Subject: Puzzle with floating point underflow
Posted by Martin Schultz on Thu, 23 Aug 2001 07:30:35 GMT
View Forum Message <> Reply to Message

Hi all,

How can a float number be something e-42 if the system says it can only represent numbers down to 1.e-38 in a float????????

```
test= 8.1047657d-42
IDL> tmp=float(test)
% Program caused arithmetic error: Floating underflow
% Detected at  MGS_RGRID::REGRID  203 /pf/m/m218003/home/IDL/lib/mgs_newobjects
  /mgs_rgrid__define.pro
IDL> help,tmp
TMP             FLOAT     =   8.10511e-42

IDL> help,machar(),/stru
** Structure MACHAR, 13 tags, length=52:
   IBETA           LONG                 2
   IT              LONG                24
   IRND            LONG                 5
   NGRD            LONG                 0
   MACHEP          LONG               -23
   NEGEP           LONG               -24
   IEXP            LONG                 8
   MINEXP          LONG              -126
   MAXEXP          LONG               128
   EPS             FLOAT       1.19209e-07
   EPSNEG          FLOAT       5.96046e-08
   XMIN            FLOAT       1.17549e-38    <<<<<<<<<<<
   XMAX            FLOAT       3.40282e+38
```

(For those, who haven't figured this out from the Machar output: I am running IDL5.3 on a Linux system (kernel 2.2.14))

Not a real problem because underflows are considered harmless, but I am curious about this, anyhow.

Cheers,

Martin

--

[[[[[[[[[[
[[Dr. Martin Schultz Max-Planck-Institut fuer Meteorologie [[
[[Bundesstr. 55, 20146 Hamburg [[
[[phone: +49 40 41173-308 [[
[[fax: +49 40 41173-298 [[
[[martin.schultz@dkrz.de [[
[[[[[[[[[[

Subject: Re: Puzzle with floating point underflow
Posted by Craig Markwardt on Thu, 23 Aug 2001 15:21:25 GMT
View Forum Message <> Reply to Message

Martin Schultz <martin.schultz@dkrz.de> writes:

...

```
>     How can a float number be something e-42 if the system says it can only
>  represent numbers down to 1.e-38 in a float????????
>
>  test= 8.1047657d-42
>  IDL> tmp=float(test)
>  % Program caused arithmetic error: Floating underflow
>  % Detected at  MGS_RGRID::REGRID  203 /pf/m/m218003/home/IDL/lib/mgs_newobjects
>    /mgs_rgrid__define.pro
>  IDL> help,tmp
>  TMP             FLOAT     =   8.10511e-42
```

You have a denormalized number! Internally most floating point numbers are normalized, which means that the mantissa and exponent are adjusted so that the leading digit in the mantissa is unity. In a denormalized quantity the mantissa doesn't have that property.

It's better to make an example, in decimal arithmetic. We all know that we can change this value:

```
  0.02500 x 10^{-38}      Denormalized
  Mantissa      Exponent
```

into this value

```
  2.50000 x 10^{-40}       Normalized
  Mantissa      Exponent
```

That is a good way to do it with floating point numbers since there are a fixed number of bits that can be used to represent the mantissa. Normalizing maximizes the number of precision bits available for a given computation.

But what happens when we also only have a fixed number of bits to

represent the *exponent*? Then it may not be possible to represent
10^{-40} since the smallest is 10^{-38}. In that case one has to be
content with the denormalized quantity, like 0.025 x 10^{-38} above.

You don't get something for nothing, though. The trade-off is that
you start to lose precision in the mantissa. The worst case happens
when you have the number 2.5 x 10^{-38} / 10^5, something like this:

  0.00002 x 10^{-38}   Normalized
  Mantissa   Exponent

The "5" in 2.5 just got lost! Because the number of available bits of
precision varies with the magnitude of the number, it is best to avoid
these kinds of situations. :-)

Craig

--

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

Subject: Re: Puzzle with floating point underflow
Posted by Karl Schultz on Thu, 23 Aug 2001 15:29:40 GMT
View Forum Message <> Reply to Message

What is probably happening here is that the floating point system stored the
result in denormalized form in an attempt to do the best thing it could in
this underflow situation. In denormalized form, the number is shifted right
within the mantissa, while still using the maximum magnitude allowed in the
exponent. This is trading off precision for range, but it is always better
to keep the exponent correct, if possible. Note that the precision
maintained in your example is less than the usual 5 or 6 decimal digits.

"Martin Schultz" <martin.schultz@dkrz.de> wrote in message
news:ylwwv3v1bdw.fsf@faxaelven.dkrz.de...
>
> Hi all,
>
>
>   How can a float number be something e-42 if the system says it can only
> represent numbers down to 1.e-38 in a float????????
>
> test= 8.1047657d-42
> IDL> tmp=float(test)
> % Program caused arithmetic error: Floating underflow

> % Detected at MGS_RGRID::REGRID 203
/pf/m/m218003/home/IDL/lib/mgs_newobjects
> /mgs_rgrid__define.pro
> IDL> help,tmp
> TMP FLOAT = 8.10511e-42

Subject: Re: Puzzle with floating point underflow
Posted by Paul van Delst on Fri, 24 Aug 2001 14:45:09 GMT
View Forum Message <> Reply to Message

Martin Schultz wrote:
>
> Paul van Delst <paul.vandelst@noaa.gov> writes:
>
>> Martin Schultz wrote:
>>>
>>> Not a real problem because underflows are considered harmless
>>
>> Aargh! wash your mouth out with soap! :o)
>>
>> Sigh.
>>
>> Paul "!EXCEPT=2" van Delst
>>
>
> Can you recommend a soap brand that doesn't taste too bad?

Well, there's this brewery in Pennsylvannia called Yuengling that makes a pretty good lager.... oh. _Soap_. Sorry. :o)

> In fact, I only quoted from the IDL online help:
>
> "In the vast majority of cases, floating-point underflow errors
> are harmless and can be ignored."

Oh man, what a ridiculous statement to make in a manual. Makes me think the writer(s) never actually thought IDL would be used for anything _really_ important. I hope anyone using IDL to, say, process MRI images to look for anomalies never read that part of the manual. Given tortuous enough code, the same can be said for fp overflow errors. Or fp inexact errors. Divide by zero will probably break things everytime depending on how the result is handled. IDL has a great way of handling these sorts of errors (typically with minimum user intervention) that doesn't translate into other programming languages, and bad habits can be hard to break.
>
> I don't like statements like this either, but !EXCEPT=2 really
> slows down my code too much ;-)

Well, I don't recommend it (!EXCEPT=2) as something one has turned on all the time. But for code testing it's a must.

paulv

--
Paul van Delst A little learning is a dangerous thing;
CIMSS @ NOAA/NCEP Drink deep, or taste not the Pierian spring;
Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
Fax:(301)763-8545 And drinking largely sobers us again.
Alexander Pope.
