IRENE: AE9/AP9/SPM Radiation Environment Model

Build Instructions

Version 1.50.001

The IRENE (International Radiation Environment Near Earth): (AE9/AP9/SPM) model was developed by the Air Force Research Laboratory in partnership with MIT Lincoln Laboratory, Aerospace Corporation, Atmospheric and Environmental Research, Incorporated, Los Alamos National Laboratory and Boston College Institute for Scientific Research.

IRENE (AE9/AP9/SPM) development team: Wm. Robert Johnston[1] (PI), T. Paul O'Brien[2] (PI), Gregory Ginet[3] (PI), Stuart Huston[4], Tim Guild[2], Christopher Roth[4], Yi-Jiun Su[1], Rick Quinn[4], Michael Starks[1], Paul Whelan[4], Reiner Friedel[5], Chad Lindstrom[1], Steve Morley[5], and Dan Madden[6].

To contact the IRENE (AE9/AP9/SPM) development team, email ae9ap9@vdl.afrl.af.mil .

The IRENE (AE9/AP9/SPM) model and related information can be obtained from AFRL's Virtual Distributed Laboratory (VDL) website: https://www.vdl.afrl.af.mil/programs/ae9ap9

V1.00.002 release: 05 September 2012

V1.03.001 release: 26 September 2012

V1.04.001 release: 20 March 2013

V1.04.002 release: 20 June 2013

V1.05.001 release: 06 September 2013

V1.20.001 release: 31 July 2014

V1.20.002 release: 13 March 2015

V1.20.003 release: 15 April 2015

V1.20.004 release: 28 September 2015

V1.30.001 release: 25 January 2016

V1.35.001 release: 03 January 2017

V1.50.001 release: 01 December 2017

[1] Air Force Research Laboratory, Space Vehicles Directorate
[2] Aerospace Corporation
[3] MIT Lincoln Laboratory
[4] Atmospheric and Environmental Research, Incorporated
[5] Los Alamos National Laboratory
[6] Boston College Institute for Scientific Research

Build Instructions for IRENE (AE9/AP9/SPM) Radiation Environment Model Software, Version 1.50.001

The base distribution of the IRENE model package ships with pre-compiled single- and multi-threaded application binaries for the Windows platform only. To build the model and tools on other platforms, *the source code distribution must be obtained from the model development team*; send requests for the source code to: ae9ap9@vdl.afrl.af.mil .

IRENE (AE9/AP9/SPM) uses a CMake-based build process. CMake is a cross-platform make tool that supports most major operating systems and compilers. It should be possible to build and run the software in any CMake-supported environment that also supports all required third-party library dependencies.

Version 1.50.001 of the AE9/AP9/SPM software has been tested under Windows7 (32-bit and 64-bit) and under CentOS 6.x version of the Linux operating system (64-bit). The software has been reported to be successfully built and run on other operating systems and versions of these operating systems, but formally verified on only those listed here.

Third-Party Dependencies

The list of third-party libraries shown below specifies the respective versions that were used when building and testing this IRENE (AE9/AP9/SPM) software release. More recent versions of these libraries may be available and will possibly work (but QT **4**.x and Qwt **5**.x are specifically required, if the GUI application is to be built). However, these versions of the libraries are preferred where available; some of the *older* versions are known to have issues. The IRENE package distribution does not ship with these libraries - they must be downloaded and installed separately before building or running the model. For Linux platforms, install the Linux distribution 'devel' packages for these dependencies, if appropriate versions are available. Please note their installation locations, as the 'include' and 'library' directories will need to be referenced in the model build configuration files. *See Appendix A for tips installing on Linux distributions other than CentOS*. For Windows, it is recommended they be installed in the "*C:/Program Files*" location.

HDF5 libraries version 1.8.8
http://www.hdfgroup.org/HDF5/release/obtain5.html

Boost template library version 1.58.0
http://www.boost.org/users/download/
The build of the Boost *binary* libraries is *NOT* required – only the header files are used.

Installation of the MPI libraries is optional; they are only required for the building of the multi-threaded version of the software.
Linux: OpenMPI library version 1.8.1
https://www.open-mpi.org/software/ompi/v1.8/
Windows: the Intel MPI Library 2017, Update 3 *Runtime Environment* files are included in the model

distribution, but the commercial development product is required for the *building* of the multi-threaded version. A free, *30-day trial* version is available at https://software.intel.com/en-us/intel-mpi-library .

Installation of the Qt and Qwt libraries is optional; they are only required for the building of the GUI application.
Qt libraries version 4.6.2 (release 2010.02.1) on Windows, version 4.7.2 on Linux
http://qt-project.org/downloads/

Qwt libraries version 5.2.1
http://sourceforge.net/projects/qwt/
Please note that proper build of Qwt requires use of the above Qt installation, especially 'qmake'.

Important: On Linux systems, an update of the 'path' environment variable may be necessary to ensure the correct version of qmake is used during the build process. Test this via the command 'which qmake'.

Required Build Tools

CMake version 2.6 or later
http://cmake.org/cmake/resources/software.html

Windows: Microsoft Visual Studio 2008 or 2012 (includes C++ compiler)
https://www.visualstudio.com/en-us/visual-studio-homepage-vs.aspx
A Windows-based FORTRAN compiler is also needed for a *full* source code compilation, but unfortunately, is not freely available. However, by default, the build process links to an included pre-compiled library of these FORTRAN-based routines.

Linux: Gnu compiler (C++ and FORTRAN) version 4.4.4 or later (problems using version 6.x have been reported)
http://gcc.gnu.org/
For CentOS or RHEL version 5.X installations, this is fulfilled with the "gcc44" package; an additional option in the build commands is required.

Recommended Build Tools

Python scripting language
http://www.python.org/

IRENE (AE9/AP9/SPM) Distribution Directory Structure

The IRENE (AE9/AP9/SPM) software distribution ships as a zipfile, which unzips to this directory structure:

```
Ae9Ap9
  bin
    win32              -  pre-built 32-bit Windows (XP/7) binaries
    win64              -  pre-built 64-bit Windows 7 binaries
  documents            -  release notes, user's guide, build instructions, license information, etc
```

modelData	- model database files
samples	- sample model run input files
unitTests	- collection of extended test cases that exercise the capabilities of the model
source	- source code root and build script (source code files are not present in base distribution)
Ae9Ap9Mpi	- Ae9Ap9 model source code tree
SpWx_Ae9Ap9	- source tree for space weather models upon which Ae9Ap9 depends
Ae9Ap9GuiMpi	- GUI application source tree
buildAe9Ap9.py	- python script for automated build on 'linux', 'win32' and 'win64' platforms

Once the required third-party dependencies are successfully installed, several build configuration files for the model source code will need to be updated with their install locations. The Ae9Ap9 model and utility executable files can then be built using the automated build script, or using the detailed set of commands. When completed, the executable files (and other supporting files) are placed in the platform-specific subdirectory of the 'bin'. The model should be run from this directory.

Steps to Building IRENE (AE9/AP9/SPM) Model Library and Applications

<u>Precursors, for currently *supported* platforms</u>

a) Download and install all required third-party support libraries for the desired platform using the standard instructions and procedures provided by the manufacturers. See the complete list of third-party support libraries above for download links and platform dependencies.

b) Taking note of the include and library paths for the third-party dependencies installed in the previous step, using a plain text editor, update these in the three build script configuration files:

- Ae9Ap9/source/SpWx_Ae9Ap9/Common/internal_utils.cmake

Scroll down in the file to where these lines are showing:

```
################################################################
# config_include_lib_paths
```

From here, all settings that may require modification are preceded with this line:

```
# <<== END-USER MODIFIABLE ==>>
```

The configuration file contains several platform-specific sections of commands that set these paths. Each section is started with a line similar to:

```
     if(CMAKE_BUILD_TYPE MATCHES "<platform_name>")
```

Find the appropriate platform section, and modify the existing lines to match the installation locations for the third-party libraries. For Windows, use *forward* slashes (‘`/`’) in path references. Note the ‘set’ of the `EXT_LIBS_ROOT` variable, and its subsequent reference `$(EXT_LIBS_ROOT)`, for ease of use in multiple places. For Windows, it is suggested to set `EXT_LIBS_ROOT` to `“C:/Program Files”`.

Some Windows-specific references to FORTRAN and HDF5 library file locations are also found in the *Ae9Ap9/source/SpWx_Ae9Ap9/Models/CMakeLists.txt* file, which may need updating.

- Ae9Ap9/source/Ae9Ap9Mpi/internal_utils.cmake

This file is updated in the same way as the previous file, if the GUI application is desired.

- Ae9Ap9/source/Ae9Ap9GuiMpi/trunk/Ae9Ap9Gui.pro

The Ae9Ap9Gui source tree utilizes ‘qmake’ (part of the Qt development environment) for the build process, and thus uses a slightly different file and syntax for defining the include and library paths. *Important*: on Linux systems, verify that the correct ‘qmake’ version is being used; update the ‘path’ environment variable as necessary.

The Qt include and library paths will be automatically set properly with the installation of the Qt software. This file must only explicitly set paths for the Qwt include and library paths, as shown below. Note the platform-specific sections are also present here.

```
win32 {
    INCLUDEPATH += c:\Qt\qwt-5.2.1\src
    LIBS += "c:\Qt\qwt-5.2.1\lib\libqwt5.a"
}
win64 {
    INCLUDEPATH += c:\Qt64\qwt-5.2.1\src
    LIBS += "c:\Qt64\qwt-5.2.1\lib\libqwt5.a"
}
unix {
    INCLUDEPATH += /nas/ExternalLibs/Qt/qwt-5.2/src
    LIBS += /nas/ExternalLibs/Qt/qwt-5.2/lib/libqwt.so
}
```

Precursors, for currently *unsupported* platforms

The current source distribution of IRENE model pacakage has built-in support for *linux*, *win32* and *win64* platforms. To utilize the software on other operating systems and development environments will require more extensive modifications to the existing configuration files and build scripts. Use the following steps to modify the build for additional platforms.

a) Verify that all third party dependencies are available and supported on the target operating system and development environment. This can be done by consulting the manufacturers' websites or customer support contacts.

b) The configuration files (described in the 'supported platform' section) will require new platform-specific sections to be added and updated, instead of editing existing settings. In the two 'internal_utils.cmake' files, the platform-specific settings are placed within sections that are delimited by the lines:
```
if(CMAKE_BUILD_TYPE MATCHES "<platform_name>")
   ...
endif (CMAKE_BUILD_TYPE MATCHES "<platform_name>")
```
Copy one of the existing sections, and insert it below, then change the platform name (in both places), and edit the lines within this new section as previously described.

Similarly, in the 'Ae9Ap9Gui.pro' file, copy an existing platform-specific section, then change the platform name and paths for the new platform.

c) Finally, within the Ae9Ap9 and SpWx_Ae9Ap9 source trees, a number of CMake scripts perform platform-specific tasks. It may be necessary to examine each of these to determine if such operations are required for this new platform. These platform-specific operation sections are delimited in the same manner as in the 'internal_utils.cmake' files.

Automated Build

The preferred method of building a source distribution of IRENE (AE9/AP9/SPM) on a *supported* platform is to utilize the python script called *buildAe9Ap9.py*, located in the Ae9Ap9/source directory of the distribution. This script will build the model library and the command-line and gui driver applications, then place them in a platform-specific subdirectory under Ae9Ap9/bin in the distribution. During the build process, the script will generate a log file called *buildAe9Ap9.log* in the Ae9Ap9/source directory; this can be used to identify any build errors encountered. At a command prompt in the 'Ae9Ap9/source' directory, invoke the python build script, specifying the operating system name and, if needed, the mode (defaults to 'release'):

```
  python buildAe9Ap9.py --os=<platform_name> [ --mode=<type> ] [ --nompi ] [ --nogui ]
```

where `<platform_name>` can be `linux` , `win32` or `win64`, and `<type>` can be `release` or `debug` .
By default, both the single- and multi-threaded versions of the programs are built, unless the optional argument '`--nompi`' (required for *win32* builds) is specified. The GUI application is also built by default; to skip this, specify the optional argument '`--nogui`'. For RedHat EL or CentOS 5.x installations only, the `--gcc=gcc44` option should also be included.
Warning: 'debug' mode increases program execution time by a factor of 10 to 20.

Manual Build

Because python is not required for IRENE (AE9/AP9/SPM) and is not automatically installed on all operating systems, the following instructions are provided for building the model and applications manually. The steps shown below are used to build Linux or Windows release-mode binaries. Substitute 'linux' with 'win32' or 'win64' in these commands as needed. Additional options that are needed for building debug-mode binaries are shown where required. For 'win64' builds only, the first two cmake commands also require the option: ' `-G"Visual Studio 9 2008 Win64"` ' (including quotes).

1. From a command prompt, navigate to the Ae9Ap9/source directory.

2. Create a directory (mkdir), appropriately named 'SpWx_Ae9Ap9_linux_release' or 'SpWx_Ae9Ap9_win*XX*_release', then navigate to that directory.

3. Enter the command:
   ```
    cmake -DCMAKE_BUILD_TYPE=linux -DBUILD_AE9AP9_SPWX=ON ../SpWx_Ae9Ap9
   ```
   For Windows platforms, replace `linux` with the appropriate `win32` or `win64`.
   For RedHat EL or CentOS **5**.x Linux installations, also include the `-DUSE_GCC44=ON` parameter.
   For debug mode, also include the `-DCMAKE_DEBUG=Y` parameter [*debug mode slows execution ~x10*].

4. Assuming the previous run of *cmake* produced no errors, it will have generated platform-specific make scripts in the newly created directory. On Linux, these will be in the form of makefile(s). On Windows, these will take the form of Visual Studio solution and project files. To perform the final step of the build process for this portion of the source tree, do one of the following:

   Linux: Enter the command: `make`

Windows: From Windows Explorer, double-click the solution file *SpWx.sln* in the Ae9Ap9/source/ SpWx_Ae9Ap9_win*XX*_release directory. From within Visual Studio, select the project "ALL_BUILD" in the project explorer pane. Verify that the build type is shown as '*Release*' mode in the toolbar along the top. Select "Build ALL_BUILD" from the Build menu. Close Visual Studio when the build is complete.

This completes the build process for the SpWx_Ae9Ap9 portion of the source tree.

5. At the command prompt, navigate back to the Ae9Ap9/source directory

6. Create a directory 'Ae9Ap9_linux_release' below it and then navigate to that directory.

7. Enter the command:
```
 cmake -DCMAKE_BUILD_TYPE=linux -DSPWX_BUILD=../SpWx_Ae9Ap9_linux_release ../Ae9Ap9Mpi
```
By default, both the single- and multi-threaded versions of the programs are built. To build *only* the single-threaded version, add the optional argument '`-DUSE_MPI=OFF`' (required for *win32* builds). For RedHat EL or CentOS **5**.x Linux installations, also include the `-DUSE_GCC44=ON` parameter. For debug mode, also include the `-DCMAKE_DEBUG=Y` parameter.
Warning: 'debug' mode increases program execution time by a factor of 10 to 20.

8. Once again, this is a two-step build process with the next step being platform-specific.

   Linux: Enter the command: `make`

   Windows: From Windows Explorer, double-click the solution file *ae9ap9.sln* in Ae9Ap9/source/Ae9Ap9_win*xx*_release. From Visual Studio, select project "ALL_BUILD". Verify that the current build type is '*Release*' mode in the toolbar along the top. Select "Build ALL_BUILD" from the Build menu. Close Visual Studio when the build is complete.

   Assuming all builds completed successfully, this completes the build of the Ae9Ap9 source tree.

9. At the command prompt, navigate back to the Ae9Ap9/source directory. If no GUI application is desired, skip ahead to step 13.

10. Create a directory 'Ae9Ap9Gui_linux_release' below and navigate to that directory. (The precompiled 'win32' and 'win64' versions of the GUI program are 32-bit only. The 'win64' version of the GUI program has *not* been tested; 64-bit QT and Qwt libraries would be required.)

11. Enter the command:
```
 cmake -DCMAKE_BUILD_TYPE=linux ../Ae9Ap9GuiMpi
```
For debug mode, also include the `-DCMAKE_DEBUG=Y` parameter. When building a single-threaded model, disable the Mpi-related controls in the GUI with the parameter '`-D NOMPI=On`'.

12. We once again find ourselves at the platform-specific build step.

    Linux: Enter the command: `make`

    Windows: From Windows Explorer, double-click the solution file *Project.sln* in

Ae9Ap9/source/Ae9Ap9Gui_win*xx*_release. From Visual Studio, select project “ALL_BUILD”. Verify that the current build setting is ‘*Release*’ mode in the toolbar along the top. Select “Build ALL_BUILD” from the Build menu. Close Visual Studio when the build is complete.

This completes the build of all source trees. All that remains is to copy binary executable files to the appropriate location beneath Ae9Ap9/bin.

13. Navigate to the Ae9Ap9/bin directory and create a ‘linux’ or ‘win*XX*’ subdirectory; navigate into that directory.

14. From the command prompt, perform the following platform-specific copy commands:
Linux:

```
cp ../../source/Ae9Ap9_linux_release/trunk/bin/CmdLineAe9Ap9 .
cp ../../source/Ae9Ap9_linux_release/trunk/bin/EphemTask .
cp ../../source/Ae9Ap9_linux_release/trunk/bin/FluxTask .
cp ../../source/Ae9Ap9_linux_release/trunk/bin/LegFluxTask .
cp ../../source/Ae9Ap9_linux_release/trunk/bin/ConcatTask .
cp ../../source/Ae9Ap9_linux_release/trunk/bin/FlueDoseTask .
cp ../../source/Ae9Ap9_linux_release/trunk/bin/AggregTask .
cp ../../source/Ae9Ap9_linux_release/trunk/bin/ConvertTask .
cp ../../source/Ae9Ap9_linux_release/trunk/bin/IntegralPlasma .
cp ../../source/Ae9Ap9_linux_release/trunk/bin/TotalDose .
cp ../../source/Ae9Ap9Gui_linux_release/trunk/Ae9Ap9Gui .     (only if GUI was built)
(the following may be omitted if only the single-threaded versions were built)
cp ../../source/Ae9Ap9_linux_release/trunk/bin/CmdLineAe9Ap9Mpi .
cp ../../source/Ae9Ap9_linux_release/trunk/bin/EphemTaskMpi .
cp ../../source/Ae9Ap9_linux_release/trunk/bin/FluxTaskMpi .
cp ../../source/Ae9Ap9_linux_release/trunk/bin/LegFluxTaskMpi .
cp ../../source/Ae9Ap9_linux_release/trunk/bin/ConcatTaskMpi .
cp ../../source/Ae9Ap9_linux_release/trunk/bin/FlueDoseTaskMpi .
cp ../../source/Ae9Ap9_linux_release/trunk/bin/AggregTaskMpi .
cp ../../source/Ae9Ap9_linux_release/trunk/bin/ConvertTaskMpi .
```

Windows:

```
copy ..\..\source\Ae9Ap9_winXX_release\trunk\bin\Release\CmdLineAe9Ap9.exe .
copy ..\..\source\Ae9Ap9_winXX_release\trunk\bin\Release\EphemTask.exe .
copy ..\..\source\Ae9Ap9_winXX_release\trunk\bin\Release\FluxTask.exe .
copy ..\..\source\Ae9Ap9_winXX_release\trunk\bin\Release\LegFluxTask.exe .
copy ..\..\source\Ae9Ap9_winXX_release\trunk\bin\Release\ConcatTask.exe .
copy ..\..\source\Ae9Ap9_winXX_release\trunk\bin\Release\FlueDoseTask.exe .
copy ..\..\source\Ae9Ap9_winXX_release\trunk\bin\Release\AggregTask.exe .
copy ..\..\source\Ae9Ap9_winXX_release\trunk\bin\Release\ConvertTask.exe .
copy ..\..\source\Ae9Ap9_winXX_release\trunk\bin\Release\IntegralPlasma.exe .
copy ..\..\source\Ae9Ap9_winXX_release\trunk\bin\Release\TotalDose.exe .
copy ..\..\source\Ae9Ap9Gui_winXX_release\trunk\release\Ae9Ap9Gui.exe .   (only if GUI was built)
(the following may be omitted if only the single-threaded versions were built or 32-bit Windows)
copy ..\..\source\Ae9Ap9_winXX_release\trunk\bin\Release\CmdLineAe9Ap9Mpi.exe .
copy ..\..\source\Ae9Ap9_winXX_release\trunk\bin\Release\EphemTaskMpi.exe .
copy ..\..\source\Ae9Ap9_winXX_release\trunk\bin\Release\FluxTaskMpi.exe .
copy ..\..\source\Ae9Ap9_winXX_release\trunk\bin\Release\LegFluxTaskMpi.exe .
copy ..\..\source\Ae9Ap9_winXX_release\trunk\bin\Release\ConcatTaskMpi.exe .
copy ..\..\source\Ae9Ap9_winXX_release\trunk\bin\Release\FlueDoseTaskMpi.exe .
copy ..\..\source\Ae9Ap9_winXX_release\trunk\bin\Release\AggregTaskMpi.exe .
copy ..\..\source\Ae9Ap9_winXX_release\trunk\bin\Release\ConvertTaskMpi.exe .
```

Testing of the Build

At this point, the source distribution has been built and deployed to a location where it should run successfully using test input files. With the command prompt at the 'Ae9Ap9/bin/*<platform_name>*' directory, verify the operational status of the software installation by entering this command:

```
  CmdLineAe9Ap9 -i ../../samples/Ap9ShortInput.txt
```

The command-line utility should run successfully and produce output files in the 'Ae9Ap9/samples' directory, matching the contents of files in the 'Ae9Ap9/samples/expectedOutput' directory.

Similarly, the GUI application can also be invoked from this location with the command:

```
  Ae9Ap9Gui
```

or, double-clicking its icon in the file manager window. Refer to the User's Guide for instructions and examples for using the GUI application.

If either fails to load, it is likely that a dependency (.dll file on windows) or (.so on linux) has not been installed in a location on the appropriate system path. For example, if a windows debug build were created and installed to Ae9Ap9/bin/win*XX*_debug directory, one must copy the dependent dlls from the pre-built windows release mode build directory Ae9A9/bin/win*XX*_release to that location.

On Windows systems, problems with the GUI application have been reported. Please verify that the GUI source code, the Qt and Qwt dependencies are all in the same 'release' or 'debug' mode – no mixing of 'release' and 'debug' parts is permitted.

To report build problems with the distribution, please include any log files generated by the process and all error messages displayed in the command window. We will be unable to diagnose the source of the problem without this information.

If successful in porting the software to a currently unsupported platform, please send a copy of the modified files to the development team (ae9ap9@vdl.afrl.af.mil). This information will be greatly appreciated, and will be incorporated into future releases.

Appendix A: Non-CentOS Linux build configuration tips

The following configuration settings and steps were used for a full compilation and installation of the IRENE model package software on a Linux Ubuntu 14.04 machine. It is assumed that these settings, or similar, will also enable the successful installation on other package-based Linux distributions.

- Install the development packages for HDF5, Boost, OpenMPI and QT ('libqt4-dev'), if not already present. (skip the QT install if the GUI application is not needed)
  - The header files for these packages are placed in '/usr/include', and their associated library files are placed in '/usr/lib/x86_64-linux'; both are system default locations.
- Install the CMake package, if not already present.

  ----skip the next two steps if the GUI application is not needed
- Download 'qwt-5.2.1.zip' from http://sourceforge.net/projects/qwt/files/qwt/5.2.1/
  - Unzip the file, cd to the new 'qwt-5.2.1' subdirectory
  - `$ qmake`
  - `$ make`
  - `$ sudo make install`
- Edit the 'Ae9Ap9Gui.pro' file, in the 'unix' section, changing it to be:

```
unix {
    INCLUDEPATH += /usr/include/qt4
    INCLUDEPATH += /usr/include/qt4/QtCore
    INCLUDEPATH += /usr/include/qt4/QtGui
    INCLUDEPATH += /usr/local/qwt-5.2.1/include
    LIBS += /usr/lib/x86_64-linux-gnu/libQtGui.so
    LIBS += /usr/lib/x86_64-linux-gnu/libQtCore.so
    LIBS += /usr/local/qwt-5.2.1/lib/libqwt.so
}
```

- In a terminal window at the 'Ae9Ap9/source' directory, invoke the build script:
  - `python buildAe9Ap9.py --os=linux`

    (if the GUI application is not needed, add the argument `--nogui`)
    - No modification of the two 'internal_utils.cmake' files is necessary, because the 3rd-party packages that are being used have been installed in the standard system directories, and so do not need to be explicitly specified.
    - Many warnings regarding non-existent directories may be seen during the compilation process, but these do not affect the compilation success.
- All the required executable files should now be present in the 'Ae9Ap9/bin/linux' directory.
- For the execution of the GUI application, setting of an environment variable may be needed:
  - `$ export LD_LIBRARY_PATH=/usr/local/qwt-5.2.1/lib/`
  - alternatively, the library files contained here may be copied to '/usr/lib/x86_64-linux'

Contact Information

Please send any questions, comments and/or bug reports to: ae9ap9@vdl.afrl.af.mil

The IRENE model package and related information can be obtained from AFRL's Virtual Distributed Laboratory (VDL) website: https://www.vdl.afrl.af.mil/programs/ae9ap9