## Subject: Re: A bug in MOD ?
## Posted by Paolo Grigis on Mon, 27 Sep 2004 07:37:47 GMT

View Forum Message <> Reply to Message

Well, IDL does give the right answer (modulo 0.1 of course!)
within the floating point precision limits...

```
print,(1. mod 0.1),format='(f20.15)'
   0.099999986588955
print,abs((1. mod 0.1)-0.1) LT (machar()).eps
   1
```

and of course 0.09999... is approximately equal 0.0000...
(modulo 0.1). So the question would be: why does it matter?
The whole point of taking the modulo is to have numbers near
0.1 being "close neighbours" to numbers near 0.0 anyway...

Paolo

Christopher Lee wrote:

```
>  In article <cj1cqe$gut$1@canarie.caspur.it>, "sandrokan"
>  <mura@remove.ifsi.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 mod 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 internallyin 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.