
Subject: Re: A bug in MOD ?

Posted by [Chris Lee](#) on Fri, 24 Sep 2004 17:06:53 GMT

[View Forum Message](#) <> [Reply to Message](#)

In article <cj1cqesgut\$1@canarie.caspar.it>, "sandrokan"
<mura@remove.ifs.rm.cnr.it> wrote:

```
> I don't know much about libs, I only have IDL and another s/w: IDL>
> print, 1.0 mod 0.1
> 0.100000
> but:
>
>>> mod(1.0, 0.1)
> ans =
>      0
>>>
```

Hi,

Ah, matlab, wonderful matlab. I think Matlab uses arbitrary precision math. where this answer is correct. I could be wrong of course.

The answer lies in the floating point representation of 1.0 and 0.1, or any number. One of the numbers are really what they appear (not sure which one) and the result is that ..

```
floor(1.0/0.1)=9
1.0 mod 0.1 = 0.1
```

;these may not work in any known language, but they do show what's happening.

Calculating $0.8 \bmod 0.1$, you get the correct answer, because whatever representation error exists in 0.8 also exists in 0.1 .similarly for 1.0 and 0.5|0.25|0.125 (powers of 2).

This is true of the IDL mod, the C++ fmod call (and probably the C library fmodf call, as its used internally in C++), the fortran mod function, the python mod function, etc.

I'm not sure what the correct method would be. I can't really round a value to zero when the value is comparable to the denominator in the 'mod' equation. It gets worse when I realize I've used 'mod' on a floating point before, in FORTRAN code.

Chris.
