Subject: Re: Comparing 2 arrays
Posted by David Fanning on Sun, 26 Aug 2007 04:45:31 GMT
View Forum Message <> Reply to Message

teich@atmsci.msrc.sunysb.edu writes:

> I would like to know if the data in two different 1 dimensional arrays
> are the same or not. This should be for every index. For example, if
> A=[0,1,2,3] and B=[0,1,3,2], then I would want the answer to be that A
> and B are not equal even though they happen to have the same values in
> different order. A problem with comparing each element is that the
> numbers are floating point so I would prefer not to compare with
> 'EQ'. Has anyone any suggestions?
>
> Alternatively, my arrays are actually from different structures. Is
> there a way to compare structures? For example, data1.A vs data2.B in
> the above example.

If your arrays are the same size, there is no problem using structures.
You could subtract the two values and call the result "equal" if
the difference was "sufficiently" close to zero.

```
IF Total( (data1.A - data2.B) LT 1e-6) EQ N_Elements(data1.A) THEN $
   Print, 'The arrays are equal' ELSE Print, 'The arrays are unequal'
```

Cheers,

David
--
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: Comparing 2 arrays
Posted by David Fanning on Sun, 26 Aug 2007 04:47:51 GMT
View Forum Message <> Reply to Message

David Fanning writes:

> If your arrays are the same size, there is no problem using structures.
> You could subtract the two values and call the result "equal" if
> the difference was "sufficiently" close to zero.
>
> IF Total((data1.A - data2.B) LT 1e-6) EQ N_Elements(data1.A) THEN $
>   Print, 'The arrays are equal' ELSE Print, 'The arrays are unequal'

Well, you probably need to check whether the absolute
value of the difference (ABS) is sufficiently close to
zero. But other than that, I think this should work.

Cheers,

David
--
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: Comparing 2 arrays
Posted by teich on Sun, 26 Aug 2007 05:25:53 GMT
View Forum Message <> Reply to Message

On Aug 26, 12:47 am, David Fanning <n...@dfanning.com> wrote:
> David Fanning writes:
>> If your arrays are the same size, there is no problem using structures.
>> You could subtract the two values and call the result "equal" if
>> the difference was "sufficiently" close to zero.
>
>> IF Total((data1.A - data2.B) LT 1e-6) EQ N_Elements(data1.A) THEN $
>>   Print, 'The arrays are equal' ELSE Print, 'The arrays are unequal'
>
> Well, you probably need to check whether the absolute
> value of the difference (ABS) is sufficiently close to
> zero. But other than that, I think this should work.
>
> Cheers,
>
> David
> --
> 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.")

Thanks David for showing me the clever way of doing this!

Howard

Subject: Re: Comparing 2 arrays

Posted by airy.jiang on Sun, 26 Aug 2007 09:39:53 GMT
View Forum Message <> Reply to Message

On 8 26 ,   1 25 , te...@atmsci.msrc.sunysb.edu wrote:
> On Aug 26, 12:47 am, David Fanning <n...@dfanning.com> wrote:
>
>
>
>
>
>> David Fanning writes:
>>> If your arrays are the same size, there is no problem using structures.
>>> You could subtract the two values and call the result "equal" if
>>> the difference was "sufficiently" close to zero.
>
>>> IF Total((data1.A - data2.B) LT 1e-6) EQ N_Elements(data1.A) THEN $
>>>   Print, 'The arrays are equal' ELSE Print, 'The arrays are unequal'
>
>> Well, you probably need to check whether the absolute
>> value of the difference (ABS) is sufficiently close to
>> zero. But other than that, I think this should work.
>
>> Cheers,
>
>> David
>> --
>> 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.")
>
> Thanks David for showing me the clever way of doing this!
>
> Howard-   -
>
> -   -

I suggest use the IDL function Array_Equal,that's a direct way.

Subject: Re: Comparing 2 arrays
Posted by David Fanning on Sun, 26 Aug 2007 14:32:55 GMT
View Forum Message <> Reply to Message

airy.jiang@gmail.com writes:

> I suggest use the IDL function Array_Equal,that's a direct way.

Well, there is no discussion of "sufficiently close" in
the on-line help, so I think I would be VERY careful with
FLOATS. Personally, I wouldn't trust it.

Cheers,

David
--
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: Comparing 2 arrays
Posted by Jean H. on Sun, 26 Aug 2007 15:40:44 GMT
View Forum Message <> Reply to Message

```
>  If your arrays are the same size, there is no problem using structures.
>  You could subtract the two values and call the result "equal" if
>  the difference was "sufficiently" close to zero.
>
>  IF Total( (data1.A - data2.B) LT 1e-6) EQ N_Elements(data1.A) THEN $
>     Print, 'The arrays are equal' ELSE Print, 'The arrays are unequal'
>
>  Cheers,
>
>  David
```

David,

to get back to a previous discussion we had a few month ago about being
"sufficiently close to zero", shouldn't it be (data1.A - data2.B) LT
epsilon * data1.A , with epsilon=(machar()).eps?

Jean

Subject: Re: Comparing 2 arrays
Posted by David Fanning on Sun, 26 Aug 2007 16:43:46 GMT
View Forum Message <> Reply to Message

Jean H. writes:

> to get back to a previous discussion we had a few month ago about being
> "sufficiently close to zero", shouldn't it be (data1.A - data2.B) LT
> epsilon * data1.A , with epsilon=(machar()).eps?

Humm, I don't recall that discussion. But I can see how
this number might meet the criteria of "sufficiently close".
On the other hand, I can also envision situations where
the number could be orders of magnitude larger and still
work for a particular application. I'm probably mistaken,
but it seems to me "sufficiently close" is an arbitrary
value that must be picked empirically to match the data
and what you are trying to do with it.

Cheers,

David

P.S. I'm just thinking that "sufficiently close" to a
black hole, for example, might be a completely different
number than "sufficiently close" to my house.

--
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: Comparing 2 arrays
Posted by Conor on Mon, 27 Aug 2007 13:48:13 GMT

View Forum Message <> Reply to Message

On Aug 26, 12:43 pm, David Fanning <n...@dfanning.com> wrote:
> Jean H. writes:
>> to get back to a previous discussion we had a few month ago about being
>> "sufficiently close to zero", shouldn't it be (data1.A - data2.B) LT
>> epsilon * data1.A , with epsilon=(machar()).eps?
>
> Humm, I don't recall that discussion. But I can see how
> this number might meet the criteria of "sufficiently close".
> On the other hand, I can also envision situations where
> the number could be orders of magnitude larger and still
> work for a particular application. I'm probably mistaken,
> but it seems to me "sufficiently close" is an arbitrary
> value that must be picked empirically to match the data
> and what you are trying to do with it.
>
> Cheers,
>
> David
>

> P.S. I'm just thinking that "sufficiently close" to a
> black hole, for example, might be a completely different
> number than "sufficiently close" to my house.
>
> --
> 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.")

Hmm... I think Jean might be on to something. After all, the error in
question hear is the rounding error of the computer, and that rounding
error is always an error on the last 'bit' of a floating point
number. So for instance if you had two floating point numbers:

1.1123453e15
and
1.1123454e15

These might be the same number (to within the rounding error) but the
difference between them is about 6.7e07. That's assuming of course
that I'm properly understanding floating point representation (I'm an
astronomer, not a computer engineer).

Subject: Re: Comparing 2 arrays
Posted by David Fanning on Mon, 27 Aug 2007 15:06:08 GMT
View Forum Message <> Reply to Message

Conor writes:

> Hmm... I think Jean might be on to something. After all, the error in
> question hear is the rounding error of the computer, and that rounding
> error is always an error on the last 'bit' of a floating point
> number. So for instance if you had two floating point numbers:
>
> 1.1123453e15
> and
> 1.1123454e15
>
> These might be the same number (to within the rounding error) but the
> difference between them is about 6.7e07. That's assuming of course
> that I'm properly understanding floating point representation (I'm an
> astronomer, not a computer engineer).

I guess I'm thinking more about how a number got INTO
the array in the first place. If it got there as a result
of some kind of calculation, accumulative rounding errors

could be a great deal larger than machine precision. And
yet, you might still want these numbers to be "equal" for
analysis purposes.

Cheers,

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

Subject: Re: Comparing 2 arrays
Posted by James Kuyper on Mon, 27 Aug 2007 17:40:40 GMT

View Forum Message <> Reply to Message

Conor wrote:
> On Aug 26, 12:43 pm, David Fanning <n...@dfanning.com> wrote:
>> Jean H. writes:
>>> to get back to a previous discussion we had a few month ago about being
>>> "sufficiently close to zero", shouldn't it be (data1.A - data2.B) LT
>>> epsilon * data1.A , with epsilon=(machar()).eps?
>>
>> Humm, I don't recall that discussion. But I can see how
>> this number might meet the criteria of "sufficiently close".
>> On the other hand, I can also envision situations where
>> the number could be orders of magnitude larger and still
>> work for a particular application. I'm probably mistaken,
>> but it seems to me "sufficiently close" is an arbitrary
>> value that must be picked empirically to match the data
>> and what you are trying to do with it.
>>
>> Cheers,
>>
>> David
>>
>> P.S. I'm just thinking that "sufficiently close" to a
>> black hole, for example, might be a completely different
>> number than "sufficiently close" to my house.
>>
>> --
>> 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.")
>
> Hmm... I think Jean might be on to something. After all, the error in

> question hear is the rounding error of the computer, and that rounding
> error is always an error on the last 'bit' of a floating point
> number. So for instance if you had two floating point numbers:
>
> 1.1123453e15
> and
> 1.1123454e15
>
> These might be the same number (to within the rounding error) but the
> difference between them is about 6.7e07. That's assuming of course
> that I'm properly understanding floating point representation (I'm an
> astronomer, not a computer engineer).

The direct effect of a single roundoff error shouldn't be more than 1
bit in the last position. However, it is often the case that two
different numbers that should mathematically be the same, have been
brought together through a long series of operations. A roundoff error
in the first operation could be magnified or reduced by the next
operation, in addition to that operation creating round-off errors of
its own. In general, you must either analyze the propogation of error
through the calculations, or at least measure the typical error sizes
empirically, as David suggested.

Subject: Re: Comparing 2 arrays
Posted by Paul Van Delst[1] on Mon, 27 Aug 2007 17:52:04 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Conor writes:
>
>> Hmm... I think Jean might be on to something. After all, the error in
>> question hear is the rounding error of the computer, and that rounding
>> error is always an error on the last 'bit' of a floating point
>> number. So for instance if you had two floating point numbers:
>>
>> 1.1123453e15
>> and
>> 1.1123454e15
>>
>> These might be the same number (to within the rounding error) but the
>> difference between them is about 6.7e07. That's assuming of course
>> that I'm properly understanding floating point representation (I'm an
>> astronomer, not a computer engineer).
>
> I guess I'm thinking more about how a number got INTO
> the array in the first place. If it got there as a result
> of some kind of calculation, accumulative rounding errors

> could be a great deal larger than machine precision. And
> yet, you might still want these numbers to be "equal" for
> analysis purposes.

Jean H. is absolutely correct. The magnitude of the numbers in question must enter into the computation of the value that corresponds to "close enough to zero".

E.g. If elements of the array that you are comparing are 1.23456789e+64 and 1.23456783e+64, then comparing to an absolute value that assumes the values are near 1.0 (such as 1.0e-06) will alway fail even if they are considered equal for the application in question.

In Fortran95-speak, the equivalent is:

```
  ABS(x - y) < (ulp * SPACING(MAX(ABS(x),ABS(y))))
```

where if the result is .TRUE., the numbers are considered equal.

The intrinsic function SPACING(x) returns the absolute spacing of numbers near the value of x,

```
                 {    EXPONENT(x)-DIGITS(x)
                 {  2.0                    for x /= 0
   SPACING(x) = {
                 {
                 {  TINY(x)                  for x == 0
```

The "ulp" factor scales the comparison.

So, Jean H's "epsilon * data1.A" is a way of computing the absolute spacing between two numbers in IDL-space.

cheers,

paulv

Subject: Re: Comparing 2 arrays
Posted by David Fanning on Tue, 28 Aug 2007 15:15:20 GMT

View Forum Message <> Reply to Message

Jean H. writes:

> to get back to a previous discussion we had a few month ago about being
> "sufficiently close to zero", shouldn't it be (data1.A - data2.B) LT
> epsilon * data1.A , with epsilon=(machar()).eps?

OK, I found that discussion and read it eight or ten times until

I finally understood it. (Probably why I forgot it before.)

I've put a significantly edited discussion of this
problem here:

http://www.dfanning.com/code_tips/comparearray.html

In my preferred solution now, I choose a number that
is "sufficiently close" to zero like this:

```
epsilon = (MACHAR()).eps
NUMBER = (array_1 > array_2) * epsilon
```

Then, the comparison between arrays is done like this:

```
IF Total(Abs(array_1 - array_2) LT NUMBER) EQ N_Elements(array_1) $
   THEN RETURN, 1 ELSE RETURN, 0
```

Additional comments welcome if you want to argue further. :-)

Cheers,

David
--
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.")