ERDC/CRREL SR-22-3

Cold Regions Research and Engineering Laboratory

US Army Corps of Engineers®
Engineer Research and Development Center

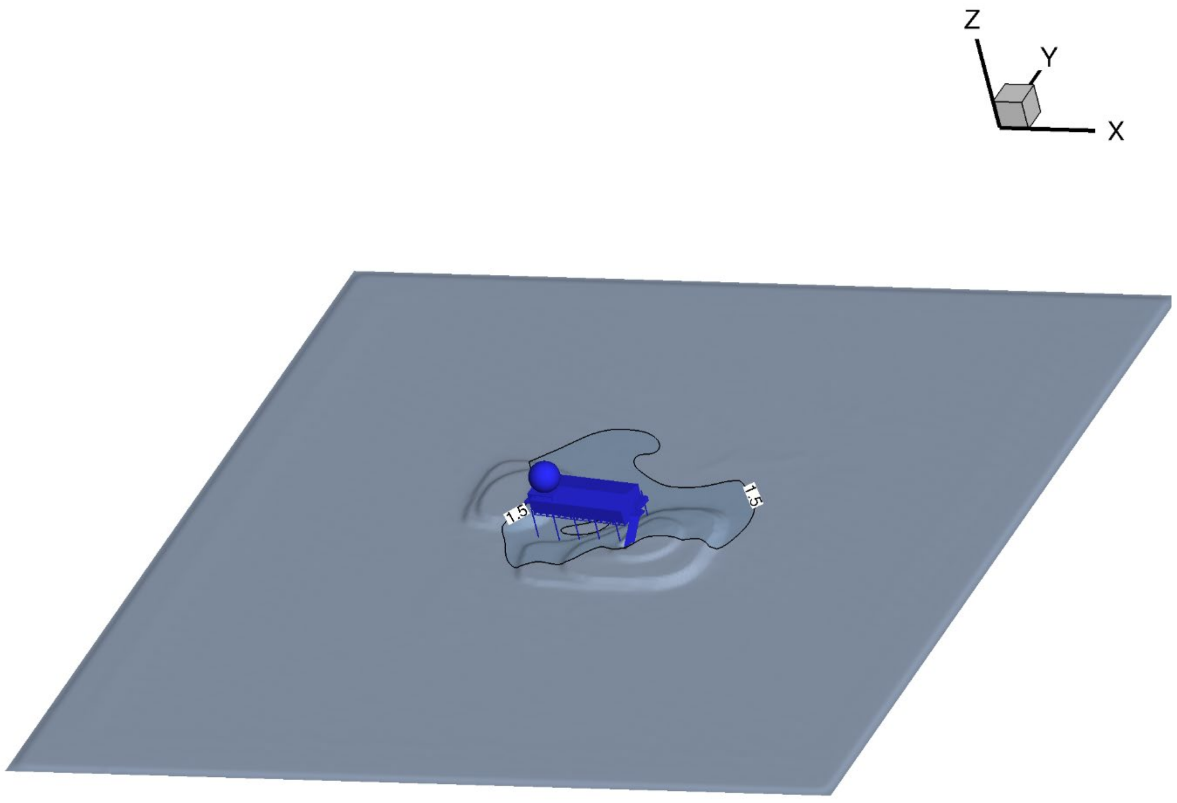
ERDC
ENGINEER RESEARCH & DEVELOPMENT CENTER

Engineering for Polar Operations, Logistics, and Research (EPOLAR)

SAGE-PEDD User Manual

Yonghu Wenren, Luke Allen, and Robert Haehnel

August 2022



Engineering for Polar Operations, Logistics,
and Research (EPOLAR)

ERDC/CRREL SR-22-3
August 2022

SAGE-PEDD User Manual

Yonghu Wenren, Luke Allen, and Robert Haehnel

US Army Engineer Research and Development Center (ERDC)
Cold Regions Research and Engineering Laboratory (CRREL)
72 Lyme Road
Hanover, NH 03755-1290

Final Report

Prepared for National Science Foundation
Office of Polar Programs
Antarctic Infrastructure and Logistics
2415 Eisenhower Avenue
Alexandria, VA 22314

Under Engineering for Polar Operations, Logistics, and Research (EPOLAR)
ANT-21-54, “Impact of Lifting the Elevated South Pole Station on Snow Drift Accumulation on Nearby Facilities”

Abstract

SAGE-PEDD is a computational model for estimating snowdrift shapes around buildings. The main inputs to the model are wind speed, wind direction, building geometry and initial ground or snow-surface topography. Though developed mainly for predicting snowdrift shapes, it has the flexibility to accept other soil types, though this manual addresses snow only. This manual provides detailed information for set up, running, and viewing the output of a SAGE-PEDD simulation.

Contents

Figures and Tables

Preface

This study was conducted for the National Science Foundation (NSF), Office of Polar Programs, Antarctic Infrastructure and Logistics, under Engineering for Polar Operations, Logistics, and Research (EPOLAR) ANT-21-54, “Impact of Lifting the Elevated South Pole Station on Snow Drift Accumulation on Nearby Facilities.” The technical monitor was Michael Gencarelli, NSF.

The work was performed by the Terrestrial and Cryospheric Sciences Branch (Dr. John Weatherly, chief) and the Engineering Resources Branch (Dr. Melisa Nallar, acting chief) of the Research and Engineering Division, US Army Engineer Research and Development Center (ERDC), Cold Regions Research and Engineering Laboratory (CRREL). At the time of publication, Dr. Caitlin A. Callaghan was division chief, and Dr. Rosa Affleck was the EPOLAR program manager. The acting deputy director of ERDC-CRREL was Mr. Bryan E. Baker, and the director was Dr. Joseph L. Corriveau.

COL Christian Patterson was commander of ERDC, and Dr. David W. Pittman was the director.

1 Introduction

This manual explains the use of the SAGE flow solver (Steinhoff and Wenren 2006; Wenren, Steinhoff, and Caradonna 2005) coupled with the particle entrainment, dispersion, and deposition (PEDD) model for windblown snowdrift formation around building structures. The version of SAGE-PEDD described here is V1.0. Refer to the *SAGE-PEDD Theory Manual* for an in depth look at the strategies and mathematics employed by the program and its subroutines (Haehnel, Wenren, and Allen 2022).

While we attempted to make the code as simple as possible, still consider it research software, not a production code. We also assume that the user is familiar with computational fluid dynamics (CFD) concepts and file formats. This is helpful to understand simulation setup, postprocessing, and troubleshooting. Additionally, the code is for a UNIX environment and is driven primarily from the command line; therefore, we recommend familiarity with a terminal interface. Throughout, this manual provides basic commands for the UNIX bash shell. Owing to the computational expense of CFD calculations, the code takes advantage of the MPI (message passage interface) parallel computing libraries available on most computer platforms; use of MPI allows SAGE-PEDD to function on both desktop and supercomputing resources. This guide covers the peculiarities of SAGE-PEDD but is not an exhaustive tutorial on these aforementioned topics (e.g., CFD, UNIX environment, and MPI).

Notes for using this manual:

- All file and directory names in this manual are in bold print (e.g., **sage.in**).
- All variable names or arguments used in input files or on the command line are in a fixed-width font (e.g., `vinf`).
- All other code, including shell commands, is in a fixed-width font and gray background (e.g., `cd $SAGE_ROOT`)

2 Getting Started

2.1 Prerequisites

SAGE-PEDD was written in Fortran 90 and was developed in a UNIX environment (Linux/MacOS) for use on high-performance computing (HPC) systems. It is possible to run the code on a local workstation with a few cores (e.g., 12), though we discourage this for all but the smallest problems. The code has been tested on several DoD Supercomputing Resource Centers (DSRCs) running various implementations of Linux. We highly recommend using SAGE on these, or similar, systems where realistic simulations containing upwards of 12 million grid cells can be run on 200 cores at a time, allowing a moderate-sized simulation to complete in around 48 hr.

If building on an HPC, the required software is likely already installed (possibly in a module). Contact a system admin or help desk to verify these requirements. Otherwise, preinstalled software varies considerably by platform. The SAGE-PEDD build system will find the proper programs in most cases. Nonstandard MPI installations may require providing additional information to the SAGE-PEDD configuration step (details in section 2.3). See below for specific software requirements.

2.1.1 Fortran 90 compiler

A base compiler for Fortran 90 is required (e.g., gfortran, ifort, f90, etc.). Some MPI implementations may come with their own compilers (see below). If the system lacks a Fortran compiler, the GNU project has installation packages for gfortran (opensource Fortran) for all platforms (Windows/Mac/Linux) available at GCC Wiki (2021) https://gcc.gnu.org/wiki/GFortranBinaries.

2.1.2 MPI implementation

SAGE uses MPI libraries for parallelization. There are a number of MPI implementations that include wrapper compilers for Fortran (e.g., mpifort, mpif90, ftn, etc.). These wrappers use a base Fortran compiler, usually the system default, and add MPI functionality by linking the included MPI libraries. OpenMPI and MPICH versions of mpifort are popular choices and are available for all platforms.

In addition to steps taken at compile time, MPI programs must be launched using an appropriate MPI launch command (e.g., mpirun, mpiexec, aprun, etc.). These are generally distributed along with the MPI compiler, requiring no further action to install these MPI utilities.

2.2 Getting the code

Currently, the best way to obtain the SAGE-PEDD source code is by contacting one of the authors directly, listed below. The code is available to stakeholders at The National Science Foundation (NSF). Distribution to parties outside of NSF is subject to evaluation.

- Robert Haehnel—Robert.B.Haehnel@erdc.dren.mil
- Yonghu Wenren—Yonghu.Wenren@erdc.dren.mil
- Luke Allen—Luke.D.Allen@erdc.dren.mil

The code is distributed as an archive file (.tar). On UNIX systems, extract the contents by double-clicking on the file icon and following the prompts. Alternatively, open a terminal window and execute the following commands:

```
cd folder/containing/sage.tgz
tar -zxvf sage.tgz
```

This will create a new folder, **sage_snow**, within the folder where **sage.tgz** is placed, containing all of the documentation and source code for SAGE-PEDD and the associated tools. Documentation (e.g., this user manual) is in the **docs/** folder. The source code is contained in the **src/** folder.

2.3 Installation

Installation of SAGE requires only a few steps. The build system distributed with SAGE-PEDD (autotools) will locate an appropriate compiler and determine some of the appropriate compiler options (or flags). It does not require administrative permission to install in the user's home directory. The instructions given below are for a Linux environment as we expect that, under most circumstances, SAGE-PEDD will be run on an HPC.

2.3.1 Install notes and procedure

To install, enter the UNIX bash shell commands below in the terminal. While in the folder where SAGE-PEDD is going to be built (e.g., **$HOME/Sage_build/**), enter the following commands:

```
export SAGE_ROOT=/path/to/sage_snow   # (1)
cd $SAGE_ROOT
touch src/*                           # (2)
mkdir build
cd build
../src/configure                      # (3)
make
make install                          # (4)
make clean                            # (5)
```

Refer to the notes below for details on certain steps (numbered items below refer to numbers in the code block above):

1. *Optional.* This saves the root directory of the SAGE source code as a variable to use later.
2. The `touch` command is required only for installations of code acquired from GitLab. Errors similar to “aclocal-1.16 not found” are likely if this step is neglected. This is due to GitLab not preserving timestamps, triggering an attempt to rebuild the `configure` file. This step is unnecessary for code obtained via other means (e.g., from a .tar file).
3. The `configure` script performs a series of behind-the-scenes checks and produces the system-dependent **Makefile**. It accepts many arguments to customize the installation. For example, changing the default install location for the `sagempi` executable can be done by appending `--prefix=/my/install/path` to the `configure` command. Section 2.3.2 provides further details for troubleshooting this step.
4. The `make install` command will install the `sagempi` executable to **${HOME}/bin** by default. The bin directory will be created if it does not already exist.
5. *Optional.* This removes intermediate files created by the compiler. This saves disk space and keeps the directory clean.

The above instructions are a suggestion, and advanced users may wish to modify portions of the install process.

2.3.2 Customizing configure

While the default options provided in the `configure` script are sufficient in most cases, there are instances when additional options or settings are desirable. Table 1 provides the default flags while Table 2 provides some of the most common issues. Type `path/to/src/configure –help` or refer to Free Software Foundation (2021) for more information and details on available options.

Table 1. Default Fortran compiler flags.

Compiler-specific flags	gfortran	Other Compilers
accepts -g	"-g -02"	"-g"
does not accept -g	"-02"	""
with –enable-double-precision	"-g -02 -fdefault-real-8"	"-g -r8"

Table 2. Common issues and solutions during compiling.

Issue	Solution
`configure` is not finding the correct Fortran + MPI compiler.	The build system checks a predefined set of locations. If the system has components installed in nonstandard locations, the user will have to tell it where to find that component. The `configure` script checks a number of environment variables for this purpose. For example, to set the Fortran compiler to a specific instance of mpifort, execute the following: `export FC=/custom/path/to/mpifort` Repeat the `configure` step after setting to resume the install.
The user wants to add additional options or flags for the compiler.	For the user to add their own set of compiler flags, use the `FCFLAGS` environment variable prior to the `configure` command. **Note that setting this variable will overwrite the default flags!** Be sure to surround multiple flags with quotes. For example, for the UNIX bash shell: `export FCFLAGAS="-Wall -Wextra"`

2.4 Testing

sage_snow is distributed with a script to run a small test case. The script, `run_test`, is located in **$SAGE_ROOT/sample_inputs/test**. The script will run on four CPUs and will take approximately one minute to complete, depending on processor speed. The test case is intended only to check the installation of the software and does not provide meaningful results. The outcome of the test, passed or failed, is printed to the terminal upon completion. An additional folder called **test-out** will also be created that contains the full results of the test simulation. To run the script, execute the following:

```
cd $SAGE_ROOT/sample_inputs/test
./run_test
```

3 Simulation Setup

3.1 Standard input files

There are a number of input files that SAGE uses for a simulation. These files specify runtime options and provide reference libraries for the SAGE-PEDD model. All files are ASCII based and should be edited using a text editor (a word processor or text-rich editor will place special characters in the file that will make it unusable as an input file for SAGE-PEDD).

3.1.1 sage.in

This is the main input file for SAGE-PEDD that a user will likely need to modify the most. **sage.in** is a Fortran namelist file with a number of options and definitions. The Table 3 describes the definition and functionality of each of the variables. If a variable is not specified in this input file, the default value will be used. Variables that do not have a default value need to be specified.

Table 3. Variable descriptions for the sage.in input file. The file sage.in contains two namelists, controldata and mpidata.

Variable	Units	Default	Type	Description
`Namelist=controldata`				
`vinf`	ft/s*	1.0	Float	Free stream wind velocity. This velocity is assumed to be the velocity at a reference height of 32.8 ft (10 m) above the ground surface. This variable is ignored if meteorological data is used for input conditions.
`alpha, beta`	degrees	0.0, 0.0	Float	Free stream wind direction vertical and horizontal components, respectively. Measured from the *x*-axis. These variables are ignored if meteorological data is used for input conditions.
`nfcstart`		0	Integer	Time step to begin field confinement.
`f_conf_eps`		0.0	Float	Field confinement factor. Applying the default means no field confinement will be applied.
`init`		1	Integer	Indicates whether to start from scratch (`init` = 1) or to read from a restart file (`init` = 2).
`ifstep`		1	Integer	The total number of time steps to run.
`cfl`		0.35	Float	Courant–Friedrichs–Lewy condition. This value, along with the cell size, determines time step according to `dt=cfl*cell_size/vinf`.
`noutput`		10000	Integer	Sets number of time steps for output file frequency. The affected files are **sand.1***, **snowdepth.***, **snow2dout.***, and **snowvel.***. The asterisks represent a number that is incremented, starting with `lunit`.

Table 3 (cont.). Variable descriptions for the sage.in input file. The file sage.in contains two namelists, controldata and mpidata.

Variable	Units	Default	Type	Description
`lunit`		10001	Integer	Initial number to append to output files. Incremented for each output iteration.
`amu`	ft^2/s	0.35	Float	Diffusion coefficient.
`idbody`		0	Integer	Indicates if body file is used. 1 = yes; 0 = no
`ibc1, ibc2, jbc1, jbc2, kbc1, kbc2`		2 for all	Integer	Apply the boundary condition types for the *x*, *y*, and *z* directions, respectively. 0 = no-slip; 1 = inflow; 2 = outflow. The boundary labels can be viewed in Figure 2.
`igread`		0	Integer	Indicates whether to read in a predefined grid file (`igread` = 1) or to use a uniform grid (`igread` = 0). If a grid file is used, the following input variables from **sage.in** are ignored: `xmin`, `xmax`, `ymin`, `ymax`, `zmin`, and `zmax`. Additionally, the variables `nx`, `ny`, and `nz` must match those from the input file or an error will result.
`sandout`		1	Integer	Indicates whether to output PEDD-related files (`sandout` = 1) or not (`sandout` = 0). Affected files are **sand.***, **snowdepth.***, **snow2dout.***, and **snowvel.***. If `sandout` = 0, only air velocity data will be output.
`au`		5.2444e−11	Float	Empirical inlet concentration coefficient. The value of 5.2444e−11 is valid for English units. Setting `au` = 0 has the effect of removing the inlet concentration boundary condition.
`snow_density`	lbm/ft^3	21.85	Float	Initial bulk density of snow on the ground.
`zeta0`	ft	0.0	Float	Initial uniform snow depth.
`isandtest`		0	Integer	Indicates whether to include drag effects for particles. 1 = yes (inertial); 0 = no (passive, flow following).
`idsnowage`		0	Integer	Indicates whether to include snow aging effects (hardening/sintering). 1 = yes; 0 = no.
`age_scale`		None	Float	Factor for scaling the snow aging (sintering) rate. `age_scale` = 1 means there will be no scaling of the snow aging process.
`airtemp`	°F	30.0	Float	Air temperature. Not used if `idinflow` = 1.
`idinflow`		0	Integer	Indicates use of meteorological data for inflow conditions. 1 = yes; 0 = no (constant wind, temp). Note: The use of meteorological data is currently in beta.
`snow_scale`		500.0	Float	Factor for scaling the snow deposition and erosion rate. `snow_scale` = 1 indicates that the rate of drift build up is controlled by the computational time step (i.e., the fluid flow and the snowdrift deposition rate are tightly coupled).
`Namelist=mpidata`				
`nx, ny, nz`		None	Integer	Number of grid cells in *x*, *y*, and *z* directions. If `igread` = 1, these quantities must match those in **grid3d4mud.in**. $nx = ixp * 2^{(iex-1)}, ny = jyq * 2^{(jey-1)}$, $nz = kzr * 2^{(kez-1)}$

Table 3 (cont.). Variable descriptions for the sage.in input file. The file sage.in contains two namelists, controldata and mpidata.

Variable	Units	Default	Type	Description
`ixp, jyq, kzr`		None	Integer	Base grid dimensions for the flow solver multigrid method.
`iex, jey, kez`		None	Integer	Level of the multigrid. $ixp \geq 2(nxprocs), jyq \geq 2(nyprocs), kzr \geq 2(nzprocs)$
`nxprocs, nyprocs, nzprocs`		None	Integer	Number of domain partitions in *x*, *y*, and *z* directions. The product must equal the total number of processors being used.

* For a full list of the spelled-out forms of the units of measure used in this document and their conversions, please refer to *US Government Publishing Office Style Manual*, 31st ed. (Washington, DC: US Government Publishing Office, 2016), 245–252, https://www.govinfo.gov/content/pkg/GPO-STYLEMANUAL-2016/pdf/GPO-STYLEMANUAL-2016.pdf.

3.1.2 sand.in

This is the input file for particle and ground cover definitions. Table 4 describes the variables for the file. Note that there are some variables in the file that are not described here. These variables are used only for Lagrangian simulations (i.e., individual particle tracking) and are not supported in SAGE-PEDD at this time. They remain in place for compatibility with previous versions of SAGE that supported particle tracking.

Table 4. Variable descriptions for the sand.in input file.

Variable	Units	Default	Type	Description
`soiltype`		None	Integer	Designations for soil types are included in **soils.txt.** Alternatively, use −1 to define a custom soil type using **soil_user.txt**.
`Veg Legend`		**NLDC2011.txt**	String	Specifies reference file containing vegetation codes and descriptions.
`Veg code`		None	Integer	Defines vegetation type. Use 12 for snow.
`Coverage`	%	None	Float	Indicates percentage of land covered by `Veg code`
`zo`	ft	None	Float	Aerodynamic roughness height. Use `zo` = 0 to let surface roughness (from `soiltype`) and vegetation dominate.
`pa`	lbm/ft^3	None	Float	Fluid density for buoyancy effects.
`nu`	ft^2/s	None	Float	Fluid kinematic viscosity.
`mois`		None	Float	Fraction of moisture content in soil. Not used for snow.
`shear_ht`		None	Float	Currently unused.
`Trans model`		1	Integer	Transmission model type employed. 1 = MODTRAN. Currently not used for snow.
`Wavelength`	μm	0.55	Float	Wavelength for transmission calculation. Currently not used for snow.

3.1.3 bodygeo.in

Use this file to tell SAGE which geometry files to use for the simulation. The version adapted for this code is simple. The first two lines are comments and are ignored by SAGE-PEDD. The number of bodies (e.g., buildings) to import is input on line 3. The fourth line is also a comment. Then the file names for the respective geometry files are provided, starting on line 5. Each body requires two files, as described in section 3.1.5. The two file names needed for each body are given on the same line, separated by a comma. Figure 1 shows an example input file for a simulation with two buildings.

Figure 1. Example bodygeo.in input file.

```
1    body geo for sage-rotor -- bodygeo.in
2  number_of_object (integer)
3  2
4  tri_filename, f_filename
5  'building1_tri.in', 'building1_F.in'
6  ‘building2_tri.in', 'building2_F.in'
```

3.1.4 sagegrid3d.in

If present, this file contains the computational grid for SAGE. This file is binary and cannot be edited directly. It is produced by the grid creation tool **grid3d4mud** described in section 3.2.2. **grid3d4mud.in** is not used if `igread` equals 0.

3.1.5 Geometry files

SAGE requires two input files for each body geometry. These are **[body_name]_F.in** and **[body_name]_tri.in**. The first contains the distance function, F, used to determine the location of each grid node relative to the body surface. Locations where F is greater than zero are outside of the body, while locations where F is zero or less indicate that the node is inside of the body. The flow velocity is set to zero for all points inside of the body. The second file contains the locations of the triangular facets that make up the body surface.

The formats of the geometry input files are unique to SAGE. However, they can be produced automatically from a file in stereolithography (.stl) format using the included Fortran utility: **stl2f** (section 3.2.1). This tool will need to be compiled on the system it will be used on before it can be used. See the instructions in section 3.2 below for details and instructions.

3.1.6 met_data_for_sage_snow.in

This input file contains recorded meteorological data for a specified time span in a particular season. The file provides time-dependent wind direction, wind speed, and air temperature for SAGE-PEDD. SAGE-PEDD does not read meteorological data directly from a NOAA data file. The input file is generated by the **met2sage_in.py** Python utility (section 3.2.3). Note that the use of meteorological data is currently in beta.

3.2 Utilities

This section highlights the use of several tools included with the distribution. Their source code can be found in the **utilities** folder.

3.2.1 stl2f: creating geometry files

This tool converts geometry files from .stl to a format usable by SAGE and requires two files to use. The first is the source code, **stl2f.f90**, which is written in Fortran and must be compiled before use. The second is the input file, **stl2f.in**, which is used at execution time to tell the program about the user's geometry. Below is a list of some sample steps to take.

Note: Align positive y-axis in the .stl file with north (true or grid) if using meteorological data (this feature is in beta).

1. Compile **stl2f**.

   The source code for **stl2f** is a single file, so a make file is unnecessary. Instead, the tool should be built with a traditional Fortran 90 compiler (no MPI needed). It may be desirable to build the code in a separate directory to avoid cluttering the source folders, though this is optional. It is often convenient to add the folder containing the binary to the system search path (i.e., `$PATH`) or place the binary in a location that is already in `$PATH`.

   The example code snippet below can be used to compile the **stl2f** tool and place it in the directory **$HOME/bin**. Note that this is the default install location for the `sagempi` program executable, and is recommended for consistency. Users can substitute a compiler of choice if necessary.

```
cd $SAGE_ROOT/utilities/
gfortran -o stl2f stl2f.f90
mv stl2f $HOME/bin
```

- To check if **$HOME/bin** is in `$PATH`, use the following command:

  ```
  echo $PATH
  ```

- Search the output for the folder name (entries are separated by ":"). If **$HOME/bin** is not in `$PATH`, add it to the environment using the command below. Make this change permanent by adding this command to **~/.bashrc** (or equivalent).

  ```
  export PATH=$HOME/bin:$PATH
  ```

2. Configure the input file.

   The first step is to place a copy of the input file, **stl2f.in**, in the same folder as the .stl body file. The input file contains basic data about the desired file properties. Do not change the name of this file. Open the file in a text editor and edit the following options:

   - Scale (line 3)—SAGE expects geometry files in feet. Use the scale factor to convert from a .stl file with different units to feet (e.g., use 3.28 if the model is in meters). If the model is already in feet, use 1.0.
   - STL file name (line 5)—The name of the .stl file (including the .stl extension). This file should be located in the same folder.
   - Geometry file name (line 7)—The name of the output file to contain body coordinates. The convention is to append "_tri.in" to the name of the body.
   - F function file name (line 9)—The name of the output file to contain the distance function. The convention is to append "_F.in" to the name of the body.

3. Execute file conversion.

   The example below provides two ways to run the **stl2f**: (A) if the executable is in `$PATH`, and (B) if the executable is not in `$PATH`.

A
```
cd path/to/geom_folder
stl2f
```

B
```
cd path/to/geom_folder
path/to/stl2f
```

4. Repeat.

   If multiple body files are required, repeat steps 2 and 3 for each .stl file.

3.2.2 grid3d4mud: generating a computational grid

One of the many challenges associated with CFD simulations is in developing an appropriate computational grid. SAGE eases this process somewhat by using a uniform rectangular grid and vorticity confinement techniques. However, if a nonuniform grid is required, there are still a number of parameters that must be defined to generate a suitable mesh. This tool produces a uniform grid with stretched boundary regions compatible with the MUDPACK partial differential equations solver that was once used by SAGE, though MUDPACK is not used anymore, the utility creates these stretched computational grids when needed. The program is written in Fortran and must be compiled before use. The source code consists of a single file, **grid3d4mud.f90**. It reads an input file, **grid3d4mud.in**, and outputs the grid file sagegrid3d.in. Steps for using the tool are below.

1. Compile **grid3d4mud.f90**.

   Compiling **grid3d4mud** is done in the same way as **stl2f**. To compile the code and install it to **$HOME/bin**, type the commands below in the terminal.

```
cd $SAGE_ROOT/utilities/
gfortran -o grid3d4mud grid3d4mud.f90
mv grid3d4mud $HOME/bin
```

2. Configure the input file.

   The **grid3d4mud.in** input file is a Fortran namelist file containing options to define and set up the main, uniform grid, as well as a "stretched" region at the domain boundaries. The stretched region may be different for each boundary. The boundary labels (`i1`, `i2`, `j1`, `j2`, `k1`, `k2`) are defined according to the diagram in Figure 2. Figure 3 shows an example of a simple, two-dimensional grid with stretching. Table 5 describes the variables. The total number of grid cells in a given direction is the sum of the number of fine cells, the number of stretch cells on each side, and the number of cells beyond the stretch region on each side. Using the *x* direction as an example:

```
nx=nxfine+i1cells+i2cells+i1out+i2out.
```

Note: Align the positive *y*-axis with north (true or grid) if using meteorological data. (This feature is in beta.)

3. Create the grid.

   Generate **sagegrid3d.in** by executing the **grid3d4mud** utility in the directory containing **grid3d4mud.in**. For example, if the input file is located in a folder called **rundir**, and the **grid3d4mud** executable is located in `$PATH`, generate the grid simply by the following:

   ```
   cd path/to/rundir
   grid3d4mud
   ```

Figure 2. Domain boundary labels. This scheme is consistent throughout the code.

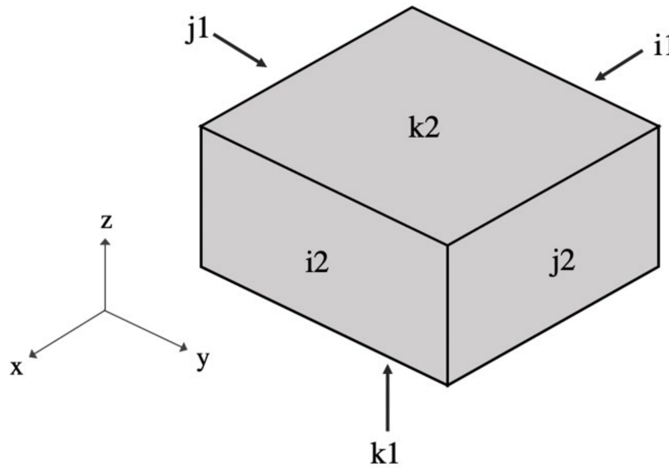


Figure 3. Simple, two-dimensional grid in *x* and *y* directions, showing the effects of the various grid parameters. The grid definition in the *z* direction can be defined by extending this logic to nzfine, k1cells, k2cells, k1out, and k2out. The diagram shows (1) the uniform, fine grid; (2) the stretch regions; and (3) the outer, course grid. The grid does not need to be symmetrical, as the one shown here.

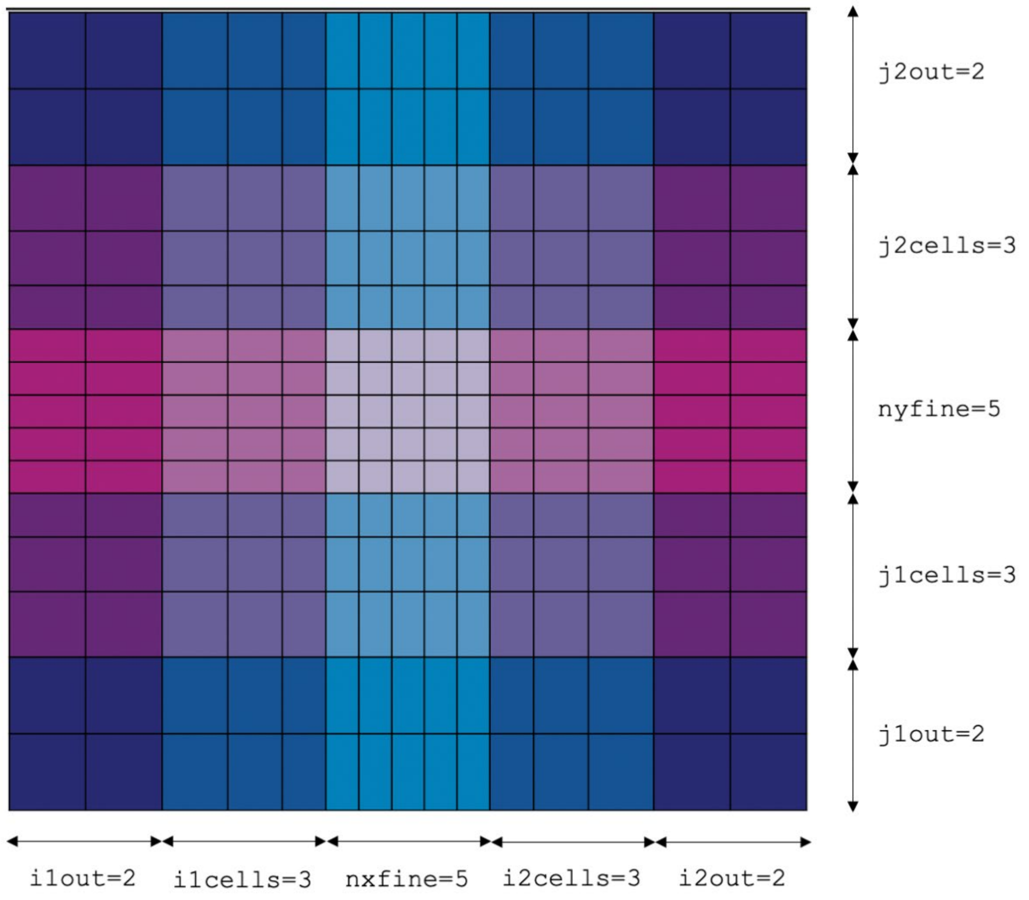


Table 5. Variable descriptions for the grid3d4mud.in input file.

Variable	Description
`xfinemin`, `xfinemax`, `yfinemin`, `yfinemax`, `zfinemin`, `zfinemax`	Extents of the fine grid domain.
`nxfine`, `nyfine`, `nzfine`	Number of uniform grid cells in the *x*, *y*, and *z* directions, respectively. These quantities should match `nx`, `ny`, and `nz` from **sage.in**. See Figure 3 for clarification.
`i1cells`, `i2cells`, `j1cells`, `j2cells`, `k1cells`, `k2cells`	Number of cells to include in the stretch region for each boundary. The maximum recommended value for any of these variables is 16. See Figure 3 for clarification.
`rx1`, `rx2`, `ry1`, `ry2`, `rz1`, `rz2`	Stretch ratio for each boundary. The maximum recommended value for any of these variables is 0.13 (represents 13% growth between adjacent cells).
`i1out`, `i2out`, `j1out`, `j2out`, `k1out`, `k2out`	Number of grid cells outside of the stretched region. See Figure 3 for clarification.

3.2.3 met2sage_in.py: generating weather data for SAGE

This utility converts meteorological data from the standard NOAA format into a space-delimited input file for SAGE. This utility is a Python script, compatible with versions 2.7 and 3. Inputs to the script are the name of the meteorological data file, as well as the start and end dates for the simulated time period. Inputs are provided by editing the script in a text editor and executing the Python interpreter:

```
python met2sage_in.py
```

3.2.4 snowdepth2tecplot: file conversion for animations

This program is an MPI-enabled Fortran utility that converts the Plot3D formatted **snowdepth.*** files to the Tecplot formatted data files. The files it generates (with .dat extensions) can be viewed by Tecplot or ParaView. After a series of files are converted and loaded into ParaView, they can be used to create animations of the evolving snow surface. The following are instructions for compiling and using the code.

1. Compile **snowdepth2tecplot**.

   Similar to the other Fortran utilities, the source code for **snowdepth2tecplot** is a single file and can be easily compiled directly. Unlike the other utilities, this program uses MPI libraries and must be compiled using an MPI wrapper compiler for Fortran (e.g., mpifort, mpif90, ftn, etc.). To compile the code and install it to **$HOME/bin**, type the commands below in the terminal:

   ```
   cd $SAGE_ROOT/utilities/
   mpifort -o snowdepth2tecplot mpi_snowdepth2tecplot.f90
   ```

   By default, the program finds and converts all files with names **snowdepth.10001** though **snowdepth.12000**. This behavior is based on the default value of `lunit` in the **sage.in** input file. If `lunit` is changed, modify the values of `lunitst` and `lunited`, located on lines 115 and 116 of the **mpi_snowdepth2tecplot.f90** source file, to agree with the value of `lunit`. Recompile the utility for this change to take effect.

2. Generate Tecplot files.

   To generate Tecplot files from the completed **snowdepth.*** files, move to the directory containing those output files, and execute the program using an appropriate MPI launch command. The example command below assumes that the executable was placed in **$HOME/bin** and that this directory is in the user's system search `$PATH`.

   ```
   cd path/to/outputdir
   mpiexec -n 4 snowdepth2tecplot
   ```

   Since this will take some time to execute, do not run it from the command line on an HPC system, but run using the batch queuing system available on the HPC (e.g., portable batch system [PBS] scripting as is available on the DoD DSRCs) so that it will run on the compute notes. Discussion of batch queuing systems is beyond the scope of this work, though section 4.2 below provides limited information on PBS queuing. The user should consult documentation on the queuing system available on the HPC system they are using to use that capability.

3.3 Other files

3.3.1 soils.txt

The **soils.txt** file contains a library of soil classifications as defined by Fast All-Season Soil Strength (FASST; Frankenstein 2008; Frankenstein and Koenig 2004), Unified Soil Classification System (USCS), and the USDA classification systems. In this context, soil refers to any loose ground type (e.g., sand or snow). This file is for reference and should not be modified. Use the `soiltype` variable in **sand.in** to select the soil class. Class 30 defines a typical snowpack with three particle sizes. If a custom soil class is desired, use the **soil_user.txt** file.

Starting at line 16, each column defines a soil property that can be determined from the legend at the top of the file. Quantities include the identifiers for the three different classes; the number of different particle sizes in the soil (`nbins`); the percentages of sand, silt, and clay; the surface area per unit mass; the aerodynamic roughness height; the fraction of the soil made up by each particle size; the size of the particles; and the density of the particles.

3.3.2 soil_user.txt

Use the **soil_user.txt** file to define a custom soil composition. This file is used if `soiltype` is less than zero in **sand.in**. The user may specify the number of different particle sizes, `nbin`; the surface area per unit mass, `As`; the percent clay (0.0 for snow); the aerodynamic roughness height, `zo_bare`; the fraction of the soil composition for each size class, `F`; the diameter of each size class, `ds`; and the particle density for each size class, `ps`. Expected units are provided in the file.

3.3.3 NLDC2011.txt

The **NLDC2011.txt** file contains a library of vegetation and ground cover types. This file is read based on the value of `Veg code` given in **sand.in**.

4 Launching a Simulation

The simulation is ready to be launched once all input files are in place. There are two optional utilities included to aid in launching the `sagempi` program. These scripts are helpful starting places and may not support all desired run configurations. The choice of script depends on the host machine. For systems with no queuing system (e.g., a typical desktop computer), one should use the **run_sage** utility. For systems using the PBS queuing system, use **run_sage.pbs**.

4.1 run_sage

The **run_sage** file is an executable shell script (bash) for running `sagempi`. The script accepts several command line arguments to ease simulation input and output setup. This script can be used to run SAGE directly on systems that do not have queuing systems. Table 6 lists the supported options. The short and long names (given in parentheses) are equivalent. Though each item is optional, it is necessary to provide the specified information if used.

Table 6. Descriptions of the optional flags for the run_sage execution utility.

Command Line Option	Details
`-i dir, --input-dir dir`	Specifies the directory containing the SAGE input files. Attempts to use current directory if not specified.
`-o dir, --output-dir dir`	Specifies the location for SAGE output. Input files are also copied here. Uses current directory if not specified. Using this option prompts the creation of a time-stamped folder in the target location (e.g., **sage_snow_2020-01-01-1430**).
`-c mpi_executable, --use-command mpi_executable`	Overrides the default MPI launch command (mpiexec).
`-n, --nprocs N`	Sets the number of processors to use. Default is 1.
`--mpi-opts "option ..."`	Specifies additional options to pass to MPI. Must wrap arguments in quotes.
`-h, --help`	Displays help and usage screen and exit.

Example usage:

- Run in the current directory using 48 cores.

```
./run_sage -n 48
```

- Run using 48 cores, reading from the current directory and writing output to **/workdir/Sage**. This creates the directory **/workdir/Sage/sage_snow**.

```
./run_sage -n 48 -o /workdir/Sage
```

- Run using 96 cores, reading from **/home/Sage/inputdir** and writing to **/workdir/Sage**. Also enable the use of hardware threads.

```
./run_sage -n 96 -i /home/Sage/inputdir -o /workdir/Sage --
mpi-opts "--use-hwthread-cpus"
```

4.2 run_sage.pbs

The file **run_sage.pbs** is a job script for running SAGE-PEDD on hosts with the PBS queuing system. Changes are made by opening the file and editing the options and PBS directives. Note that the PBS directives begin with `#`, which ordinarily denotes a comment. However, the PBS queuing system interprets `#PBS` as a directive. To comment out a PBS directive, use two `#` (e.g., `##PBS`). To submit the job, make all appropriate changes described in Tables 7 and 8, and execute `qsub run_sage.pbs` in the terminal.

Table 7. Descriptions of PBS directives.

PBS Directive	Details
`#PBS -N job-name`	The name of the job.
`#PBS -o job.out`	The file for job stdout.
`#PBS -e job.err`	The file for job stderr.
`#PBS -j oe`	(Optional) Merges stderr into stdout.
`#PBS -A ABCD1234`	The subproject ID. There must be an allocation on a subproject to run this script.
`#PBS -q standard`	The queue to submit the job (e.g., debug, standard).
`#PBS -l walltime=hh:mm:ss`	The maximum amount of time to allocate to the job.
`#PBS -l select=N:mpiprocs=n:ncpus=n`	The number of nodes (N) and processors per node (n) to allocate to the job. The total number of cores used will be N*n.
`#PBS -l application=sage`	The name of the application (for internal use).
`#PBS -M me@mail.com`	(Optional) An email to receive notifications from the job.
`#PBS -m be`	(Optional) When to send notification emails (b = beginning, e = end, be = both).

Table 8. Descriptions of the runtime quick options provided for the run_sage.pbs utility.

Runtime Options	Details
`EXEC`	Path to the `sagempi` executable.
`INPUT_DIR`	Location of the SAGE input files directory. The location of `INPUT_DIR` may be anywhere the user has read permissions on the system. The script will automatically copy files to a folder in the user's `$WORKDIR` if `INPUT_DIR` is not already part of that directory. This may cause problems if the `$WORKDIR` environment variable is not set.
`N_PROCS`	Number of processors to use. Should be equal to (number of nodes) × (cores per node).

4.3 Restarting a simulation

To restart a simulation from one that has completed, change the name of the file **q.save** to **restart.in**, and then edit **sage.in** and set `init` to 2. To have the new output file names incremented correctly, update the `luinit` variable as well (e.g., if one of the last output files created is **snow-depth.10399**, then `lunit = 10400` will start the next set of output files with 10400 and increment from there). Then, restart the simulation.

5 Postprocessing

5.1 Output files

SAGE produces a number of output files. Each one is listed below with a brief summary of its purpose and its contents.

5.1.1 sage.out + sagempi.out

These files summarize the input variables and initial conditions for the simulation.

5.1.2 tm3.out

This file provides time history information for select variables at each time step.

5.1.3 out_***

These files are the standard output for each MPI rank.

5.1.4 q.save

q.save contains the saved state of all quantities required for restart (three-dimensional, flow field velocity, airborne snow concentration, snow depth, etc.). Convert to a restart file by changing the name of the file to **restart.in**.

5.1.5 sage.g + sage.q

These files are the main output files for the three-dimensional flow field. They are output in NASA's Plot3D format, with modifications to the variable assignments. **sage.g** is the Plot3D grid file. It contains the *x*, *y*, and *z* locations of all points on the computational grid. **sage.q** is a Plot3D solution file. It contains the variables `RHO`, `RHO-U`, `RHO-V`, `RHO-W`, and `E`. Table 9 provides standard and modified definitions for these variables.

Table 9. Plot3D solution file format with adaptations for SAGE-PEDD.

Variable	Standard Definition	Modified SAGE-PEDD Definition
`RHO`	Fluid density	Body distance function in units of feet. `RHO` = 0 defines building surfaces.
`RHO-U, RHO-V, RHO-W`	Fluid momentum in *x*, *y*, and *z* directions	Fluid velocity in *x*, *y*, and *z* directions. Output in ft/s.
`E`	Fluid energy	Vorticity magnitude.

5.1.6 sand.*

This is a Plot3D function file and should be loaded along with the **sage.g** grid file for viewing. The file contains a single scalar field representing the snow concentration at each node for a particular time step. Note that concentration is assigned the variable name `F1V1` when loaded in Tecplot. The output frequency is set in **sage.in** using the variable `noutput`.

5.1.7 snowdepth.*

This is a Plot3D grid file representing the *x*, *y*, and *z* locations of the snow surface. The output frequency is set in **sage.in** using the variable `noutput`.

5.1.8 snow2dout.*

This file is a Plot3D function file containing several scalar field variables associated with the snow surface. These variables are (1) the top of the ice layer (*z* direction), (2) the threshold friction velocity, (3) the bulk density of the snow, (4) the change in snow depth in the previous time step, and (5) the surface shear stress. The output frequency is set in **sage.in** using the variable `noutput`.

5.1.9 snowvel.*

These are Plot3D function file containing the velocity components of the suspended or saltating snow. If `isandtest` is zero, this will be the same as the air velocity. Otherwise, inertial and drag forces on the particles are considered. Output frequency is set in **sage.in** using the variable `noutput`.

5.2 Data visualization

Geometry files are in STL format. Field outputs are provided in Plot3D format. The results may be viewed by any visualization software with STL and Plot3D readers. ParaView is an open-source program that can access data remotely (no need to download data to a local workstation) and can be used to create animations of the snowdrift history. However, the Plot3D format output from SAGE-PEDD needs to be converted to a Tecplot (.dat) format file to be read by ParaView. We primarily use Tecplot for results visualization.

Plot3D readers require users to load multiple files to view results: a grid file (often with a .g extension, though this is not always required) and at least one solution or function file. Refer to the list below for files pertaining to each of the two main sets of Plot3D outputs.

- Viewing the flow field
  - Grid file: **sage.g**
  - Solution file: **sage.q**
  - Function files: **sand.***, **snowvel.***
- Viewing the snow surface
  - Grid File: **snowdepth.***
  - Solution file: none
  - Function file: **snow2dout.***

The building-geometry files (STL) used for the simulations can also be overlayed in the view of these Plot3D solution files. The specifics of how to import and view all of these files into a particular postprocessor or viewer such as ParaView, Tecplot, Fieldview, etc. varies and is beyond the scope of this manual. Refer to the user manuals for the specific post processor used.

References

Frankenstein, S. 2008. *FASST Soil Moisture, Soil Temperature: Original vs. New*. ERDC/CRREL TR-08-7. Hanover, NH: US Army Engineer Research and Development Center, Cold Regions Research and Engineering Laboratory.

Frankenstein, S., and G. G. Koenig. 2004. *FASST Vegetation Models*. ERDC/CRREL TR-04-25. Hanover, NH: US Army Engineer Research and Development Center, Cold Regions Research and Engineering Laboratory.

Free Software Foundation. 2021. "How Configuration Should Work." *GNU Coding Standards*. https://www.gnu.org/prep/standards/html_node/Configuration.html.

GCC Wiki. 2021. "GFortranBinaries." GCC Wiki. Last modified 3 November 2021. https://gcc.gnu.org/wiki/GFortranBinaries.

Haehnel, Robert B., Yonghu Wenren, and Luke D. Allen. 2022. *SAGE-PEDD Theory Manual: Modeling Windblown Snow Deposition around Buildings*. ERDC/CRREL TR-22-8. Hanover, NH: US Army Engineer Research and Development Center, Cold Regions Research and Engineering Laboratory. http://dx.doi.org/10.21079/11681/44942.

Steinhoff, J., and Y. Wenren. 2006. "An Efficient Vorticity Confinement Based Lifting Surface Method for Rotor Wake Computations." In *Proceedings of the 32nd European Rotorcraft Forum*, Maastricht, Netherlands.

Wenren, Y., J. Steinhoff, and F. Caradonna. 2005. "Application of Vorticity Confinement to Rotorcraft Flows." In *Proceedings of the 31st European Rotorcraft Forum*, Florence, Italy.

Acronyms and Abbreviations

CFD	Computational Fluid Dynamics
DSRC	DoD Super Computing Resource Center
FASST	Fast All-Season Soil Strength
HPC	High-Performance Computing
MPI	Message Passage Interface
NSF	National Science Foundation
PBS	Portable Batch System
PEDD	Particle Entrainment, Dispersion, and Deposition
USCS	Unified Soil Classification System

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE (DD-MM-YYYY) August 2022	**2. REPORT TYPE** Technical Report / Final	**3. DATES COVERED (From - To)** FY19–FY21
4. TITLE AND SUBTITLE SAGE-PEDD User Manual		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT
6. AUTHOR(S) Yonghu Wenren, Luke Allen, and Robert Haehnel		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army Engineer Research and Development Center (ERDC) Cold Regions Research and Engineering Laboratory (CRREL) 72 Lyme Road Hanover, NH 03755-1290		**8. PERFORMING ORGANIZATION REPORT NUMBER** ERDC/CRREL SR-22-3
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Science Foundation Office of Polar Programs Antarctic Infrastructure and Logistics 2415 Eisenhower Avenue Alexandria, VA 22314		**10. SPONSOR/MONITOR'S ACRONYM(S)** NSF
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
Funded under Engineering for Polar Operations, Logistics, and Research (EPOLAR), ANT-21-54, "Impact of Lifting the Elevated South Pole Station on Snow Drift Accumulation on Nearby Facilities"

14. ABSTRACT
SAGE-PEDD is a computational model for estimating snowdrift shapes around buildings. The main inputs to the model are wind speed, wind direction, building geometry and initial ground or snow-surface topography. Though developed mainly for predicting snowdrift shapes, it has the flexibility to accept other soil types, though this manual addresses snow only. This manual provides detailed information for set up, running, and viewing the output of a SAGE-PEDD simulation.

15. SUBJECT TERMS
Amundsen Scott South Pole Station (Antarctica)--Buildings, Engineering--Cold weather conditions, Computational fluid dynamics, EPOLAR, NSF, Snow--Computer simulation, Snow--Antarctica, Snow mechanics, Wind forecasting

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	34	**19b. TELEPHONE NUMBER (include area code)**