Subject: Re: Plot bug or another "strange feature" ?
Posted by Paul van Delst on Tue, 16 May 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Nicolas Decoster wrote:
>
> Paul van Delst wrote:
>>
>> Nicolas Decoster wrote:
>>>
>>> Hi.
>>>
>>> Does anybody notice and, eventually, know how to handle this "strange
>>> feature" of the plot procedure ?
>>>
>>> IDL> s = indgen(1000) + 1000000000
>>> IDL> plot, s, /ynozero
>>>
>>> The plot is not a clean line joining down-left and up-right corners, but
>>> an ugly stairway...
>>
>> I thought that sort of thing came about because:
>>
>> a) PLOT converts all its arguments to single precision floats. Do a
>> PRINT, FLOAT(s[0:100]),FORMAT='(e20.13)' to see what happens to the
>> float'd integers,
>>
>> b) people think they can represent numbers at the extremes of machine
>> precision exactly. Your "s" array, when converted to single precision
>> float, changes in the 8th or 9th d.p. While I think that IDL should
>> allow users to set the PLOT conversion to double precision if they want
>> (e.g. with a DOUBLE keyword or something), in general you can't expect
>> these sorts of numbers to be represented well - in IDL or any language.
>> That's not how floating point arithmetic works. All floating point
>> numbers are approximations to their actual value.
>
> I mainly agree with you. But "s" array is not a double array, it is a
> long array:

It doesn't matter what "s" is originally. When you PLOT it, at some
point the contents are "converted" to *single precision* floating point
numbers.

> I agree with you that plot converts everything to floating point number.
> But I think that it is important to see exactly what are the values of
> my integers (in fact my long integers) in a graph. And it is the same
> with double precision floating point numbers: I am not talking about the
> representation of real world numbers using floating point numbers.

> Example:
>
> IDL> b = [100000000.00000001d, 100000000.00000003d]
> IDL> print, b, format = '(e22.16)'
> 1.0000000000000001e+08
> 1.0000000000000003e+08
> IDL> plot, b, /ynozero
> % PLOT: Data range for axis has zero length.
> % Execution halted at: $MAIN$

Well now I'm confused because your example above deals *explicitly* with the representation (or the problems therein) of real world numbers using floating point arithmetic. By using PLOT, "b" effectively becomes an array of the same two numbers. Personally, I would like IDL's PLOT to "convert" everything to double precision but maybe that raises resource issues when you want to plot x/y arrays of, say, 2^19 point - which I do all the time. What IDL does satisfies all but a miniscule percentage of scenarios. To me that reflects good design philosophy (you know, diminishing returns and all that).

Taking your example above, I would prefer to plot the data (and similar) like so:

IDL> b = [100000000.00000001d, 100000000.00000003d]
IDL> print, b, format = '(e22.16)'
1.0000000000000001e+08
1.0000000000000003e+08
IDL> plot, b-(total(b)/double(n_elements(b))),
/ynozero,ytitle='y-ave(y)'

or even plot the percentage change or something. Anything than the actual magnitudes.

I think I understand what you mean and what you want - my point is (and I'm sure you already know this) that when you want to plot/print/whatever real numbers that reside near the boundaries of the ability of various architectures to represent them, you're going to run into problems as we have seen above. It's not a particularly IDL thing, you just have to be careful.

paulv
--
Paul van Delst Ph: (301) 763-8000 x7274
CIMSS @ NOAA/NCEP Fax: (301) 763-8545
Rm.202, 5200 Auth Rd. Email: pvandelst@ncep.noaa.gov
Camp Springs MD 20746