Subject: Interesting Rant
Posted by David Fanning on Tue, 14 Nov 2006 17:40:56 GMT
View Forum Message <> Reply to Message

Folks,

Someone sent me a link to this interesting IDL rant this morning:

http://www.sccs.swarthmore.edu/users/07/sstvinc2/research/st upid.html

This is what happens, I suppose, when the language
you are using is older than you are. Very little
historical perspective. But it may go a little way
towards explaining why people coming out of college
seem to prefer Matlab by a large margin.

I disagree with much of what he has to say, but I
can't tell whether this is because I am old enough
to remember what things were like *before* IDL, or
whether I've just become inured over the years. :-(

Cheers,

David

P.S. Odd, but he didn't mention the Contour plot anywhere!

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Interesting Rant
Posted by Richard Edgar on Wed, 15 Nov 2006 18:31:27 GMT
View Forum Message <> Reply to Message

Earl F. Glynn wrote:

>> Row-major vs column major is a silly point... you just need to know
>> which way the language does it, and that's the end of the matter. And
>> I'd dispute the bit about 'every other language in the history of
>> mankind' too ;-)
>
> Perhaps a bit exaggerated above, but I don't think this is a silly point at
> all, especially when one must switch between programming languages to
> maintain software. Perhaps, I'm wrong, but IDL's arrays seem to be unique,

> and unlike C or FORTRAN.

That's true. And I have been caught on numerous occasions by IDL starting its indices from zero.

> FORTRAN or R: column-major format [nrows, ncolumns], 1-origin

Small point, but Fortran will start indexing from whatever number you tell it. Of course, the default is 1.

> The lack of a standard convention here could easily contribute to
> programming errors, especially if one uses IDL and other programming
> languages. Converting code from other languages could also be problematic
> because of silly mistakes getting subscripts right.

That's true, although since IDL always (?) does bounds-checking, I've usually found my slips get caught pretty fast. That's the only annoyance with this issue I've found. Of course, I have a particular perspective on this.... I'm not doing image processing, and I'm only using IDL to process output from other code, rather than doing significant computation in its own right (hence I consider optimising my IDL programs to be less important, and I generally restrict myself to making sure that something is vectorised). For people who are using IDL to do image processing and number crunching, I can see that the inconsistency would get particularly annoying.

Richard

Subject: Re: Interesting Rant
Posted by news.qwest.net on Wed, 15 Nov 2006 19:36:40 GMT
View Forum Message <> Reply to Message

"David Fanning" <news@dfanning.com> wrote in message news:MPG.1fc50df19601da27989dbe@news.frii.com...

> In the old days, you could count on about 1/3 of

I think you mean 1./3

:)

Subject: Re: Interesting Rant

Posted by gknoke on Wed, 15 Nov 2006 20:13:27 GMT
View Forum Message <> Reply to Message

Correct me if I'm wrong, but it's my understanding that for loops in IDL are so painfully slow because so much effort has been put into making vector operations so mind-numbingly fast, and there's some trade-off to be made there. I think the ranter has overlooked that critical point. As someone who does a lot of image processing, I rarely have occasion to operate on particular array elements anyway. I'm still a relative newcomer to IDL, but my biggest gripe has (and always will be) inefficient memory allocation... it's a pain when you can't hold two large arrays (maybe 1/4 of available memory each) in memory even though IDL has 2GB to play with. The rest is details and idiosyncracies. Every language has its strengths and weaknesses, the trick is to know what language is ideal for your task, or if you're forced to use a particular language, learn to write code that exploits the strengths of that language.

Subject: Re: Interesting Rant
Posted by Foldy Lajos on Wed, 15 Nov 2006 20:38:30 GMT
View Forum Message <> Reply to Message

On Wed, 15 Nov 2006, gknoke wrote:

> I'm still a relative newcomer to IDL, but my biggest gripe has (and
> always will be) inefficient memory allocation... it's a pain when you
> can't hold two large arrays (maybe 1/4 of available memory each) in
> memory even though IDL has 2GB to play with.

IDL is innocent here. The allocation is done by the underlying operating system. I guess you are on windows :-)

On my linux box, with 1 GByte memory:

```
IDL> a1=bytarr(1000,1000,500)
IDL> a2=bytarr(1000,1000,500)
IDL> help, /mem
heap memory used: 1000577500, max: 1000577516, gets:  423, frees:  115
```

You can read about memory allocation on windows here:

http://www.ittvis.com/services/techtip.asp?ttid=3346

regards,
lajos

Subject: Re: Interesting Rant
Posted by Mike[2] on Wed, 15 Nov 2006 21:21:14 GMT
View Forum Message <> Reply to Message

Francis Burton wrote:
> In MATLAB, both 7/2 and 7/2. evaluate to 3.5 (displayed as
> 3.5000 by default).

Python behaves like IDL:
>>> 7/2
3
>>> 7/2.
3.5
>>>

This is one of the joys of dynamic typing...

Mike

Subject: Re: Interesting Rant
Posted by James Kuyper on Wed, 15 Nov 2006 23:55:56 GMT
View Forum Message <> Reply to Message

gknoke wrote:
> Correct me if I'm wrong, but it's my understanding that for loops in
> IDL are so painfully slow because so much effort has been put into
> making vector operations so mind-numbingly fast, and there's some

My understanding, which may also be incorrect, is that you've got cause and effect reversed. Loop operations are inherently slow because IDL is fundamentally an interpreted language, so the loop body has to be re-interpreted during each pass through the loop, just in case something has happened that would change the interpretation of the code.
Because loop operations are so slow, there's been a strong incentive to provide ways of avoiding them, by making vector operations fast.

Subject: Re: Interesting Rant
Posted by Nigel Wade on Thu, 16 Nov 2006 09:51:50 GMT
View Forum Message <> Reply to Message

Francis Burton wrote:

> In article <1163605646.246847.326220@i42g2000cwa.googlegroups.com>,
> Braedley <mike.braedley@gmail.com> wrote:
>> His comments about the 7/2 by themselves tell me that he doesn't know

>> jack about programming.

It's not just that example which show his general ignorance. Apparently he's a CS major who has never bothered to learn anything about actually programming and using computers.

>> I know for sure that at least 2 of the 4
>> competing languages (the C family, Java and possibly Maltab) all do the
>> exact same thing as IDL. EXACT! (Okay, maybe Matlab doesn't, I
>> haven't used it in 8 months.) The fourth is Maple, and it may still do

Every language which provides integer arithmetic does this. It's a fact of life of integer arithmetic. That the author of that rant doesn't know this only demonstrated his ignorance, it does not show a fault in any of those languages. The author would probably be equally surprised and annoyed by the loss of precision in floating point and have a rant at other languages because they can't do simple arithmetic correctly.

>
> In MATLAB, both 7/2 and 7/2. evaluate to 3.5 (displayed as
> 3.5000 by default).

MATLAB defaults to using double precision for all variables. As a corollary to the IDL "problem" of 7/2, try the same integer calculation in MATLAB to see how useful the opposite camp can be:

>> int16(7)/int16(2)

That one rather annoyed me when MATLAB first introduced non-double matrices. After I spent several days reprogramming a MATLAB to C interface so it returned integer matrices for integer data I discovered just how comprehensive their support of non-double data types was.

For those of you who don't have access to MATLAB, the result of the MATLAB integer division is:

??? Error using ==> /
Function '/' is not defined for values of class 'int16'.

I wonder how much that would annoy our CS major ranter?

--
Nigel Wade, System Administrator, Space Plasma Physics Group,
University of Leicester, Leicester, LE1 7RH, UK
E-mail : nmw@ion.le.ac.uk
Phone : +44 (0)116 2523548, Fax : +44 (0)116 2523555

Subject: Re: Interesting Rant
Posted by greg michael on Thu, 16 Nov 2006 17:35:42 GMT
View Forum Message <> Reply to Message

> Because loop operations are so slow, there's been a strong incentive to
> provide ways of avoiding them, by making vector operations fast.

I'm sure this is still backwards - this is how I see it:
- vector operations are needed for their power of expression
- IDL exists for this reason, and incidentally, makes them fast
- loops turn out to be slow because IDL is interpreted
- this is of secondary importance, because they're rarely necessary in IDL

regards,
Greg

Subject: Re: Interesting Rant
Posted by Steve Eddins on Fri, 17 Nov 2006 01:00:48 GMT
View Forum Message <> Reply to Message

Nigel Wade wrote:
[snip]
>> In MATLAB, both 7/2 and 7/2. evaluate to 3.5 (displayed as
>> 3.5000 by default).
>
> MATLAB defaults to using double precision for all variables. As a corollary to
> the IDL "problem" of 7/2, try the same integer calculation in MATLAB to see how
> useful the opposite camp can be:
>
>>> int16(7)/int16(2)
>
> That one rather annoyed me when MATLAB first introduced non-double matrices.
> After I spent several days reprogramming a MATLAB to C interface so it returned
> integer matrices for integer data I discovered just how comprehensive their
> support of non-double data types was.
>
>
> For those of you who don't have access to MATLAB, the result of the MATLAB
> integer division is:
>
> ??? Error using ==> /
> Function '/' is not defined for values of class 'int16'.
>
> I wonder how much that would annoy our CS major ranter?

Those who have access to MATLAB 7.0 (June 2004) or later would get this:

```
 >> int16(7) / int16(2)
ans =
      4
```

(MATLAB rounds the fractional part instead of truncating it.)

--
Steve Eddins
http://blogs.mathworks.com/steve

Subject: Re: Interesting Rant
Posted by Nigel Wade on Fri, 17 Nov 2006 13:00:58 GMT
View Forum Message <> Reply to Message

Steve Eddins wrote:

>
> Those who have access to MATLAB 7.0 (June 2004) or later would get this:
>
>>> int16(7) / int16(2)
> ans =
>   4
>
> (MATLAB rounds the fractional part instead of truncating it.)
>

We don't pay maintenance for MATLAB any more :-)

--
Nigel Wade, System Administrator, Space Plasma Physics Group,
          University of Leicester, Leicester, LE1 7RH, UK
E-mail :   nmw@ion.le.ac.uk
Phone :   +44 (0)116 2523548, Fax : +44 (0)116 2523555

Subject: Re: Interesting Rant
Posted by JD Smith on Fri, 17 Nov 2006 19:44:56 GMT
View Forum Message <> Reply to Message

On Thu, 16 Nov 2006 09:35:42 -0800, greg michael wrote:

>
>> [quoted text muted]
>
> I'm sure this is still backwards - this is how I see it:

> - vector operations are needed for their power of expression
> - IDL exists for this reason, and incidentally, makes them fast
> - loops turn out to be slow because IDL is interpreted
> - this is of secondary importance, because they're rarely necessary in

I'd love to see an overhead budget for a single trip around the IDL interpreter loop. There are algorithms which no amount of cleverness can recast into vector operations. For these, you can either code as a DLM, or eat the horrible loop overhead. I've long argued for a "side loop" capability of the language that would greatly reduce the per-iteration overhead, at the cost of skipping processing of keyboard input, widget events, etc., etc. Another (likely better) option would be a more coherent interface to C code, i.e. make DLM writing more akin to assembly writing, having it auto-compile, etc.

JD
