Calling C functions from NONMEM®

UPPSALA UNIVERSITET

Rikard Nordgren, Chunli Chen, Ulrika S H Simonsson and Sebastian G Wicha
Pharmacometrics group, Department of Pharmaceutical Biosciences, Uppsala University

Objective

Some PK-PD modelling activities require the use of root finding techniques when an explicit solution of a function is not available. In a recent pharmacometric study [1], Loewe additivity [2], a criterion to define a pharmacodynamic additive interaction, was required, which only has an implicit solution. A C function had previously been developed to solve this equation using the GNU scientific library [3]. When implementing the model in NONMEM® [4], we needed either a rootfinder written in Fortran or a way to be able to call the C function from NONMEM. We choose the second option to be able to reuse our code. There are two ways of calling external functions from NM-TRAN: via abbreviated function calls or via verbatim code function calls. Both these ways will be explained below as well as usage of the enhanced abbreviated function calls of NONMEM 7.4

The C function

The Loewe Additivity rootfinder function is a C function taking 12 doubles as arguments and returning one double. It has the following prototype:

```
double Ecomb_LA_root(
  double C_A, double C_B, double Emax_A, double EC50_A,
  double H_A, double Emax_B, double EC50_B, double H_B,
  double Int_AB, double Int_BA, double EC50_Int_AB,
  double EC50_Int_BA);
```

We would like to be able to call this function from NM-TRAN. When using abbreviated functions in NONMEM 7.3 the limitation of maximum nine arguments has to be overcome. This can be done by splitting the function in two. The first function call will offload three of the arguments into global variables and the second function call will take the remaining 9 arguments, combine these with the first three and do the actual calculation using all 12 arguments. As NONMEM only uses process-level parallelism the use of global variables for offloading does not cause any race conditions.

The two new C functions will have the following prototypes:

```
void store_arguments(double Int_BA, double EC50_Int_AB, double EC50_Int_BA);
double wrapper_Ecomb_LA_root(double C_A, double C_B, double Emax_A,
  double EC50_A, double H_A, double Emax_B, double EC50_B, double H_B,
  double Int_AB);
```

Note that this splitting of the C function is not needed when using abbreviated functions in NONMEM 7.4 or verbatim function calls.

Fortran 2003 interface with iso_c_binding

Before the creation of the Fortran 2003 ISO standard interoperability between C and Fortran was very much dependent on the compilers. Fortran 2003 specifies a new portable intrinsic module iso_c_binding [5] that gives a well defined way of interfacing C from Fortran. As the only C type that is used in our functions is double, which is an interoperable type of kind c_double, the functions are interoperable. Here is the interface definition for the second C function (note the keyword `value` that is important as C functions are call by value and Fortran functions call by reference):

```
interface
  function rootfinder(C_A, C_B, Emax_A, EC50_A, H_A, Emax_B, &
    EC50_B, H_B, Int_AB, Int_BA, EC50_Int_AB, EC50_Int_BA) &
    bind(C, name="Ecomb_LA_root")
    use iso_c_binding
    real(kind=c_double),value :: C_A, C_B, Emax_A, EC50_A, H_A, &
      Emax_B, EC50_B, H_B, Int_AB, Int_BA, EC50_Int_AB, EC50_Int_BA
    real(kind=c_double) :: rootfinder
  end function
end interface
```

To use an abbreviated function an extra fortran wrapper function has to be used. Note that the NONMEM kind DPSIZE and the c_double kind both represent the same double precision type. The code below is for NONMEM 7.4

```
FUNCTION ROOTFINDERW(X, X1, X2, NDIM)
   USE SIZES, ONLY: DPSIZE
   INTEGER NDIM
   REAL(KIND=DPSIZE), INTENT(IN) :: X(NDIM)
   REAL(KIND=DPSIZE), INTENT(IN OUT) :: X1(NDIM), X2(NDIM)
   REAL(KIND=DPSIZE) :: ROOTFINDERW
   ROOTFINDERW=ROOTFINDER(X(1), X(2), X(3), X(4), X(5), X(6), X(7), &
     X(8), X(9), X(10), X(11), X(12))
END FUNCTION
```

The code needed for NONMEM 7.3 abbreviated functions need to repeat the procedure for the argument offloading functions but this time as a subroutine as it has no return value. See [1] for details.

Calling Fortran from NM-TRAN

Abbreviatied function call NONMEM 7.3

Using an abbrevitated function in NONMEM 7.3 is a matter of setting the arguments in the VECTRA for the call to FUNCA and in VECTRB for the call to FUNCB. Note that the call to the offloading function has to return a dummy value to fit the interface of the abbreviated function.

```
VECTRA(1) = C_A
VECTRA(2) = C_B
VECTRA(3) = Emax_A
VECTRA(4) = EC50_A
VECTRA(5) = H_A
VECTRA(6) = Emax_B
VECTRA(7) = EC50_B
VECTRA(8) = H_B
VECTRA(9) = Int_AB
VECTRB(1) = Int_BA
VECTRB(2) = EC50_Int_AB
VECTRB(3) = EC50_Int_BA
DUMMY = FUNCB(VECTRB)
ROOT = FUNCA(VECTRA)
```

Abbreviated function call NONMEM 7.4

NONMEM 7.4 has generalized the abbreviated function functionality and there is no longer a limit to how many arguments a function can have. The function and the argument vector can now have any name. Funcion name and number of arguments must be declared in a $ABBR record.

```
$ABBR FUNCTION ROOTFINDERW(ROOTARGS, 12)
ROOTARGS(1) = C_A
ROOTARGS(2) = C_B
ROOTARGS(3) = Emax_A
ROOTARGS(4) = EC50_A
ROOTARGS(5) = H_A
ROOTARGS(6) = Emax_B
ROOTARGS(7) = EC50_B
ROOTARGS(8) = H_B
ROOTARGS(9) = Int_AB
ROOTARGS(10) = Int_BA
ROOTARGS(11) = EC50_Int_AB
ROOTARGS(12) = EC50_Int_BA
ROOT = ROOTFINDERW(ROOTARGS)
```

Verbatim function call

A verbatim function call to the fortran interface function is in this case the fastest and most elegant solution as there is no need to store the arguments in a vector before the call.

```
" RESULT = ROOTFINDER(C_A, C_B, Emax_A, HC50_A, H_A, Emax_B, EC50_B, &
"       & H_B, Int_AB, Int_BA, EC50_Int_AB, EC50_Int_BA)
```

Compilation and linking

The abbreviated functions and or interface functions are added to the NM-TRAN control stream by referencing the source file in a SUBROUTINE record:

```
$SUBROUTINE OTHER=rootfinder.f03
```

The C function should be compiled before running NONMEM, for example with gcc.

Linking custom object code to NONMEM has to be done by modifying the nmfe script. The one line calling gfortran to do the linking was modified by adding the rootfinder object code and the needed gsl libraries. It would also be possible to set the u variable in the beginning of the script. The linux version of the changed line follows (the line breaks are not in the original script):

```
$f $o $opstat -o $nmexec  $s $ndir/PNM_MPID.o rootfinder.o -I$tempdir \
-I$dir/resource `cat LINK.LNK`  $dir/resource/PRGLOBAL.o \
$rdir/PRGLOBALP.o $pdir/MUMODEL.o $dir/util/PARALLEL.o $pdir/PRPNM.o \
$n $u $r -lgsl -lgslcblas > gfortran.txt
```

Conclusion

Our C function could be called from an NM-TRAN control stream via an NM-TRAN abbreviated function or verbatim code. The Fortran 2003 iso_c_binding module allows for C functions to be called from Fortran and therefore also NM-TRAN in a portable way. A slightly specialized nmfe script is needed to add the C file and dependencies to the linkage. The possibility to add files for linkage to nmfe without changing the script would simplify the future use of C together with NONMEM.

[1] C. Chen, S. G. Wicha, R. Nordgren, and U. S. H. Simonsson. Comparisons of analysis methods for assessment of pharmacodynamic interactions including design recommendations. *Submitted*, 2017.

[2] S. Loewe. The problem of synergism and antagonism of combined drugs. *Arzneimittelforschung. 1953 Jun;3(6):285–90.*, 1953.

[3] M. Galassi et al. Gnu scientific library reference manual (3rd ed.). 2016.

[4] S. Beal, L. B. Sheiner, A. Boeckmann, and R. J. Bauer. Nonmem user's guide. 2017.

[5] Fortran 2003 iso/iec 1539-1:2004. 2004.