Subject: Re: Testing for NODATA presence in a dataset
Posted by thompson on Tue, 24 Dec 2002 15:11:32 GMT
View Forum Message <> Reply to Message

tam@lheapop.gsfc.nasa.gov (Tom McGlynn) writes:

> David Fanning <david@dfanning.com> wrote in message
news:<MPG.186e97123b2778b2989a81@news.frii.com>...
>> Jonathan Greenberg (greenberg@ucdavis.edu) writes:
>>
>>> I'm having a problem testing for whether an entry in an array is NAN --
>>> doing something like:
>>>
>>> If (value EQ !VALUES.F_NAN) then begin
>>> print,'Not a number'
>>> Endif else begin
>>> print,'Is a number!'
>>> Endelse
>>>
>>> Will always return 'Is a number', even if I set:
>>> value = !VALUES.F_NAN
>>>
>>> What's going wrong with this?
>>
>> The problem is that NAN is ... well, not a number.
>> Thus, you can't use it in expressions that
>> require a number. (Think of it as a mathematical
>> Catch-22, if you like.)
>>
>> The proper way to write this code is like this:
>>
>> If Finite(value) EQ 0 then begin
>> print,'Not a number'
>> Endif else begin
>> print,'Is a number!'
>> Endelse

> That doesn't distinguish NaN from the infinities.
> The standard trick in any language for looking for NaN's is

> if x ne x then begin
> print,'This is a NaN'
> endif else ...

> This can get optimized away if the compiler/interpreter
> is poorly designed. Seemed to work for me in a quick
> test though for IDL 5.2 on Linux. NaN's are not equal

> to anything --- even themselves.

> Regards,
> Tom McGlynn

I would like to echo Tom's response, and remind people that there is a whole family of values which are interpreted as NaN. The values returned by !VALUES.F_NAN and !VALUES.D_NAN are just the simplest forms. The "X NE X" syntax will catch them all, and has the additional bonus of working no matter whether the data is single precision, double precision, or even complex.

Bill Thompson
