Subject: Re: Should IDL throw a warning in this case?
Posted by Paul Van Delst[1] on Wed, 06 Jul 2011 16:36:33 GMT
View Forum Message <> Reply to Message

Paolo raises a good point. Ideally the TOTAL function, being relatively generic, should use a compensated summation
algorithm (e.g. Kahan's is a simple one) that would also allow the caller to sort the data prior to the summation
(preferably via a /SORT keyword or somesuch). Checking the TOTAL documentation doesn't seem to indicate anything special
is done in the summation -- although the impact of summation order on the result is discussed in the "thread_pool" section.

Even though I do it myself, stabilising numerics by simply increasing the floating point precision is a rather lazy
approach.

cheers,

paulv

Paolo wrote:
> Well, this has nothing to do with numbers being to big,
> the total of that array is only about 2.16E9, way below
> the overflow limit for floats (about 1E39 or so).
>
> The problem is the limited precision of floats, so when you
> add 2E9 and a number of order 100 you lose the last few
> digits of precision
>
> print,(2E9+300)-2E9
>   256.000
>
> due to the facts that the floats can only carry about 7-8
> digits worth of information.
>
> Now you correctly pointed out that you can solve the
> problem by using doubles, however this is not very
> satisfactory (after all, you may run into a similar problem
> where even the added precision of doubles is not sufficient).
>
> Alternatively, you could use a different way to compute the total.
>
> I suggest the following algorithm: sort the input array, then
> add elements 1 and 2, 3 and 4, 5 and 6 and so on.
> Then repeat the steps on the summed array and so on.
> The function below performs it. It is of course significantly

> slower, but returns a better value for the total. When done
> on your example array, keeping floats everywhere:
>
> IDL> help,v
> V   FLOAT   = Array[150, 150, 27, 13]
> IDL> print,max(v),min(v)
>   273.150   273.150
> IDL> print,pg_robust_total(v)/n_elements(v)
>   273.150
>
>
>
> ;computes the total of the input array in a slow
> ;but more robust fashion
> FUNCTION pg_robust_total,x
>
> nx=n_elements(x)
>
> ind=sort(x)
>
> xsorted=x[ind]
>
> numberOfSteps=alog(nx)/alog(2)
>
>
> FOR i=0,floor(numberOfSteps)-1 DO BEGIN
>   xsorted=[xsorted[0:*:2],0]+[xsorted[1:*:2],0]
> ENDFOR
>
> total=total(xsorted)
>
> return,total
>
> END
>
>
>
> Ciao,
> Paolo
>
>
> Fabien wrote:
>
> Hi IDLers,
>
> Just a thought about the last 10 minutes I lost understanding why the
> MEAN() function was computing wrong values:
>

> IDL> print, !VERSION
> { x86_64 linux unix linux 7.1.1 Aug 21 2009 64 64}
> IDL> tk = FLTARR(150,150,27,13, /NOZERO)
> IDL> tc = tk - 273.15
> IDL> print, min(tk-tc), max(tk-tc)
> 273.150 273.150
>
> everything OK, until:
>
> IDL> print, mean(tk-tc)
> 267.597
>
> Oh my god, how is this even possible???? Am I getting crazy?
>
> And then, after 5 minutes and a coffee break:
>
> IDL> print, mean(tk-tc, /DOUBLE)
> 273.14999
>
> Uf, thank god I'm not crazy.
>
> My feeling would say: IDL should throw a warning when you are
> manipulating too big numbers (in my case: too big arrays) with IDL
> built-in functions.
>
> However, you IDL experts may not think so. What would be the reasons
> for not throwing a warning? Thanks!
>
> Fabien
