Subject: IDL's BESELJ returns NAN for small argument and large order
Posted by David Ruffner on Tue, 18 Mar 2014 14:55:16 GMT
View Forum Message <> Reply to Message

Hi all,
I found what I think is a bug in IDL's BESELJ function when there is small argument and large order. For instance,

```
IDL> print,beselj(0.1d,103.d)
           NaN
% Program caused arithmetic error: Floating underflow.
```

I think the problem is that the number is too small for IDL to handle because if I do a smaller order then the result is a very small number,

```
IDL> print,beselj(0.1d,102.d)
  2.0511844e-295
% Program caused arithmetic error: Floating underflow.
```

Shouldn't IDL just return zero in this case? Any thoughts would be helpful.

Thanks,
David

Subject: Re: IDL's BESELJ returns NAN for small argument and large order
Posted by Phillip Bitzer on Tue, 18 Mar 2014 23:47:28 GMT
View Forum Message <> Reply to Message

"That's not a bug, it's a feature" :-)

The reason you get a NaN can be found in the help:

ITER
Set this keyword equal to a named variable that will contain the number of iterations performed. If the routine converged, the stored value will be equal to the order N. If X or N are arrays, ITER will contain a scalar representing the maximum number of iterations.
Note: If the routine did not converge for an element of X, the corresponding element of the Result array will be set to the IEEE floating-point value NaN, and ITER will contain the largest order that would have converged for that X value.

So,

```
IDL> print,beselj(0.1d,103.d, ITER=n)
IDL> print, n      ;get n=102
```

So, the algorithm properly converges for order 102, but not 103+. This is why get a number for your second example.

Subject: Re: IDL's BESELJ returns NAN for small argument and large order
Posted by David Ruffner on Wed, 19 Mar 2014 20:48:42 GMT
View Forum Message <> Reply to Message

On Tuesday, March 18, 2014 7:47:28 PM UTC-4, Phillip Bitzer wrote:
> "That's not a bug, it's a feature" :-)
>
>
>
> The reason you get a NaN can be found in the help:
>
>
>
> ITER
>
> Set this keyword equal to a named variable that will contain the number of iterations performed. If the routine converged, the stored value will be equal to the order N. If X or N are arrays, ITER will contain a scalar representing the maximum number of iterations.
>
> Note: If the routine did not converge for an element of X, the corresponding element of the Result array will be set to the IEEE floating-point value NaN, and ITER will contain the largest order that would have converged for that X value.
>
>
>
> So,
>
> IDL> print,beselj(0.1d,103.d, ITER=n)
>
> IDL> print, n ;get n=102
>
>
>
> So, the algorithm properly converges for order 102, but not 103+. This is why get a number for your second example.

Hi Phillip,
Thanks for your reply. I see what you are saying. It's better that beselj returns a NAN than silently returning something that could be wrong.

I checked the series expansion on Mathworld and found that for integer order and small argument J_n(x) should go to zero. I'm not sure about fractional order which may be why beselj returns the NAN. In my code I only need to use integer order so I just catch the NANs and set beselj to zero whenever I am sure it should be zero.

Thanks,
David

Subject: Re: IDL's BESELJ returns NAN for small argument and large order
Posted by dg86 on Fri, 21 Mar 2014 00:27:21 GMT
View Forum Message <> Reply to Message

On Tuesday, March 18, 2014 7:47:28 PM UTC-4, Phillip Bitzer wrote:
> "That's not a bug, it's a feature" :-)
>
>
>
> The reason you get a NaN can be found in the help:
>
>
>
> ITER
>
> Set this keyword equal to a named variable that will contain the number of iterations performed. If the routine converged, the stored value will be equal to the order N. If X or N are arrays, ITER will contain a scalar representing the maximum number of iterations.
>
> Note: If the routine did not converge for an element of X, the corresponding element of the Result array will be set to the IEEE floating-point value NaN, and ITER will contain the largest order that would have converged for that X value.
>
>
>
> So,
>
> IDL> print,beselj(0.1d,103.d, ITER=n)
>
> IDL> print, n ;get n=102
>
>
>
> So, the algorithm properly converges for order 102, but not 103+. This is why get a number for your second example.

This is a pretty crummy feature, even if it is documented. IDL's implementation of Bessel functions is not nearly so comprehensive as python's (using the mpmath package) or Mathematica's. Even if the bad behavior is documented, it's still bad. These seemingly extreme cases come up regularly in light-scattering calculations, so they're not outlandish.

The developers should update IDL's special functions to bring them up to industry standards. The present Numerical Recipes implementations are not up to that level. The new versions of the Bessel functions might even be spelled correctly, leaving the oddly speled versions for backward compatibility.

Just my 2c,

David