Fortran calling C: Clock drift correction

Stewart A. Levin

ABSTRACT

Seafloor nodal clock drift is a skew that is taken as distributed linearly across the time between GPS synchronizations. When it came time to correct such skew in a nodal field dataset, I wanted to use the well-tested, accurate 8 point sinc resampler in the Colorado Schools of Mines Seismic Unix (SU) package while staying within the existing Fortran 2003 code I was modifying. In this short note I illustrate how I used the BIND features of modern Fortran to accomplish this.

INTRODUCTION

It is currently infeasible to synchronize actively deployed seafloor nodal recorders with GPS signals. As a result, the internal clocks of the nodes can, and almost surely will, drift away from the GPS standard between the time they are deployed and the time they are retrieved. Lacking any other detail on the drift progress, it is assumed to be linear with time.

Correcting for this drift requires some interpolation between samples, which, as with statics corrections, is best deferred to as late as possible in any processing sequence in order to do the least damage to the signal. In most cases, the drift is treated directly as a static. However, during extended recording such as multi-day passive recording, the change in drift between the start and end of such traces should, and can, be taken into account by adjusting the sample interval from, e.g., 2 msec to 2.000004 msec, and then, for convenience, resampling back to the nominal sample interval.

CWP TO THE RESCUE

While a quick and dirty linear interpolation was enough to generate some preliminary displays of moved-out nodal common receiver records this summer, linear interpolation is generally too damaging to higher frequency content than is normally acceptable. (See, however, Levin (2012) for exceptions.) Fortunately, the widely used SU software (`www.cwp.mines.edu/cwpcodes`) contains a high quality[1] 8-coefficient approximate sinc interpolator `shfs8r` written in C by Dave Hale that has a *maximum* error less than one percent for frequencies up to 60% Nyquist.

[1]Admittedly, we caused a segfault the first time we used it! Bug fix sent to CWP.

Much of the software at SEP is written in Fortran. As the reader is probably aware, invoking a C routine from Fortran has traditionally been a system and compiler dependent process, involving special wrappers to translate between naming conventions and argument handling. Recognizing this, modern Fortran standards, starting with the Fortran 2003 standard, provide a BIND language mechanism for calling C routines. This, in effect, allows us to write a "universal wrapper" for the C routine by means of an interface definition. For `shfs8r` this definition is:

```
interface
   subroutine CWPshfs8r(dx,nxin,fxin,yin,yinl,yinr,nxout,fxout,yout) &
      BIND(C,name="shfs8r")
      use iso_c_binding, only: c_int, c_float
      REAL(C_FLOAT), VALUE :: dx
      INTEGER(C_INT), VALUE :: nxin
      REAL(C_FLOAT), VALUE :: fxin
      REAL(C_FLOAT), DIMENSION(nxin) :: yin(:)
      REAL(C_FLOAT), VALUE :: yinl
      REAL(C_FLOAT), VALUE :: yinr
      INTEGER(C_INT), VALUE :: nxout
      REAL(C_FLOAT), VALUE :: fxout
      REAL(C_FLOAT), DIMENSION(nxout) :: yout(:)
   end subroutine CWPshfs8r
end interface
```

where I opted to use a different Fortran name for `shfs8r` just to provide a clue as to the C routine's origin.

REFERENCES

Levin, S. A., 2012, Integral operator quality from low order interpolation: SEP-Report, **147**, 323–332.