Subject: accuracy problems
Posted by on Thu, 27 Apr 2000 07:00:00 GMT
View Forum Message <> Reply to Message

HY to all!

I've a very strange problem: I'm running IDL 5.2.1 on a Pentium III 500 with Windows NT. When I read data from a file into a variable, IDL always makes a truncation. Even when I type it at the IDL prompt.
For example:

IDL> a=2.83456
IDL> print, a

IDL prints : 2.00000

Its equal if I define the variable as float or double, the result is always the same.

Does anyone have a idea how this can happen,

Subject: Re: Accuracy problem
Posted by David Fanning on Sat, 11 Mar 2006 05:43:38 GMT
View Forum Message <> Reply to Message

Juan Arrieta writes:

> I am preparing a simple code to transform between cartesian vectors
> and Keplerian elements (semimajor axis, inclination, eccentricity, and
> so forth). This is a simple problem in astrodynamics.
>
> At some point in my code, I need to obtain unit vectors. For instance,
> a line of the code is:
>
> U0 = ACOS((TRANSPOSE(NDVCT) # R) / (NORM(NDVCT) * NORM(R)))
> % Program caused arithmetic error: Floating illegal operand
> print,U0
> NaN
>
> Any comments as for what would the problem be? This seems like a
> roundoff error somewhere in the program, but I am not doing anything
> "fancy" here.

Getting errors with computers (alas!) doesn't require much "fancy" programming. I don't know exactly what the problem

is here, but clearly you need more precision. I'd start by doing everything in DOUBLE precision. You don't tell us what datatype NDVCT and R are but the NORM function is being done as a FLOAT, since you haven't set the DOUBLE keyword.

I'd step back a couple of steps before you introduced us to the problem and make sure everything is done in double precision values. Then, after you have convinced yourself you've done everything humanly possible, I think you might be justified in confining the arguments to ACOS to the proper range:

```
arg = ACOS(0.0 > expr < 1.0)
```

Cheers,

David

P.S. I will say that I run into a LOT of floating underflow errors when I am working with astronomy data. Mostly when values get close to 0. (I see a -0.000000 value in your example.) Maybe these are generated when images are flat-fielded, or processed in other ways. I've even resorted to cleaning these "close to zero" values up before I work with the images. It tends to keep the blood pressure down a little.

```
I = Where(image GT -1e-8 AND image LT 1e-8, count)
if count GT 0 THEN image[I] = 0.0
```

I just checked my pulse, and look at that, my blood pressure is rising just *thinking* of those damn floating underflow messages!
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Accuracy problem
Posted by mmeron on Sat, 11 Mar 2006 06:52:55 GMT
View Forum Message <> Reply to Message

In article <MPG.1e7c116c2aa1c294989bd5@news.frii.com>, David Fanning <davidf@dfanning.com> writes:
> Juan Arrieta writes:
>
>> I am preparing a simple code to transform between cartesian vectors
>> and Keplerian elements (semimajor axis, inclination, eccentricity, and

>> so forth). This is a simple problem in astrodynamics.
>>
>> At some point in my code, I need to obtain unit vectors. For instance,
>> a line of the code is:
>>
>> U0 = ACOS((TRANSPOSE(NDVCT) # R) / (NORM(NDVCT) * NORM(R)))
>> % Program caused arithmetic error: Floating illegal operand
>> print,U0
>> NaN
>>
>>   Any comments as for what would the problem be? This seems like a
>> roundoff error somewhere in the program, but I am not doing anything
>> "fancy" here.
>
> Getting errors with computers (alas!) doesn't require much
> "fancy" programming. I don't know exactly what the problem
> is here, but clearly you need more precision. I'd start by
> doing everything in DOUBLE precision. You don't tell us
> what datatype NDVCT and R are but the NORM function is
> being done as a FLOAT, since you haven't set the DOUBLE
> keyword.
>
> I'd step back a couple of steps before you introduced us
> to the problem and make sure everything is done in double
> precision values. Then, after you have convinced yourself
> you've done everything humanly possible, I think you might
> be justified in confining the arguments to ACOS to the
> proper range:
>
>   arg = ACOS(0.0 > expr < 1.0)
>
> Cheers,
>
> David
>
> P.S. I will say that I run into a LOT of floating underflow
> errors when I am working with astronomy data. Mostly when
> values get close to 0. (I see a -0.000000 value in your
> example.) Maybe these are generated when images are flat-fielded,
> or processed in other ways. I've even resorted to cleaning
> these "close to zero" values up before I work with the images.
> It tends to keep the blood pressure down a little.
>
>   I = Where(image GT -1e-8 AND image LT 1e-8, count)
>   if count GT 0 THEN image[I] = 0.0
>
> I just checked my pulse, and look at that, my blood pressure
> is rising just *thinking* of those damn floating underflow messages!

> --
Well, they're a nuisance, or rather used to be one. I've a little
routine in my library (called FPU_FIX) which is killing them on sight
so nowadays I hardly ever see them.

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

Subject: Re: Accuracy problem
Posted by James Kuyper on Sat, 11 Mar 2006 11:56:40 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Juan Arrieta writes:
>
>> I am preparing a simple code to transform between cartesian vectors
>> and Keplerian elements (semimajor axis, inclination, eccentricity, and
>> so forth). This is a simple problem in astrodynamics.
>>
>> At some point in my code, I need to obtain unit vectors. For instance,
>> a line of the code is:
>>
>> U0 = ACOS((TRANSPOSE(NDVCT) # R) / (NORM(NDVCT) * NORM(R)))
>> % Program caused arithmetic error: Floating illegal operand
>> print,U0
>> NaN
...
> "fancy" programming. I don't know exactly what the problem
> is here, but clearly you need more precision. I'd start by
> doing everything in DOUBLE precision. You don't tell us
> what datatype NDVCT and R are but the NORM function is
> being done as a FLOAT, since you haven't set the DOUBLE
> keyword.
>
> I'd step back a couple of steps before you introduced us
> to the problem and make sure everything is done in double
> precision values. Then, after you have convinced yourself
> you've done everything humanly possible, I think you might
> be justified in confining the arguments to ACOS to the
> proper range:
>
> arg = ACOS(0.0 > expr < 1.0)

Using double precision floating point will help, but even with double
precision, for some sufficiently small value epsilon, any expression
whose mathematically exact result lies between 1.0-epsilon and 1.0 has
a chance of evaluating numerically to a result >1.0, due to roundoff

error. You HAVE to cap ACOS(-1.0 > expr < 1.0) to cope with these problems. I wouldn't recommend a lower limit of 0.0 unless that's actually indicated by the nature of your problem. Small negative arguments to ACOS don't pose the same kind of problem that arguments slightly greater than 1.0 do.

Subject: Re: Accuracy problem
Posted by David Fanning on Sat, 11 Mar 2006 14:14:25 GMT
View Forum Message <> Reply to Message

Mati Meron writes:

> Well, they're a nuisance, or rather used to be one. I've a little
> routine in my library (called FPU_FIX) which is killing them on sight
> so nowadays I hardly ever see them.

My goodness, Sir! May I nominate you for RSI's Man of the Year?

Cheers,

David

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

Subject: Re: Accuracy problem
Posted by mmeron on Sun, 12 Mar 2006 00:25:10 GMT
View Forum Message <> Reply to Message

In article <MPG.1e7c892ff165dada989bd6@news.frii.com>, David Fanning <davidf@dfanning.com> writes:
> Mati Meron writes:
>
>> Well, they're a nuisance, or rather used to be one. I've a little
>> routine in my library (called FPU_FIX) which is killing them on sight
>> so nowadays I hardly ever see them.
>
> My goodness, Sir! May I nominate you for RSI's Man of the Year?
>
I'll settle for a beer, instead:-)

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

Subject: Re: Accuracy problem
Posted by Juan Arrieta on Sun, 12 Mar 2006 03:37:54 GMT
View Forum Message <> Reply to Message

Dear All:
Many thanks for your replies.
I have implemented both corrections, namely using the double precision version of the NORM subroutine, and adding the argument bounds for the ACOS function. Now the code is working as expected.

I yet have to see exaclty where was the accuracy lost. As soon as I find it out, I will post another message in this same topic tree, so others can be benefited from my (bad) experience.

Again, many thanks for your replies. Happy IDL-ing.

Juan J. Arrieta-Camacho
Carnegie Mellon University

Subject: Re: Accuracy problem
Posted by kehcheng on Sun, 12 Mar 2006 06:45:21 GMT
View Forum Message <> Reply to Message

A better solution is to use the formula ATAN(NORM(C), TRANSPOSE(A)#B), where C is A cross B, for the angle between vectors A and B. There is no need to limit the two arguments of ATAN, and you'll get a much more accurate result when A and B are almost parallel (or antiparallel). Example: A=[1.d0,0,0], B=[1.d0,0,1.d-9]. The ACOS formula gives 0.0 for the angle; the ATAN formula gives 1.d-9.