Subject: Re: Comparing tabulated functions
Posted by wlandsman on Thu, 03 Jan 2008 20:48:01 GMT
View Forum Message <> Reply to Message

>
> I can write simple, highly efficient C code that does exactly what I
> want; translating that code into IDL would make heavy use of loops,
> and therefore wouldn't be very efficient. I can write simple IDL code
> that takes no advantage of the fact that the arrays are monotonically
> increasing, calculating the difference of every point on the output
> grid from every point on either of the two input tables, and finding
> the minimum, but that seems extremely inefficient (and a memory hog!).
> Is there a simpler way to do this?

A quick answer.

When you do the interpolation onto the comon grid, you probably make
use of VALUE_LOCATE -- for example, the INTERPOL() function calls
VALUE_LOCATE, to find the indicies which map the original time values
into the common grid. You can reuse this index vector to know, for
each element in the oommon grid, which two input values bracket it.

For example, if your original times are
t = [3.23,5.33,5.76,7.88,7.93,10.42]
and your gridded times are
tgrid = [4,5,6,7,8,9,10]

if you want your gridded times to be no more than 1 second from actual
data
ii = value_locate(t,tgrid)
bad = where ((tgrid - t[ii]) gt 1) and ((t[ii+1] - tgrid) gt 1),
nbad)
f1[bad] = !VALUES.F_NAN

You would do a similar thing when interpolating the other function,
and then OR the NaN values. Note that VALUE_LOCATE() requires a
monotonoic vector, and that the most efficient method would be to
modify INTERPOL() or your other interpolating routine so that it
returns the VALUE_LOCATE indicies. (You also might have to worry
about end points.)

--Wayne