HP Forum Archive 13

RPN vs. AOS: *GOOD* examples [LONG]

Message #1 Posted by ***Valentin Albillo*** *on 6 Oct 2003, 6:31 a.m.*

In regard with my criticism of the Mach number formula as a totally inadequate example to prove RPN's "superiority" over capable algebraic systems, WXYZ posted:

"You say that thats a flawed example.... could you give a good one?"

Yes, I can. First of all, let's make it clear from the start that RPN is *not* superior to algebraic systems, in general: there will be cases where it will certainly be superior, there will be cases where it will come out as inferior.

In particular, the evaluation of lengthy, complicated, algebraically-written formulas is about the *worst* possible case, and the one where any owner of a decent algebraic machine *who knows how to use it proficiently* (i.e: not the algebraically-challenged, RPN fundamentalist, who won't be able to even enter the formula without getting a syntax error, and will make a mess of it all) will not only give RPN a run for its money, but will win the race outright. In the case of the Mach number formula, said efficient user of an algebraic calculator, who knows his machine inside out, will key in the formula and get the correct result before many an RPN user has even decided on "where to begin" (as this formula cannot be attacked left-to-right in a four-level RPN stack) and "the proper key sequence".

What would then be a good example ? Well, to answer this question you just need to consider what are the main RPN strengths and differential characteristics. These are, of course, the user-manageable 4-level stack (instead of the internal, non-user-accessible stack typical of algebraic machines), and the powerful set of operations acting on it, i.e: X<>Y, Roll down and up, ENTER, automatic top-level replication, LASTx, etc. Any good example will make use to the most of some or all of those characteristics, to achieve results much more efficiently, faster, and in significantly less keystrokes than any algebraic system could. The Mach number formula fails miserably in taking advantage of any of those advanced RPN capabilities to any significant extent, which brings us to the original question: what would then be a *good* example ?

Well, I can think of many. "Think" is not the correct term however, as I do not need to "think" of a proper example. I've used RPN extensively in the last 20+ years, and I've come across many computations where it was most efficient. Just to name a few: continued-fraction evaluations, rational least-squares approximations, series expansions [*very* good examples here], Fourier series, complex-number operations, coordinate transformations, ... but there's a very simple example, that doesn't require high math at all, and which HP did include in each and every manual and many brochures as well, and that is "polynomial evaluation" using Horner's rule. RPN fits like a glove for this particular computation.

An example will make it clear. How would you evaluate the polynomial

```
8*x^4 + 5*x^3 + 2*x^2 - 7*x - 6
```

for arbitrary values of x ? Well, the RPN answer is simply (assuming x is already in the display):

```
ENTER, ENTER, ENTER, 8, *, 5, +, *, 2, +, *, 7, -, *, 6, -
```

Simple as it is, you're using both the stack to hold intermediate results and new operands, and *automatic top-level replication*, to keep a *never-ending* supply of x values. The computation is as fast and efficient as it can be.

On the other hand, if the user of an algebraic system wants to maintain the valuable left-to-right, no-thinking-needed characteristics of his system, he would have to enter the expression as written, i.e:

```
8*x^4 + 5*x^3 + 2*x^2 - 7*x - 6
```

but this is *vastly inefficient*, because he ends up doing four multiplications *and*, additionally, THREE *slow* x^y operations, not to mention the ancillary additions and subtractions. This is awfully inefficient, can result in greater rounding errors, and it's painfully slow. For higher-degree polynomials, it gets much worse.

Now, a knowledgeable algebraic user would realize this, and would probably sacrifice the "as written, left-to-right" entry, and try instead evaluating the polynomial by mimicking the RPN approach, like this:

```
(((8*x + 5)*x + 2)*x - 7)*x -6
```

but he then needs to first open a number of left parentheses in a row, then keep on closing parentheses at the proper times, not to mention the need to re-enter or recall the x value constantly. This results in many extra keystrokes needed, and for sufficiently high-degree polynomials this approach *is doomed to *fail**, as soon as the maximum number of open parentheses is reached, while the RPN approach works for polynomials of arbitrary degree. There's also the fact that having many pending operations does waste a lot of memory, while the RPN approach needs no extra memory at all but the stack registers themselves.

So you see, this is a clear-cut case where RPN excels over any algebraic approach. Even HP recognized this fact, and the Curve Fitting pac for the HP-71B does include a binary (machine language) subprogram specifically designed for evaluating polynomials, which HP-71B's BASIC, awesome as it is, simply can't do efficiently, for the aforementioned reasons. Even so, said binary subprogram only succeeds in being twice as fast as its BASIC counterpart, and if I recall correctly, can only evaluate polynomials up to 18th-degree. An RPN (or RPL, or FORTH) approach can easily run rings around this.

How's that for an example ? :-)

Best regards from V.

Re: RPN vs. AOS: *GOOD* examples [LONG]

Message #2 Posted by ***hugh*** *on 7 Oct 2003, 5:50 p.m.,*
in response to message #1 by Valentin Albillo

a cheeky reply,

on the hg9g, your example is indeed shorter. working the formula as written:

M+ 8 M ^2 ^2 + 5 M 2nd x^3 + 2 M ^2 - 7 M - 6 = (20 buttons) or using nested evals: M+ - 6 + M () - 7 + M () 2 + M () 5 + 8 M = (20 again)

BUT your example falls short if you happen to have a zero coefficient (or more). if the 5 were a zero. you would still take 16 steps in your method, assuming you would enter zero for that stage. on the 9g, you would get:

M+ 8 M ^2 ^2 + 2 M ^2 - 7 M - 6 = 15 steps :-)

Re: RPN vs. AOS: *GOOD* examples [LONG]

Message #3 Posted by ***Valentin Albillo*** *on 8 Oct 2003, 5:22 a.m.,*
in response to message #2 by hugh

hugh posted:

"you would still take 16 steps in your method, assuming you would enter zero for that stage"

And why on earth should you "assume" that !? :-) Of course I wouldn't enter any "0, +" sequence, would you ? !!

Best regards from V.

[Return to Index | Top of Index]

GTO Go back to the main exhibit hall