

Re: 0^0 oh no!

Source: <http://sci.tech--archive.net/Archive/sci.math.symbolic/2008-03/msg00127.html>

- *From:* Daniel Lichtblau <danl@xxxxxxxxxxxx>
 - *Date:* Mon, 24 Mar 2008 18:29:06 -0700 (PDT)
-

On Mar 22, 7:56 am, Christopher Creutzig <christop...@xxxxxxxxxxxx> wrote:

Daniel Lichtblau wrote:

I'm not convinced this is a bad thing, so much as a bad usage of SetPrecision (in Mathematica; I suspect there are similar functions in other programs). What it can do is to expose guard bits, in the sense of claiming "regard these bad bits as good bits". Returning to the

When writing some iterative numerical method, where you know that the iteration converges, but significance arithmetics won't be able to detect this fact (*), you'd use SetPrecision to tell the system to regard those bits as good, or do I miss something again?

Correct, in Mathematica one would typically use SetPrecision if coding an algorithm with some a priori known rate of convergence (linear or quadratic, say). I think something similar would be done in other languages as well. For languages that work in fixed precision (I think this means all computational math programs other than Mathematica, at least for default behavior) this might be done simply by interjecting code to raise the internal "working precision"

That would lead to a genuine usage of that function where it would cause things like $o1 == o2$, yet $f(o1) \neq f(o2)$, as I mentioned.

Yes, I guess this can happen in legitimate usage, more or less. But I think typically one would, at the finish of an iterative algorithm, use some method to adjust and perhaps slightly downgrade final precision.

So yes, you can have $o1 == o2$ and $f(o1) \neq f(o2)$, and I suspect this can happen without any interjection of SetPrecision. But this is sort of

Re: 0^0 oh no!

independent of handling of convergent iterative algorithms. And at the end of the day, what you want from such an algorithm is some idea of what are the significant digits in the result. The fact that you might have a "nearby" number that looks like yours, but disagrees when evaluated by your function of interest, is, I think not so serious a matter. We might be in disagreement on this. I suspect we at least agree it is a "finer" point and takes a back seat to getting the result to some desired precision.

(*) This is not a critique on significance arithmetics. It has been a rather long time until interval numerical analysts found out why some of the rather basic linear algebra algorithms actually return much more precise results than could be expected from analyzing individual steps. This is even true for non-iterative algorithms such as Gaussian elimination. I would expect direct application of significance arithmetics to yield equally pessimistic estimates for the result precision.

Yes indeed, at least for Gaussian elimination with partial pivoting. I tried this in the Mathematica kernel, back around 1994. I quickly went back to fixed precision with error approximation based on condition number. Significance arithmetic was woefully too conservative a model/tactic for (over)estimating error.

This is one reason I have not been too eager to get Mathematica's symbolic/exact Gaussian elimination to work too hard on matrices with intervals. I believe the pivot selection currently in place at least attempts to enforce that pivots not be intervals containing zero. But I do not think we try to do anything tantamount to partial pivoting (with intervals, I'd imagine such a pivot selection would then be based on smallest relative error rather than largest magnitude, amongst contending matrix entries).

```
In[28]:= SetPrecision[sum2plus,5] == SetPrecision[sum2,5]
Out[28]= False
```

But these new objects really ARE different, because we promoted bad bits to become good bits.

And, in my interpretation, that implies that the old objects are different, too. After all, there is a way to tell them apart. But I realize that different programming languages have different operators for equality, equivalence, identity, etc.

Re: 0^0 oh no!

Re: 0^0 oh no!

Yes to all this. Mathematica has a fuzzy equality and also enforces trichotomy, that is to say, we cannot have $x < y$ and $x = y$ simultaneously. I am sure this makes for pathological examples. I do not see a way around this; we all must pick from amongst imperfect scenarios.

--

if all this stuff was simple, we'd probably be doing something else. -- Daniel Lichtblau, s.m.symbolic

Daniel Lichtblau
Wolfram Research