The ISML Fortran Library Tutorial

Updated: August 2012

Table of Contents

Section 1: Introduction

The IMSL library contains over one thousand Fortran subroutines and functions. These routines provide easy access to high quality implementations of numerical methods in mathematics and statistics.

Mathematical routines include: linear algebra, eigensystem analysis, interpolation and approximation, integration and differentiation, solving differential equations, Fourier and Laplace transforms, finding solutions of nonlinear equations, optimization, and special functions.

Statistical routines include: regression correlation, analysis of variance, categorical and discrete data analysis, nonparametric statistics, tests of goodness of fit and randomness, time series analysis and forecasting, covariance structures and factor analysis, discriminant analysis, cluster analysis, sampling, survival analysis, life testing, and reliability, multidimensional scaling, density and hazard estimation, probability distribution functions and inverses, and random number generation.

The IMSL subroutines are called like ordinary Fortran subroutines, then the IMSL libraries are linked into the executable when the calling program is loaded. As discussed below, the method of linking depends on the system you are using.

IMSL is proprietary software and cannot be transferred to a machine that is not licensed to run it.

For up-to-date information on where you can access this software, visit
https://stat.utexas.edu/training/software-information

Section 2: Getting Started

2.1 On the Unix Machines

The IMSL libraries are not in the default library search path. There is a script which will modify your environment to allow you to call the IMSL libraries, installed on the ITS systems. To run this script type:

```
source /usr/local/imsl/CTT6.0/ctt/bin/cttsetup.csh
```

The command above can be put at the end of your `.cshrc` (or `.profile`) so that it will be executed automatically for you. It is important that you issue this command only after you have set your environment variables such as your PATH and MANPATH. Once you have executed the script, use:

```
f90( or f77) [ compiler options ] $FFLAGS  $LINK_FNL filename
```

to compile and load the program in filename using the default (shared) version of the IMSL library. It is also possible to explicitly specify the use of the static or shared library. To load the static or shared versions of the IMSL Fortran numeric libraries, respectively, use:

```
    f90( or f77) [ compiler options ] $FFLAGS  $LINK_FNL_STATIC filename
```
or
```
    f90( or f77) [ compiler options ] $FFLAGS  $LINK_FNL_SHARED filename
```

To generate an executable eitest that uses the shared version of the IMSL libraries, compile the example file `eitest.f` with:

```
    f90( or f77) -o eitest $FFLAGS  $LINK_FNL eitest.f
```

2.2 On Departmental Alpha OpenVMS Machines

Several logicals pointing to linker option files have been defined for the IMSL Fortran Libraries. They are:

```
Logical           Points to
imslieee_share    The IEEE shared library linker options file
                  YELLOW$DRA5:[IMSL30.ADTVNI.LIB.AXPVMS]IMSLSHRE.OPT

imslieee_static   The IEEE static library linker options file
                  YELLOW$DRA5:[IMSL30.ADTVNI.LIB.AXPVMS]IMSLSTCE.OPT

imslibg_share     The G_FLOAT shared library linker options file
                  YELLOW$DRA5:[IMSL30.ADTVNI.LIB.AXPVMS]IMSLSHRG.OPT

imslibg_static    The G_FLOAT static library linker options file
                  YELLOW$DRA5:[IMSL30.ADTVNI.LIB.AXPVMS]IMSLSTCG.OPT

imslpsect         The program section linker options file
                  YELLOW$DRA5:[IMSL30.ADTVNI.LIB.AXPVMS]IMSLPSECT.OPT
                  For a detailed discussion of the contents of this
                  file see chapter 3 of the OpenVMS Linker Utility
                  Manual.
```

These logicals are used to compile Fortran programs with the desired version of the IMSL Library. The /opt switch identifies them as options files to the linker. To compile the test program `eitest.f` using the shared IEEE floating point library use the command:

```
fortran/float=ieee eitest.f
link/nomap eitest.obj, imslieee_share/opt, imslpsect/opt
```

The following commands will compile and link `eitest.f` using the static IEEE floating point library:

```
fortran/float=ieee eitest.f
link/nomap eitest.obj, imslieee_static/opt, imslpsect/opt
```

The commands below compile `eitest.f` with the shared G_FLOAT floating point library:

```
fortran/float=g_float eitest.f
link/nomap eitest.obj, imslibg_share/opt, imslpsect/opt
```

The static G_FLOAT floating point library is used with the commands:

```
fortran/float=g_float eitest.f
link/nomap eitest.obj, imslibg_static/opt, imslpsect/opt
```

Note: The IMSL Library is designed to allow quantities to underflow gracefully and be treated as zero. However, due to a flaw in the DEC runtime libraries, floating underflow warning messages may be generated. DEC has been made aware of the problem and will fix it in a future release of the libraries. *Users should expect and not be concerned by these messages*.

An example program

You may be using a completely different machine or operating system, but this example will provide some guidance even in that case. The following program uses the IMSL routines EI and UMACH.

```
C-----------------------------------------------------------------------
C
C Purpose:    Call the IMSL routine EI to evaluate the exponential integral
C             for arguments greater than zero and the Cauchy principal value
C             for arguments less than zero.
C
C                                     Declare variables
      INTEGER    NOUT
      REAL       EI, VALUE, X
      EXTERNAL   EI, UMACH
C                                     Compute
      X     = 1.15
      VALUE = EI(X)
C                                     Print the results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) X, VALUE
C  EI( 1.150) =  2.304
99999 FORMAT (' EI(', F6.3, ') = ', F6.3)
      END
```