Subject: Re: How do I prevent underflow errors?
Posted by meron on Wed, 17 Feb 1999 08:00:00 GMT
View Forum Message <> Reply to Message

In article <36CA865E.2381C0E9@uni-c.dk>, Michael Viskum <Michael.Viskum@uni-c.dk> writes:
>
>
> Craig Markwardt wrote:
>
>> Phillip & Suzanne David <pdavid@earthling.net> writes:
>>>
>>> I have a large array of data that I'd like to plot with the contour routine.
>>> However, the dynamic range of the data is very large, with values as large as
>>> 1e36 and as small as 1e-40. I noticed that contour accepts float data, not
>>> double data. This data is outside the range of float data, so it needs to be
>>> scaled for the contour routine. I don't really care to differentiate the
>>> 1e-40 from 0, but would like to be able to handle values up to the 1e36. I
>>> was going to scale the data by the largest value (i.e.,
>>> PlotData=Float(Data/Max(Abs(Data)))). This puts the data in the range of -1.0
>>> to 1.0. This should be fine for Contour, but I get an underflow error when
>>> converting from double data to float data. I understand that the data will
>>> come out with a 0 instead of 1e-76, and don't really care. How do I get IDL
>>> to ignore the underflow and just convert the value?
>>>
>>> Phillip
>>>
>>
>> Greetings,
>>
>> Try the following with check_math(), which is documented in IDL.
>>
>> dummy = check_math(1, 1)
>> .... commands with math errors are placed here ...
>> dummy = check_math(0, 0)
>>
>> Most of your messages will go away (except for maybe the last one).
>>
>> Craig
>>
>> --
>> ---------- ----------
>> Craig B. Markwardt, Ph.D. EMAIL: craigmnet@astrog.physics.wisc.edu
>> Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
>> ---------- ----------
>
> Hi,
>
> Try also to have a look at the !EXCEPT system variable. It can have 3 possible

> values.
> If !EXCEPT=0 then IDL will not report any exceptions.
>
The problem with these approaches is that they eliminate all math error messages, including some you may want to see. What you would want is to eliminate only specific error messages, the "underflow" ones in this case. I've been in touch with RSI on this issue, suggesting an upgrade to check-math. Specifically I suggested adding a keyword variable which'll enable you to specify (through a bit mask or so) which errors you want to check and clear. They promised to look into it but I haven't seen results yet.

Anyway, for the moment I wrote a routine which deals with this issue, se below. As it is calling other routines from my library, one should really copy the library to make it usable. The whole library may be found on cars3.uchicago.edu , it is accessible through anonymous FTP. do a CD to MIDL after logging in, it is all there.

Function FPU_fix, x, no_abs = nab

```
;+
; NAME:
; FPU_FIX
; VERSION:
; 3.0
; PURPOSE:
; Clears Floating Point Underflow errors, setting the offending values to
; zero.
; CATEGORY:
; Programming.
; CALLING SEQUENCE:
; Result = FPU_FIX( X)
; INPUTS:
;    X
; Arbitrary.
; OPTIONAL INPUT PARAMETERS:
; None.
; KEYWORD PARAMETERS:
;    /NO_ABS
; Switch.  If set, uses value instead of absolute value for comparison
; with machine minimum.  For internal use only.
; OUTPUTS:
; If the input is of any numeric type, returns the input, with the
; possible substitution of 0 for all occurences of Floating Point
; Underflow.  A non-numeric input is returned as is.
; OPTIONAL OUTPUT PARAMETERS:
```

```
; None.
; COMMON BLOCKS:
; None.
; SIDE EFFECTS:
; None.
; RESTRICTIONS:
; None.
; PROCEDURE:
; Straightforward.  Uses the system routines CHECK_MATH and MACHAR.  Also
; calls ISNUM and M_ABS from MIDL.
; MODIFICATION HISTORY:
; Created 30-AUG-1998 by Mati Meron.
;-

    on_error, 1
    fpucod = 32
    matherrs = ['Integer divided by zero','Integer overflow','','',$
  'Floating-point divide by zero','Floating-point underflow',$
  'Floating-point overflow','Floating-point operand error']

    chem = check_math()
    if Isnum(x,type = typ) and chem gt 0 then begin
 if chem eq fpucod then begin
     sinf = machar(double = Isnum(x,/double))
     if keyword_set(nab) then dum = where(x lt sinf.xmin, nuf) $
     else dum = where(M_abs(x) lt sinf.xmin, nuf)
     if nuf gt 0 then x(dum) = 0
 endif else message, matherrs(round(alog(chem)/alog(2)))
    endif

    return, x
end
```

Mati Meron | "When you argue with a fool,
meron@cars.uchicago.edu | chances are he is doing just the same"