4 - deciduous broadleaf forest
5 - mixed forest
6 - closed shrublands
7 - open shrublands
8 - woody savannas
9 – savannas
10 – grasslands
11 - permanent wetlands
12 – croplands
13 – urban
14 - croplands/natural mozaics
15 - snow/ice
16 - baresoil

Here is the link to details of landtypes: http://www.eomf.ou.edu/static/IGBP.pdf

A landtype index in each surface grid point automatically defines the parameters of vegetation, such as visible and near-infrared albedos, canopy roughness length, root parameters, stomata resistance parameters, basal area index, IR emissivity, displacement height, etc. None of these parameters need to be specified by hand. The only vegetation parameter that needs to be specified in addition to the landtype is the Leaf Area Index (LAI). Note that some landtypes do not require LAI, such as landtypes 0, 11, 13, 15 and 16. Other landtypes do require explicit specification of

LAI to work correctly. The LAI cannot be generally defined by the choice of landtype as it depends on place and season.

In order to activate the SLM, both flags, LAND and SLM, need to be set to true in the namelist PARAMETERS. In the case when SLM flag is not set to true, a simple land model, similar to the one found in previous versions of SAM, will be used.

The land surface can be set-up using two methods. When the whole model domain is covered by land with a single landtype, that landtype can be set by a namelist variable landtype0 in namelist SLM (see below). Also, when the chosen landtype needs a value of LAI (see above), the horizontally uniform LAI can be set by the namelist parameter LAI0.

The second method is a bit more elaborate, but it allows one to setup a horizontally inhomogeneous surface, with different landtypes in any arbitrary configuration, and also islands and lakes. For example, one can set patches of evergreen broadleaf forest surrounded by croplands, lakes, etc, and even glaciers. All it takes is to supply a binary landtype-map file. The file should be prepared using fortran and the following statements:

```
integer landtypemap(nx_gl,ny_gl)
open(1,file=landtype-map-filename, form='unformatted')
write(1) nx_gl
write(1) ny_gl
write(1) landtypemap(1:nx_gl,1:ny_gl)
close (1)
```

where landtypemap is an integer array with values from 0 to 16 corresponding to IGBP landtypes, and nx_gl, ny_gl are the global horizontal grid sizes consistent with sizes set in domain.f90. The corresponding LAI map should be consistent with the landtype map in terms of position in space. It can be specified similarly to the landtype map using the following statements:

```
real(4) LAImap(nx_gl,ny_gl)
open(1,file=LAImap-filename, form='unformatted')
write(1) nx_gl
write(1) ny_gl
write(1) LAImap(1:nx_gl,1:ny_gl)
close (1)
```

Note that the type of LAImap array should be of fortran type real(4), while the landtypemask array should be of fortran type integer. A sample program in directory LANDTYPE demonstrates how to specify a round island with uniform forest.

If one needs an island or islands with one specific landtype and a single LAI, then the namelist parameter **LAI0** can be used rather than the LAI map file.

In order to tell SAM that landtype and LAI map files should be read instead of simple parameters landtype0 and LAI0, the corresponding **readlandtype** and **readLAI** namelist flags should be set to true.

The SLM is driven by the incoming radiation; therefore, the SLM needs the interactive radiation activated by dolongwave and doshortwave set to true in namelist PARAMETERS. Also, no perpetual sun (doperpetual=.true.) is allowed in that case.

The SLM also has an interactive soil model. The soil is composed of nsoil layers. The nsoil is hardcoded in the file slm_vars.f90 (currently 9). However, the vertical soil layer thicknesses are not hardcoded, but need to be specified by the file **soil**, which should be present in any case-directory, which uses SLM. Besides the soil layer thicknesses (in the order from soil surface to bottom), the file **soil** specifies initial profiles of soil temperature, soil wetness (from 0 to 1, with 1 corresponding to the maximum moisture holding capacity, or field capacity of soil), sand and clay content in percent, and also the soil relaxation factor, which controls the nudging of the soil prognostic temperature and wetness to initial profile if one wants to minimize the 'climate drift' from desired state. The clay and send contents are needed to compute the soil hydrological properties. A sample of the soil file can be found in RCELAND directory.

There is a way to overwrite the initial soil temperature and wetness profiles set in file **soil** by using **st0** and **sw0** namelist parameters. The profiles will be uniform with height. Note that the ocean temperature is set separately, as before, that is outside of SLM.

While the soil layer thicknesses are defined only by the file **soil**, and cannot be overwritten by any other method, the other parameters (except for forcing factors) in that file can be overwritten using the variables in namelist SLM. Note though that in that case, the soil parameters will be constant with depth. See the namelist SLM for further details.

In the current version, the SLM does not write any statistics to *.stat files. The only output is written to 2D files in OUT_2D directory. Note that the temperature and wetness of all soil layers are also written to the 2D output file as separate fields for each layer. As the result, the SLM writes quite a few 2D fields to the ouput, so expect the size of 2D output files written by SAM run with SLM to increase significantly compared to a SAM run without SLM.

Namelist SLM

landtype0 – specify a single landtype across all land points. Should be in the range from 0 to 16. To specify different landtypes, use landtype-map file. If set to any nonzero value, will overwrite the landtypes-map values read from a file. Default: 0

LAI0 – specify a unique LAI (m2/m2) across all land points. To specified non-uniform LAIs, use LAI-map file. If set to any nonzero value, will overwrite the landtypes-map values read from a file. (default 0)

clay0 – specify uniform clay content of soil in percent. Will overwrite the content specified by file **soil**.

sand0 – specify uniform sand content of soil in percent. Will overwrite the content specified by file **soil**.

dosoiltnudging – nudge the soil temperature profile to initial profile using the forcing factor profile, defined also in file **soil** (last column). The nudging term at each soil level is multiplied by the forcing factor, so 1 at some soil level means full nudging, and 0 would bean no nudging at all. (default false)

dosoilwnudging – nudge the soil wetness profile to initial profile using the forcing factor profile, defined also in file **soil** (last column). The nudging term at each soil level is multiplied by the forcing factor, so 1 at some soil level means full nudging, 0 would bean no nudging at all, and 0.5 would mean nudging time-scale twice as long as set by **tausoil**. (defalt false)

tausoil – the time-scale (s) of nudging the soil profiles to the initial profiles (defined in **soil** file). (default 86400)

st0 – initial uniform soil temperature (all levels) in K. Will overwrite the initial profiles set by file **soil**.

sw0 – initial uniform soil wetness (all levels), from 0 (bone dry) to 1 (saturated). Will overwrite the initial profiles set by file **soil**.

readlandtype – logical flag to read landtype map from the file defined by namelist variable **landtypefile**.

readLAI – logical flag to read LAI map from the file defined by namelist variable **LAIfile**.

landtypefile – a string containing a path to the file with landtype map read when **readlandtype** is set to true.

LAIfile – a string containing a path to a file with LAI map read when **readLAI** is set to true.