Subject: Re: Math Question
Posted by James Kuyper on Mon, 30 Oct 2006 17:04:46 GMT
View Forum Message <> Reply to Message

Rob wrote:

```
>  In general, powers of negative numbers are complex so you should start
>  with that assumption in the expression:
>
>  IDL> print,(dcomplex(-1.0,0.0))^2.01
>  (      0.99950656,     0.031410729)
>  IDL> print,(dcomplex(-1.0,0.0))^2.0
>  (       1.0000000, -2.4492127e-016)
>
>  It would be nice if IDL told you this rather than throwing a NaN at
>  you.
>
>  Rob
>
>  On Oct 29, 11:02 am, David Fanning <n...@dfanning.com> wrote:
>>  Folks,
>>
>>  OK, I get the feeling that I am going to be referred to
>>  my own web page with this question:
>>
>>     http://www.dfanning.com/math_tips/sky_is_falling.html
>>
>>  And it is certainly true that I have been watching WAY
>>  too much TV lately (World Series, you know), but here is
>>  my question. How does one explain the following two
>>  IDL commands?
>>
>>    IDL> Help, (-0.1)^2.0
>>     <Expression>    FLOAT     =     0.0100000
>>    IDL> Help, (-0.1)^2.01
>>     <Expression>    FLOAT     =          -NaN
>>
>>  In general, raising a negative number to any integer
>>  power seems to produce a real number, whereas raising
>>  a negative number to a non-integer power causes a NAN.
>>
>>  I am sure this is explained in one of those textbooks
>>  covered with dust in my garage, but I thought one of
>>  you math guys could rescue me from a beautiful day
>>  spent covered with dust. :-)
```

If a and b are mutually prime integers, then mathematically, x^(a/b) has b different values, at most two of which are real, the others are complex. If b is odd, only one of the values is real. This is true, whether or not 'x' is negative. However, I don't think we want to have (-0.10)^2.01 return a list of 200 different complex values. Even when the operands are complex number IDL only attempts to return one of the possible values.

More imporatantly, I don't think it should return any complex values at all. In most cases, if the operands of the ^ operator aren't already complex, there's a pretty good chance that the code wasn't written to handle the possiblity of complex values from the result.

Considered solely as a function whose domain and range are restricted to real numbers, x^(a/b), should have two solutions if b is even, and 1 solution if b is odd, regardless of whether x is positive. In practice, for computer arithmetic that's not generally implementable. When you calculate x^y, y can, in general, only be a floating point approximation of a/b; it is an exact representation only if b is a power of 2. I've used implementations of the power function that attempt to recognize when y is an approximation of a/b for small values of b, however, I gather that IDL is implemented using C code, expwhich probably calls pow(x,y) to implement x^y. The pow() function does not attempt to recognise approximations to rational numbers with small denominators. It is required by the C standard to return a domain error if x is negative and y is a finite non-integer. IDL handles this by returning a NaN. This strikes me as a reasonable approach.