Subject: Re: Patch to userlib/deriv.pro to add MAX_VALUE keyword
Posted by nowicki on Mon, 27 Dec 1993 10:50:23 GMT
View Forum Message <> Reply to Message

In Article <thompson.757008292@serts.gsfc.nasa.gov>
thompson@serts.gsfc.nasa.gov (William Thompson) writes:
> rfinch@water.ca.gov (Ralph Finch) writes:
>
>> Following context diff, when run through patch, will add the MAX_VALUE
>> keyword to the userlib function deriv.pro. However, I still believe
>> it would be better if IDL had the fundamental notion of missing values
>> built-in to their basic mathematics; IEEE NaN might be a good choice.
>> If IDL understood NaN's, you wouldn't even need keywords for missing
>> values or max_value (presuming that most usage of MAX_VALUE really is
>> for missing values).
>
> However, not all computer platforms use the IEEE standard. A notable exception
> is Digital's VAX platform, which represents a large proportion of IDL's
> users--in fact IDL came out of the VAX/PDP world before it was ported to Unix,
> etc. As far as I know, there's no equivalent concept to NaNs in the VAX
> architecture.
>
> Bill Thompson
>
> P.S. Although Digital's Alpha workstations support both the VAX and IEEE
> floating point standards, my understanding is that IDL uses the VAX floating
> point representation, at least within OpenVMS, for backwards compatibility with
> the VAX/VMS architecture. I'm not sure which is used in OSF/1, although it
> would seem most advantageous to use the IEEE format for compatibility with
> MIPS/Ultrix workstations. I'm also not certain whether there's a performance
> tradeoff between using VAX and IEEE floating point numbers on the Alpha.

I can expand on some of the above. The AXP (Alpha) architecture supports
BOTH VAX (F and G) and IEEE (S and T) real formats. The conversion is done
in PAL code, (the equivalent of microcode on the CPU.) OpenVMS IDL 3.5.0
uses VAX real formats for I/O, I can't state for sure if the internal
representation is VAX (although, I'm pretty sure it is.)

OSF/1 uses IEEE exclusively, but I believe there's a compiler switch which
allows the reading and writing of VAX format in HLL's.

As far as I can tell, the performance tradeoff is nill due to the use of
PAL code routines for conversion.

-Greg

/* Greg Nowicki | Mail Stop 401A | LIDAR Applications Group */
/* NASA Langley Research Center | Hampton, Virginia 23681-0001 */

/* Voice: (804) 864-2713 | FAX: (804) 864-7790 */
/* nowicki@tardis.larc.nasa.gov | My opinions and mine alone . . . */

Subject: Re: Patch to userlib/deriv.pro to add MAX_VALUE keyword
Posted by thompson on Mon, 27 Dec 1993 16:04:52 GMT
View Forum Message <> Reply to Message

rfinch@water.ca.gov (Ralph Finch) writes:

> Following context diff, when run through patch, will add the MAX_VALUE
> keyword to the userlib function deriv.pro. However, I still believe
> it would be better if IDL had the fundamental notion of missing values
> built-in to their basic mathematics; IEEE NaN might be a good choice.
> If IDL understood NaN's, you wouldn't even need keywords for missing
> values or max_value (presuming that most usage of MAX_VALUE really is
> for missing values).

However, not all computer platforms use the IEEE standard. A notable exception is Digital's VAX platform, which represents a large proportion of IDL's users--in fact IDL came out of the VAX/PDP world before it was ported to Unix, etc. As far as I know, there's no equivalent concept to NaNs in the VAX architecture.

Bill Thompson

P.S. Although Digital's Alpha workstations support both the VAX and IEEE floating point standards, my understanding is that IDL uses the VAX floating point representation, at least within OpenVMS, for backwards compatibility with the VAX/VMS architecture. I'm not sure which is used in OSF/1, although it would seem most advantageous to use the IEEE format for compatibility with MIPS/Ultrix workstations. I'm also not certain whether there's a performance tradeoff between using VAX and IEEE floating point numbers on the Alpha.

Subject: Re: Patch to userlib/deriv.pro to add MAX_VALUE keyword
Posted by schmitt on Mon, 27 Dec 1993 20:34:37 GMT
View Forum Message <> Reply to Message

This is somewhat besides the point of the original posting, but it has to do with a (granted, nitpicking) bug in the userlib routine deriv.pro.
You might as well fix it when you make the posted patch.

Basically, here's a use of "float(...)" in deriv.pro that is unneeded, and makes the resulting answer wrong and/or inaccurate if any other type of data (complex, double, ...&c) is passed as the Y value.
The unneeded usage of float occurs about here:

>   d = float(shift(y,-1) - shift(y,1))/(shift(x,-1) - shift(x,1))
          ^^^^^

remove the "float", and the applicability of the routine improves.

Andy Schmitt / Code 6730 / Naval Research Lab / Washington DC 20375
schmitt@this.nrl.navy.mil
usual disclaimers apply

Subject: Re: Patch to userlib/deriv.pro to add MAX_VALUE keyword
Posted by thompson on Mon, 27 Dec 1993 23:23:32 GMT
View Forum Message <> Reply to Message

schmitt@ccfsun.nrl.navy.mil (Andrew Schmitt) writes:

> This is somewhat besides the point of the original posting, but it has to do
> with a (granted, nitpicking) bug in the userlib routine deriv.pro.
> You might as well fix it when you make the posted patch.

> Basically, here's a use of "float(...)" in deriv.pro that is unneeded,
> and makes the resulting answer wrong and/or inaccurate if any other type
> of data (complex, double, ...&c) is passed as the Y value.
> The unneeded usage of float occurs about here:

>>   d = float(shift(y,-1) - shift(y,1))/(shift(x,-1) - shift(x,1))
>   ^^^^^
> remove the "float", and the applicability of the routine improves.

Not having looked at the original source code, I'm not sure if this is the
case, but one can imagine that problems could occur if the call to FLOAT is
removed and SHIFT is one of the integer types. Another way to get around this,
without leading to errors if SHIFT is double-precision or complex, is to insert
a multiplication by 1.0 somewhere, e.g.

  d = (1.0*shift(y,-1) - shift(y,1))/(shift(x,-1) - shift(x,1))

Of course, this does lead to some inefficiencies.

Bill Thompson
