Subject: Re: strange behaviour of bytscl by large arrays
Posted by David Fanning on Mon, 23 Apr 2012 14:21:15 GMT
View Forum Message <> Reply to Message

Klemen writes:

```
>
>  Hi folks,
>
>  is there any explanation of why I don't get the same or at least similar results using the code below by:
>  a) using DINDGEN in line 3
>  b) using FINDGEN in line 3
>
>  pro test
>    s = 10000
>    a = sin(findgen(s, s)/100000.)
>    b = bytscl(a)
>    write_tiff, 'b.tif', b
>  end
>
>  The tif file I get using the DINDGEN function has waves all over the image. The option using FINDGEN produces strange results (a couple of waves and then wide bands of constant values). See the following link for the (resized) results.
>   https://picasaweb.google.com/112572300011512591455/Eumetsat# 5734593216558178098
>
>  I came across this problem as I tried to scale (using HIST_EQUAL and BYTSCL functions) 16-bit 5-band RapidEye data to 24-bit RGB image. Scaling the whole image produced results that were all black, smaller subsets seemed ok.
>
>  Does anybody have a suggestion how to handle this issue?
```

Do you mean other than re-read the Sky Is Falling article again?

http://www.idlcoyote.com/math_tips/sky_is_falling.html

Here are some companion articles that might shed light on this problem:

http://www.idlcoyote.com/math_tips/storenum.html
http://www.idlcoyote.com/math_tips/double.html

I think you should examine the minimum and maximum values in your BytScl
function to see what they are. :-)

Cheers,

David

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

Subject: Re: strange behaviour of bytscl by large arrays
Posted by Klemen on Mon, 23 Apr 2012 14:59:22 GMT
View Forum Message <> Reply to Message

Hi David,

I read "the sky is falling" some time ago. I guess I understand how the numbers are saved in the computer. And perhaps this is not the best case I have shown over here. But still, as far as I understand float type, a variable has first 7 figures that should be ok. But I don't really care about the precision on the 7th figure - i have problems on the 1st figure. Look at this example.

```
IDL> a=findgen(10000,10000)
IDL> print, min(a), max(a)
    0.000000 7.50000e+007              ;a huge error
IDL> h=histogram(a, BINSIZE=10.^5)
IDL> print, h
```

Not just that the maximum value is way too small. The histogram seems strange too. I don't show here the wohle histogram output. As expected the h contains in the begining 100 000 values per bin. But this is just for the first 167 bins. in 168 there are 8 300 000 values, then following about 80 bins contain no values at all, then comes one with 25 000 000, then zeros again... 25 000 000.

So this is not the precision. Is there a point that I am missing, perhaps by the mantisa of a number? If I use dindgen at the begining, everything seems ok. Or is the sky falling anyway? :)

Cheers, Klemen

Subject: Re: strange behaviour of bytscl by large arrays
Posted by David Fanning on Mon, 23 Apr 2012 15:10:40 GMT
View Forum Message <> Reply to Message

Klemen writes:

> I read "the sky is falling" some time ago. I guess I understand how the numbers are saved in the computer. And perhaps this is not the best case I have shown over here. But still, as far as I understand float type, a variable has first 7 figures that should be ok. But I don't really care about the precision on the 7th figure - i have problems on the 1st figure. Look at this example.
>

```
>  IDL> a=findgen(10000,10000)
>  IDL> print, min(a), max(a)
>       0.000000 7.50000e+007              ;a huge error
>  IDL> h=histogram(a, BINSIZE=10.^5)
>  IDL> print, h
>
```
> Not just that the maximum value is way too small. The histogram seems strange too. I don't show here the wohle histogram output. As expected the h contains in the begining 100 000 values per bin. But this is just for the first 167 bins. in 168 there are 8 300 000 values, then following about 80 bins contain no values at all, then comes one with 25 000 000, then zeros again... 25 000 000.
>
> So this is not the precision. Is there a point that I am missing, perhaps by the mantisa of a number? If I use dindgen at the begining, everything seems ok. Or is the sky falling anyway? :)

I just think you are dealing with numbers that don't
have enough significant digits to do what you want to
do. You are going to have to multiply (or divide, as
the case may be) by a number that restores enough
significant values for you to perform your operation.
Or, alternatively, use double precision arrays on
the numbers you do have.

Cheers,

David

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

Subject: Re: strange behaviour of bytscl by large arrays
Posted by Klemen on Mon, 23 Apr 2012 15:22:03 GMT
View Forum Message <> Reply to Message

I am still not sure that I don't have enough significant digits. I have 16 bit unsigned integer data (nothing larger than 10 000) and converting this to byte works fine on a small array but not on a large array.

I'll try your first option anyway. I'll let you know how it works.

Klemen

Subject: Re: strange behaviour of bytscl by large arrays
Posted by David Fanning on Mon, 23 Apr 2012 15:32:36 GMT
View Forum Message <> Reply to Message

Klemen writes:

> I am still not sure that I don't have enough significant digits. I have 16 bit unsigned integer data (nothing larger than 10 000) and converting this to byte works fine on a small array but not on a large array.

Now, wait a minute. This is a different problem then the
examples you have been giving us.

Are you saying now that the *size* of the IntArr
makes a difference in how the data is scaled?

I find that even harder to believe than the magic
of floating point numbers on a computer! ;-)

Cheers,

David

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

Subject: Re: strange behaviour of bytscl by large arrays
Posted by Lajos Foldy on Mon, 23 Apr 2012 16:14:21 GMT
View Forum Message <> Reply to Message

On Monday, April 23, 2012 4:07:00 PM UTC+2, Klemen wrote:
> Hi folks,
>
> is there any explanation of why I don't get the same or at least similar results using the code below by:
> a) using DINDGEN in line 3
> b) using FINDGEN in line 3
>
> pro test
>   s = 10000
>   a = sin(findgen(s, s)/100000.)
>   b = bytscl(a)
>   write_tiff, 'b.tif', b

> end
>
> The tif file I get using the DINDGEN function has waves all over the image. The option using FINDGEN produces strange results (a couple of waves and then wide bands of constant values). See the following link for the (resized) results.
> https://picasaweb.google.com/112572300011512591455/Eumetsat# 5734593216558178098
>
> I came across this problem as I tried to scale (using HIST_EQUAL and BYTSCL functions) 16-bit 5-band RapidEye data to 24-bit RGB image. Scaling the whole image produced results that were all black, smaller subsets seemed ok.
>
> Does anybody have a suggestion how to handle this issue?
>
> Cheers, Klemen

I think IDL's FINDGEN() implementation is wrong: it uses a float counter instead of an integer one. The following test shows the difference:

```
pro test
cpu, tpool_nthreads=1
n=10l^8
nn=n-1
a1=findgen(n)                ; real FINDGEN()
a2=fltarr(n)
count=0.0
for j=0l, nn do a2[j]=count++   ; IDL's implementation
a3=fltarr(n)
count=0ll
for j=0l, nn do a3[j]=count++   ; better implementation
print, a1[nn], a2[nn], a3[nn], format='(3F15.3)'
end
```

(Multithreading must be disabled because the starting values for the threads are calculated as an integer. So the result of FINDGEN() depends on the number of your CPU cores, too :-)

regards,
Lajos

Subject: Re: strange behaviour of bytscl by large arrays
Posted by Lajos Foldy on Mon, 23 Apr 2012 16:18:59 GMT
View Forum Message <> Reply to Message

On Monday, April 23, 2012 6:14:21 PM UTC+2, fawltyl...@gmail.com wrote:
> On Monday, April 23, 2012 4:07:00 PM UTC+2, Klemen wrote:
>> Hi folks,
>>
>> is there any explanation of why I don't get the same or at least similar results using the code

below by:
>> a) using DINDGEN in line 3
>> b) using FINDGEN in line 3
>>
>> pro test
>>   s = 10000
>>   a = sin(findgen(s, s)/100000.)
>>   b = bytscl(a)
>>   write_tiff, 'b.tif', b
>> end
>>
>> The tif file I get using the DINDGEN function has waves all over the image. The option using FINDGEN produces strange results (a couple of waves and then wide bands of constant values). See the following link for the (resized) results.
>> https://picasaweb.google.com/112572300011512591455/Eumetsat# 5734593216558178098
>>
>> I came across this problem as I tried to scale (using HIST_EQUAL and BYTSCL functions) 16-bit 5-band RapidEye data to 24-bit RGB image. Scaling the whole image produced results that were all black, smaller subsets seemed ok.
>>
>> Does anybody have a suggestion how to handle this issue?
>>
>> Cheers, Klemen
>
> I think IDL's FINDGEN() implementation is wrong: it uses a float counter instead of an integer one. The following test shows the difference:
>
> pro test
> cpu, tpool_nthreads=1
> n=10l^8
> nn=n-1
> a1=findgen(n)   ; real FINDGEN()
> a2=fltarr(n)
> count=0.0
> for j=0l, nn do a2[j]=count++   ; IDL's implementation
> a3=fltarr(n)
> count=0ll
> for j=0l, nn do a3[j]=count++   ; better implementation
> print, a1[nn], a2[nn], a3[nn], format='(3F15.3)'
> end
>
> (Multithreading must be disabled because the starting values for the threads are calculated as an integer. So the result of FINDGEN() depends on the number of your CPU cores, too :-)
>
> regards,
> Lajos

Oops! The results were missing:

```
IDL> test
  16777216.000   16777216.000  100000000.000
```

regards,
Lajos

Subject: Re: strange behaviour of bytscl by large arrays
Posted by David Fanning on Mon, 23 Apr 2012 16:50:57 GMT
View Forum Message <> Reply to Message

fawltylanguage@gmail.com writes:

```
> I think IDL's FINDGEN() implementation is wrong: it uses a float counter instead of an integer one. The following test shows the difference:
>
> pro test
> cpu, tpool_nthreads=1
> n=10l^8
> nn=n-1
> a1=findgen(n)                ; real FINDGEN()
> a2=fltarr(n)
> count=0.0
> for j=0l, nn do a2[j]=count++   ; IDL's implementation
> a3=fltarr(n)
> count=0ll
> for j=0l, nn do a3[j]=count++   ; better implementation
> print, a1[nn], a2[nn], a3[nn], format='(3F15.3)'
> end
>
> (Multithreading must be disabled because the starting values for the threads are calculated as an integer. So the result of FINDGEN() depends on the number of your CPU cores, too :-)
```

Wow! The things you learn about IDL when you look into it, huh? ;-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/

Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: strange behaviour of bytscl by large arrays
Posted by Lajos Foldy on Mon, 23 Apr 2012 16:57:54 GMT
View Forum Message <> Reply to Message

On Monday, April 23, 2012 6:50:57 PM UTC+2, David Fanning wrote:

> Wow! The things you learn about IDL when you look into it, huh? ;-)

One learns from the others' mistakes :-)

regards,
Lajos

Subject: Re: strange behaviour of bytscl by large arrays
Posted by David Fanning on Mon, 23 Apr 2012 17:06:49 GMT
View Forum Message <> Reply to Message

fawltylanguage@gmail.com writes:

> One learns from the others' mistakes :-)

Yes, I understand. Now comes the (usually futile)
process of convincing the "other" that the
mistake is theirs and not yours. My rule of thumb
is this takes about five times as long (if it can
be done!) then discovering and correcting the mistake
yourself. ;-)

Cheers,

David

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

Subject: Re: strange behaviour of bytscl by large arrays
Posted by Klemen on Mon, 23 Apr 2012 19:00:45 GMT

View Forum Message <> Reply to Message

Lajos, thank you, this is the same thing as I observed, but it's nice to know that somebody else hase similar experience. You mentioned multhreading - is it possible, that the example I showed (it looks like having 4 areas - waves first and then 3 more constant areas) are due to my processor having two cores (four threads)?

David, sorry that I wasn't so clear in my first post. Yes my problem is, that I have 5 × 7523 × 11727 pixel large dataset. All the values are long integers, relativelly small (not more than 10 000). So the values are not the problem. But coverting my array to byte made me really question what the hell is going on. Conversion worked well on the subset but on the whole image not. The same is converting large float arrays to byte.

So can it be that some IDL functions don't work proper because they include FINDGEN in their code? Or some similar bug? I have today observed strange perfomance on large arrays using BYTE, BYTSCL, HIST_EQUAL, and also the belowed HISTOGRAM. Working on the doubble precision data seems not to be affected.

Cheers, Klemen

Subject: Re: strange behaviour of bytscl by large arrays
Posted by chris_torrence@NOSPAM on Mon, 23 Apr 2012 20:22:08 GMT
View Forum Message <> Reply to Message

On Monday, April 23, 2012 10:14:21 AM UTC-6, fawltyl...@gmail.com wrote:
>
> I think IDL's FINDGEN() implementation is wrong: it uses a float counter instead of an integer one. The following test shows the difference:
>
```
> pro test
> cpu, tpool_nthreads=1
> n=10l^8
> nn=n-1
> a1=findgen(n)              ; real FINDGEN()
> a2=fltarr(n)
> count=0.0
> for j=0l, nn do a2[j]=count++   ; IDL's implementation
> a3=fltarr(n)
> count=0ll
> for j=0l, nn do a3[j]=count++   ; better implementation
> print, a1[nn], a2[nn], a3[nn], format='(3F15.3)'
> end
```
>
> (Multithreading must be disabled because the starting values for the threads are calculated as an integer. So the result of FINDGEN() depends on the number of your CPU cores, too :-)
>
> regards,

> Lajos

Well, wrong is perhaps too strong of a word. The real word is "fast". I just did a test where I changed the internal implementation of FINDGEN to use an integer counter. The "float" counter is 4 times faster than using an integer counter and converting it to floats.

However, perhaps we could look at the size of the input array, and switch to using the slower integer counter if it was absolutely necessary. I'll give it a thought.

Thanks for reporting this!

Cheers,
Chris
Exelis VIS

Subject: Re: strange behaviour of bytscl by large arrays
Posted by Klemen on Mon, 23 Apr 2012 20:54:28 GMT
View Forum Message <> Reply to Message

Hmm today I don't have just the usual problems with IDL, but also with thist post... I would assume that everything is online once you press post... Anyway, short version once again.

Lajos, thank you very much. It's nice to know that I am not the only one seeing a miracle. :) Although the sky is not falling. :) And what I find really interesting is that you observed also the correlation to the number of CPU.

David, sorry that I was not clear in the first post. Yes the thing is that using BYTSCL and HIST_EQAL have some difficulties when processing large arrays. Using the above example of Lajos, see the output of the histogram.

```
pro test
cpu, tpool_nthreads=1
n=10l^8
nn=n-1
a3=fltarr(n)
count=0ll
for j=0l, nn do a3[j]=count++   ; better implementation
h = histogram(a3, binsize=10L^6)
print, h
b = byte(a3)
h = histogram(b)
print, h
end
```

IDL prints:

```
1000000     1000000     1000000     1000000     1000000     1000000     1000000
     1000000     1000000     1000000     1000000     1000000     1000000     1000000
     1000000     1000000      999999     1000000     1000000     1000000     1000000
     1000000     1000000     1000000     1000000     1000000     1000000     1000000
     1000000     1000000     1000000     1000000     1000000      999999     1000000
     1000000     1000000     1000000     1000000     1000000     1000000     1000000
     1000000     1000000     1000000     1000000     1000000     1000000     1000000
     1000000     1000000     1000000     1000000     1000000     1000000     1000000
     1000000     1000000     1000000     1000000     1000000     1000000     1000000
     1000000     1000000     1000000     1000000      999998     1000000     1000000
     1000000     1000000     1000000     1000000     1000000     1000000     1000000
     1000000     1000000     1000000     1000000     1000000     1000000     1000000
     1000000     1000000     1000000     1000000     1000000     1000000     1000000
     1000000     1000000     1000000     1000000     1000000     1000000     1000000
     1000000     1000000           4
     2073833       65536      131072       65536      655360       65536      131072
       65536     1816871       65536      131072       65536      655360       65536
      131072       65536     2073833       65536      131072       65536      655360
       65536      131072       65536     1816871       65536      131072       65536
      655360       65536      131072       65536     2073833       65536      131072
       65536      655360       65536      131072       65536     1816871       65536
      131072       65536      655360       65536      131072       65536     2073833
       65536      131072       65536      655360       65536      131072       65536
     1816871       65536      131072       65536      655360       65536      131072
       65536     2073833       65536      131072       65536      655360       65536
      131072       65536     1816871       65536      131072       65536      655360
       65536      131072       65536     2073833       65536      131072       65536
      655360       65536      131072       65536     1816871       65536      131072
       65536      655360       65536      131072       65536     2073833       65536
      131072       65536      655360       65536      131072       65536     1816871
       65536      131072       65536      655360       65536      131072       65536
     2073833       65536      131072       65536      655360       65536      131072
       65536     1816871       65536      131072       65536      655360       65536
      131072       65536     2073833       65536      131072       65536      655360
       65536      131072       65536     1816871       65536      131072       65536
      655360       65536      131072       65536     2073833       65536      131072
       65536      655360       65536      131072       65536     1816871       65536
      131072       65536      655360       65536      131072       65536     2073833
       65536      131072       65536      655360       65536      131072       65536
     1816871       65536      131072       65536      655360       65536      131072
       65536     2073833       65536      131072       65536      655360       65536
      131072       65536     1816871       65536      131072       65536      655360
       65536      131072       65536     2073833       65536      131072       65536
      655360       65536      131072       65536     1816871       65536      131072
       65536      655360       65536      131072       65536     2073833       65536
      131072       65536      655360       65536      131072       65536     1816871
       65536      131072       65536      655360       65536      131072       65536
     2073833       65536      131072       65536      655360       65536      131072
```

```
   65536    1816871      65536     131072      65536     655360      65536
  131072      65536    2073833      65536     131072      65536     655360
   65536     131072      65536    1816871      65536     131072      65536
  655360      65536     131072      65536
```

Isn't that strange - after scaling to BYTE each histogram bin should contain the same number of values, just like the original histogram. This is not a prblem if I set n no 10^7.

Cheers, Klemen

Subject: Re: strange behaviour of bytscl by large arrays
Posted by Carsten Lechte on Tue, 24 Apr 2012 07:59:04 GMT
View Forum Message <> Reply to Message

On 23/04/12 16:59, Klemen wrote:
> So this is not the precision.[...]

It totally is:

```
IDL> a=findgen(10000,10000)
IDL> plot, a

IDL> print, a[20000000l], a[20000000l] EQ a[20000000]+1
   1.67772e+07   1
```

Since I tried it on a 2-cpu machine, the second half starts with the
correct value, which then stays the same for the whole second part of
the array:

```
IDL> print, a[60000000l], a[60000000l] EQ a[60000000]+1
   5.00000e+07   1
```

chl