Subject: Re: rounding errors
Posted by Dominic R. Scales on Fri, 27 Apr 2001 14:41:58 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
>
> In principle you should not have to worry about those extra random
> digits. Since your original data are floating point, then they
> contain no more than 7 digits of precision. Any digits beyond the 7th
> are simply *undefined*. If you want higher precision then you should
> recreate your file using double precision arithmetic from the start.
>
> Even if you use double precision then you will still have random
> digits at the end, but just farther out:
>
> IDL> print, 2.56989d, format='(D30.20)'
> 2.56989000000000000767
>
> This is a consequence of how numbers are represented in the base-two
> number system. Generally speaking the relative uncertainty will be
> (MACHAR()).EPS for floating point in IDL.
>
> Craig

Randall Skelton wrote:
>
> Perhaps I am confused, but if you want the data to be represented as
> doubles, you should read it directly into a double array
> ('array=dblarr(1000)' or something similar)
>
> While you can legitimately extend the precision of floating point numbers
> to those of doubles you must always remember the underlying IEEE
> definition which states floats only have 6 digits precision while doubles
> have 15 digits of precision. When you recast a float into a double, you
> expect decimal digits 6-15 will be noise because bits beyond the float
> precision can truly be anything. Asking IDL to make a floating point
> number into double precision with 'zero padding' like you suggest is like
> asking IDL to know what decimal digits 6-15 were before you cast them as
> floats. Using strings as an intermediate type does avoid the problem
> you describe but it also shows a genuine misunderstanding of storage
> types.
>
> For the record, I had no idea that IDL requires you to explicitly state
> 'a=2.348339d0' instead of a=double(2.348339).
>
> Randall
>
> PS: If you are still having trouble with this consider a simple C program:

>

Dear Craig, dear Randall
thanks for your answers.

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, 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.

Starting with numbers that are said to have a precision of 15 digits but have random digits after after the 6th!?! Why should I even bother?
BUT: it does work for a cast via double(string))...

Yep, it isn't idl's fault here. It only cost me some time to notice...

well, cu all,
Dominic
--
Dipl. Phys. Dominic R. Scales | Aero-Sensing Radarsysteme GmbH
Tel: +49 (0)8153-90 88 90 | c/o DLR Oberpfaffenhofen
Fax: +49 (0)8153-908 700 | 82234 Wessling, Germany
WWW: aerosensing.de | email: Dominic.Scales@aerosensing.de