Subject: Re: Interesting Filled Contour Problem
Posted by Kenneth P. Bowman on Tue, 17 Apr 2012 16:00:30 GMT
View Forum Message <> Reply to Message

In article < 9436440.509.1334632862216.JavaMail.geo-discussion-forums@vbv d13 >,
Craig Markwardt <craig.markwardt@gmail.com> wrote:

> On Monday, April 16, 2012 4:57:40 PM UTC-4, David Fanning wrote:
>> We have a winner, and from a MOST unexpected source!
>> Josh, a Technical Support Engineer at Excelis, has
>> correctly identified the problem as being caused
>> by NaNs in the original data set. When I do this:
>>
>> data = data > 0
>>
>> I apparently set the NaN values to zero, too.
>> (Not sure this is totally kosher, but it does
>> seem to work!)
>
> Whaaa? I was under the impression that
> NAN = !values.d_nan
> NAN OP X
> will always result in a value of NAN, for all operations OP and for all
> operands X. In other words, I don't think IDL's behavior is kosher.
>
> Strangely,
> NAN < 0
> produces NAN, so apparently NAN is a negative number (!!!).
>
> Craig

IEEE 754 says this

5.11 Details of comparison predicates 5.1.0

For every supported arithmetic format, it shall be possible to compare one
floating-point datum to another in that format (see 5.6.1). Additionally,
floating-point data represented in different formats shall be comparable as long

Four mutually exclusive relations are possible: less than, equal, greater than,
and unordered. The last case arises when at least one operand is NaN. Every NaN
shall compare unordered with everything, including itself. Comparisons shall
ignore the sign of zero (so +0 = ?0). Infinite operands of the same sign shall
compare equal.

Languages define how the result of a comparison shall be delivered, in one of
two ways: either as a relation identifying one of the four relations listed

above, or as a true-false response to a predicate that names the specific comparison desired.

Assuming that the floating-point hardware returns an "unordered" for any comparison involving a NaN, the question is, what does IDL do with it?

For a logical comparison, it seems to properly return false

```
IDL> PRINT, 5.0 EQ !Values.F_NaN
  0
IDL> PRINT, 5.0 LT !Values.F_NaN
  0
IDL> PRINT, 5.0 GT !Values.F_NaN
  0
```

The < and > operators, however, mean "return the lesser or greater of" the two arguments. It seems to me that by definition the result is undefined when the relation is "unordered". That is, it is neither less than, greater than, nor equal, and it looks like IDL is inconsistent in its handling of that case. At least it does generate a floating-point exception.

```
IDL> PRINT, 1.0 < !Values.F_NaN
     1.00000
% Program caused arithmetic error: Floating illegal operand
IDL> PRINT, 1.0 > !Values.F_NaN
        NaN
% Program caused arithmetic error: Floating illegal operand
```

Looks like an implementation issue.

Ken
