# Appendix D

# EGS5 INSTALLATION GUIDE

Hideo Hirayama and Yoshihito Namito
Radiation Science Center
Advanced Research Laboratory
High Energy Accelerator Research Organization (KEK)
1-1 Oho Tsukuba-shi Ibaraki-ken 305-0801 JAPAN

Alex F. Bielajew and Scott J. Wilderman
Department of Nuclear Engineering and Radiological Sciences
The University of Michigan
2355 Bonisteel Boulevard
Ann Arbor, MI 48109, USA

Walter R. Nelson
Department Associate in the Radiation Physics Group (retired)
Radiation Protection Department
Stanford Linear Accelerator Center
2575 Sand Hill Road Menlo Park, CA 94025, USA

This EGS5 Installation Guide is Appendix D of a document called
SLAC-R-730/KEK-2005-8, which can be obtained from the SLAC and KEK web sites.

## D.1 Installation of EGS5

The current version of the EGS5 code system run only on Unix-based operating system. On Windows platforms, this release of EGS5 can be run on top of **Cygwin** or any other Linux environment emulator for PCs. Copies of **Cygwin**, along with setup programs, installation instructions, users guides, and other documentation, can be obtained for free at the **Cygwin** website, http://www.cygwin.com.

The EGS5 system is distributed as a compressed (with **gzip**) Unix "tar" archive, **egs5.tar.gz**. It is trivially "installed" by typing:

```
% gunzip -c egs5.tar.gz | tar vxf -
```

from the command line in any desired directory into which a user places **egs5.tar.gz** (to somewhat mimic the organizational structure of EGS4, this command should be run from the user's top level directory). When this command is executed, a directory called **egs5** will be created, under which the subdirectories containing all of the EGS5 FORTRAN, data, tutorial, and sample problem files will be placed.

Although it employs some non-standard extensions to FORTRAN-77 the basic source code in EGS5 avoids the use of all system-dependent utilities, and all that is required to run EGS5 on any Unix-like platform is a FORTRAN compiler[1]. Further, all of the tutorial, sample, and auxiliary user codes provided in the EGS5 distribution should run without modification if the FORTRAN library routines `FDATE` and `ETIME` are available. Likewise, the sample shell scripts for controlling the compilation and execution of EGS5 jobs will run after minimal modification on the part of the user.

## D.2 Sample Scripts for Running EGS5

The EGS5 distribution comes with two shell scripts which can be used (with minimal modification) to compile, run, and re-run generic EGS5 user codes. The five steps which are required to use the sample script "egs5run" are listed below. Example implementations follow.

1. A copy of egs5run (or a symbolic link to a global copy of egs5run) must be placed in the subdirectory containing the user code.
2. In the egs5run file, the user must specify the directory in which the main EGS5 distribution was installed. This is done by setting the shell script variable `BASKET` to the name of the

[1]The one system-dependent FORTRAN implementation issue which an EGS5 user may encounter is with the `NAMELIST` extension. Different compilers may require different leading characters in the end-of-list tag, usually either `&END` or `/END`.

**egs5** directory (preferably using the full Unix pathname). `BASKET` is analogous to `HEN_HOUSE` of EGS4, in that all of the basic EGS5 system resides in this single place.

3. Also in egs5run, the user must specify the operating system type under which the current computation is to be performed by defining the variable `MY_MACHINE`. This is required primarily so that egs5run can select the appropriate compiler options.

4. If the user's operating system is not among the defaults included with the egs5run script, the user will have to modify the section of egs5run in which the name of the FORTRAN compiler and the relevant compiler options are set. An example is given below.

5. Lastly, and also in egs5run, the user must select the level of optimization to be used, if any, during compilation, by specifying the value of the variable `OPT_LEVEL`.

The preamble to egs5run contains some examples of how the variables described above might be set in egs5run:

```
#   BASKET=/home/wrn/egs5/
#   MY_MACHINE=Linux
#   OPT_LEVEL=O
#
#   BASKET=/home/Ralph/egs5/
#   MY_MACHINE=Cygwin-Linux
#   OPT_LEVEL=O2
#
#   BASKET=/afs/slac.stanford.edu/g/rp/egs5
#   MY_MACHINE=sparc
#   OPT_LEVEL=
```

The first example corresponds to a user with login id "wrn" on machine running Linux, with EGS5 installed from his top level (home) directory. The second example is representative of what a PC user named "Ralph" might see if he were running **Cygwin**, again with EGS5 installed from his top level directory. The final example is typical of what a user accessing a globally available version of EGS5 across an afs file system might see.

Note that the variable `MY_MACHINE` can be defined as anything the user wishes, provided that compiler options are included for that OS name later in egs5run. The version of egs5run in the current EGS5 distribution has defined compiler options for four operating systems: Linux, Cygwin-Linux, sparc and Digital-Unix. If EGS5 is to be used on some other Unix-like platform, the user must edit egs5run to specify the name of the FORTRAN compiler and any compiler options which are necessary. This can be done quite simply by changing the 5 instances of `user_defined_*` with the appropriate parameters in the section of egs5run shown below:

```
elif test "$MY_MACHINE" = "user_defined_machine"
```

```
then
  COMPILER="user_defined_compiler"
  DEBUG="user_defined_debug_flags"
  CFLAGS="user_defined_compilation_flags"
  if test "$OPT_LEVEL" = ""
    then
      OPTIMIZED=""
  else
      OPTIMIZED="-$OPT_LEVEL user_defined_optimizations"
  fi
```

**Running egs5run:** There are five options for executing egs5run, invoked by specifying a single command line argument, as shown below:

| Command line | Action of egs5run |
|---|---|
| `egs5run` | Compile user code and execute. |
| `egs5run comp` | Compile user code but do not execute. |
| `egs5run pegs` | Compile pegs-only user code and execute. |
| `egs5run db` | Compile user code for debug (does not execute). |
| `egs5run cl` | Clear out files (and links) and exit egs5run. |

When egs5run is executed in normal mode (*i.e.*, with no argument), the following sequence of actions is initiated:

1. Various files and links created in previous runs (if any) are deleted.
2. The specified system name `MY_MACHINE` and compiler are echoed.
3. The user is asked to key-in the name of the FORTRAN file containing the user code for this run. The file name provided by the user *must* have a `.f` extension. Is the file is not found, the script will exit.
4. A file named egs5job.f is created using the specified user code file, and all files with `.f` extensions in the following directories:

   | | |
   |---|---|
   | **user_auxcode** | user defined auxiliary subroutines, if present. |
   | **$BASKET/egs** | EGS related subroutines. |
   | **$BASKET/pegs** | subroutines related to PEGS. |
   | **$BASKET/auxcode** | auxiliary subroutines provided with EGS5. |

   Note that file extensions should not be keyed-in when specifying file names.
5. Symbolic links are created pointing to the following directories containing files of `COMMON` blocks included by the various source code files:

   | | |
   |---|---|
   | **$BASKET/include** | `COMMON` block files for EGS. |
   | **$BASKET/auxcommons** | `COMMON` block files for EGS5 auxiliary subroutines. |
   | **$BASKET/pegscommons** | `COMMON` block files for PEGS. |

6. The user is then asked to key-in the name of the EGS input file for this run. Note that for many applications, all of the necessary problem-specific data can be specified in the user code itself, and so an input problem data file need not be used. When this is the case, the user may simply enter a carriage return. If an EGS problem data file is to be used, it must have a `.data` extension. If the data file has the same name as the user code file (apart from the extensions `.f` and `.data`, of course), this can be specified by keying in a carriage return. If the user specified file exists, it is copied to file called "egs5job.inp." If the file does not exist, egs5run will create an empty (dummy) file called egs5job.inp, which is opened as unit 4 for input in many EGS5 user codes.

7. The user is next asked to key-in the name of the input data file used by PEGS. This file must have a `.inp` extension. A carriage return issued in response to this question will cause egs5run to look for `.inp` files named similarly to first the user code and then the data file. If any of the files exist, a symbolic link named "pgs5job.pegs5inp" will be created, as this file is opened by default on unit 25 in PEGS. If no appropriate `.inp` file is found, the script will exit.

8. The egs5job.f will then be compiled, with executable named "egs5job.exe."

9. If egs5job.exe does not exist because of compilation errors, the script will exit.

10. Next, a symbolic link to the data directory, $BASKET/data, will be created.

11. The user will then be asked if this user code requires input to be keyed-in from the terminal (on unit 5, standard input).

12. egs5job.exe will be executed either in the foreground (if interactive input was specified) or in the background. If the job is executed in the background, standard output (unit 6) is redirected to a file named "egs5job.log." Note that the user may elect to explicitly open unit 6 in the user code, if desired.

When egs5run is executed with one of the three arguments, "cl," "db," or "comp," the script follows the basic flow described above except that it exits either immediately after cleaning out old links (with the "cl" option) or just prior to executing the compiled code (with the "db" or "comp" arguments). When egs5run is executed with "pegs" as an argument, only the **pegs** files and the `BLOCK_DATA` FORTRAN source files from **egs** are included with the user code when **egs5job.f** is created. With that exception, egs5run with "pegs" as an argument runs exactly as it does with no argument. Note that because none of the main physics routines of EGS are included in **egs5job.f**, user codes based on PEGS only need not contain either `HOWFAR` or `AUSGAB` subroutines.

**Example transcript from egs5run** Below is a transcript of an egs5run session using the tutorial problem in **tutor1**. (Note that input keyed-in by the user is denoted by `<-------- USER_INPUT`.)

```
% egs5run
============================
```

```
egs5run script has started
==========================

working directory is /afs/slac.stanford.edu/g/rp/egs5/tutorcodes/tutor1

------------------------------------------------------------
Erasing files (and links) from previous runs (if they exist)
------------------------------------------------------------
                  OS_TYPE  = sparc
Your Compiler is f77

--------------------------------------
        Enter name of User Code
      (file extension must be '.f')
--------------------------------------
tutor1                                  <-------- USER_INPUT

----------------------------------------
      Enter name of READ(4) data file
      (file extension must be '.data')
   (<CR> for same file name as User Code)
----------------------------------------
                                        <-------- USER_INPUT (<CR>)
  --> Empty file created as egs5job.inp

----------------------------------------
  Enter name of UNIT(25) (pegs input file)
      (file extension must be '.inp')
    (<CR> for same file name as data file
      or same file name as User Code)
----------------------------------------
                                        <-------- USER_INPUT (<CR>)
  --> tutor1.inp linked to pgs5job.pegs5inp

-----------------------------------
Compiling (with optimization of O2)
-----------------------------------

-----------------------------------------------
  Does this user code read from the terminal?
  (Enter 1 for yes, anything else for no)
-----------------------------------------------
0                                       <-------- USER_INPUT

      ******************************************************
```

```
      * User code tutor1.f has been compiled and is starting *
      ******************************************************

 Running egs5job.exe in background

========================
egs5run script has ended
========================
```

**Running run5again:** If a user wishes to run previously compiled user code with different input data on either unit 4 (EGS problem data) or unit 25 (PEGS material data) or both, the script run5again can be executed instead of egs5run. The run5again script takes no arguments, and does all of what egs5run does except that it does not construct or compile **egs5job.f**. run5again will execute either an existing egs5job.exe program or any previously compiled egs5job.f file has been renamed and has a `.exe` file extension. When run5again (which requires no set up on the part of the user) is executed, the following actions are performed:

1. Symbolic link pgs5job.pegs5inp and file egs5job.inp from previous runs are deleted.
2. The user is asked to key-in the EGS data file name to be copied to egs5job.inp and opened on unit 4, as in egs5run, above.
3. The user is asked to key-in the name of the PEGS data file to be linked to pgs5job.pegs5inp and opened on unit 25, as with egs5run.
4. The user is then asked to specify the name of an existing executable (with file extension `.exe` for this run. If a carriage return is entered, run5again will look for an existing egs5job.exe executable to use. If neither the specified executable nor egs5job.exe exist, the script will exit.
5. The user will finally be asked if this user code requires input that is to be keyed-in (to unit 5, standard input).
6. Either the specified executable or egs5job.exe will be executed, either in the foreground (if interactive input was specified) or in the background. If the job is executed in the background, standard output (unit 6) is redirected to a file named "egs5job.log," as with egs5run.

**Example transcript from run5again** Below is a sample transcript from a run5again job:

```
% run5again
============================
run5again script has started
============================
```

```
working directory is /home/user/egs5/tutorcodes/tutor1

----------------------------------------
      Enter name of READ(4) data file
      (file extension must be '.data')
----------------------------------------
tutor1                                      <-------- USER_INPUT

  --> Empty file created as egs5job.inp

----------------------------------------
  Enter name of UNIT(25) (pegs input file)
      (file extension must be '.inp')
   (<CR> for same file name as data file)
----------------------------------------
                                            <-------- USER_INPUT (<CR>)

  --> d4file used, tutor1.inp linked to pgs5job.pegs5inp

-----------------------------------
    Enter name of the executable
  (file extension must be '.exe')
     (<CR> to use egs5job.exe)
-----------------------------------
                                            <-------- USER_INPUT (<CR>)

-----------------------------------------------
  Does this user code read from the terminal?
  (Enter 1 for yes, anything else for no)
-----------------------------------------------
0                                           <-------- USER_INPUT
      ******************************************
      * Previously compiled user code is starting *
      ******************************************

 Running egs5job.exe in background

=========================
run5again script has ended
=========================
```