Subject: Re: rounding errors
Posted by Craig Markwardt on Fri, 27 Apr 2001 15:52:17 GMT
View Forum Message <> Reply to Message

"Dominic R. Scales" <Dominic.Scales@aerosensing.de> writes:
> Unfortunately, the input data is a fixed format I get from an external
> source and can not change. I fully appreciate and understand the precision
> difference in significant digits between float and double, be it in idl,
> c, or any other programming language, as it is inherently diffucult
> (read impossible) to map an infinite set of numbers to a finite realm of
> 4/8 byte and not miss any.
> The case I am trying to make is this: why aren't the missing digits, i.e.
> the ones added by the cast, set to 0.? As I go along with my calculations,

Why aren't these digits rounded to zero? Because 2.5698900222778320
happens to be the "exact" representation of 2.56989 in 32-bit binary
floating point. The grid spacing of floating point numbers increments
by 3.06e-7 near a value of 2.56. There is no grid point at 2.56989.
Any number between 2.5698900222778320 +/- (0.5 * 3.06e-7) will
collapse to that number. You can verify this with the following
statements:

```
IDL> ff = 2.5698900222778320 + (findgen(100)-50)*3.06e-7/20
IDL> print, ff, format='(2(F24.20))'
```

Notice how the input values all collapse to a just a few output
values.

> these random additional digits give me a hell of a headache by accumulating.
> And, as Murphy leads us to expect, always in the 'wrong' direction.
> I know, one shouln't test floating point numbers in digital rep. against
> one another, not even if the were double.

If the errors are accumulating signifcantly in your calculations then
you have other problems. Remember, in your original float data the
fractional uncertainty is *always* +/- 1.2e-7. You can never get
around this. If this is making a difference in your output results
then your external source is obviously not providing high enough
precision data.

You can estimate confidence intervals on your output values by
shifting the inputs by +/- 1 bit, and seeing how the results change.

```
eps = (machar()).eps
fcent  = f(data)
fplus  = f(data * (1.+eps) )
fminus = f(data * (1.-eps) )
```

where F is your computational function. If FCENT, FPLUS and FMINUS
are intolerably different then you know that the problem is in the
source of your data, and not in IDL. Of course, there are numerical
techniques that you can apply within your own calculation which may
help avoid cancellation errors. For example, the expression

```
  sqrt(1+x) - 1
```

is woefully inaccurate when X is small. Routines like HYPOT in
Numerical Recipes, which seem on their face to be superfluous, treat
this kind of problem.

Craig

--
 -- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 -- --------------
