HP Forum Archive 13

[Return to Index | Top of Index]

Reverse Engineering HP Solve [long]

*Message #1 Posted by **Patrick** on 15 Oct 2003, 3:28 a.m.*

For the past couple of months I've been engaged in the rather fun intellectual exercise of trying to reverse engineer the HP Solve algorithm, as found on the HP-34C and HP-15C (experiments suggest it is the same). My source material for this effort is primarily the HP Journal article by William H. Kahan on the implementation of Solve for the HP-34C. Very similar discussions appear in other HP literature, such as the HP-15C Advanced Functions Handbook.

After writing a program for the HP-11C which incorporates *most* of the details described in those articles, I wised up and realized that I first ought to build a spreadsheet which implements the algorithm correctly. The spreadsheet is a bit cumbersome as a programming language, but at least I was not restriced to fitting the "program" in to the relatively meager amount of resources offered by the 11C.

In Kahan's article, he uses several examples to illustrate how Solve behaves based on the following family of function:

```
f(x) = exp(x) + A*x + B
```

for various values of A, B and starting guesses.

I was successful in building a spreadsheet which generates the exact (well, almost exact) sequence of root approximations for those problems as does the HP-34C or HP-15C. To say the least, I was very happy there for a while. The spreadsheet correctly computes the secant approximation and takes into account most of the algorithm features described by Kahan.

There is a feature, though, which Kahan describes as *bending the secant*. This happens when a root has been *bracketed* between two guesses but the secant method, used blindly, would land you outside of those brackets. Alas, none of the problems he gives in the article need to bend the secant in order to find a root. Once I found a problem which did have to bend the secant, my spreadsheet results diverged from those produced by my calculators.

Sigh.

So, here I am with a plea to the HP calculator community. If anyone is aware of *any* details regarding the HP Solve algorithm, especially in regards to bending secants, I would be quite grateful for a hint! Dr. Kahan, are you out there? ;-)

Re: Reverse Engineering HP Solve [long]

*Message #2 Posted by **Veli-Pekka Nousiainen** on 15 Oct 2003, 4:06 a.m.,*
in response to message #1 by Patrick

Patrick: could you publish the spreadsheet as it is, please? VPN

Re: Reverse Engineering HP Solve [long]

*Message #3 Posted by **Patrick** on 15 Oct 2003, 4:29 p.m.,*
in response to message #2 by Veli-Pekka Nousiainen

Yes, I will publish the spreadsheet but I would *really* like to get it to be much more faithful to the HP Solve algorithm first. I would also have to write a mess of documentation for it. I wrote this thing for my own use. To get it to the point where it would useful to others is a job.

My question to you all is, what is the appropriate medium for publication? There is also the HP-11C program which implements the algorithm (to some degree!). Maybe our esteemed Curator would like to weigh in on this?

Re: Reverse Engineering HP Solve [long]

*Message #4 Posted by **Bruce Horrocks** on 16 Oct 2003, 10:16 a.m.,*
in response to message #3 by Patrick

> My question to you all is, what is the appropriate medium for publication?

I would be very happy to publish your discoveries in Datafile (www.hpcc.org) if you are willing to write them up in article form.

Bruce <mailto:editor@hpcc.org_sans_spam>

Re: Reverse Engineering HP Solve [long]

*Message #5 Posted by **Dave Hicks** on 16 Oct 2003, 3:33 p.m.,*
in response to message #3 by Patrick

My first thought is the articles forum. If your post is too complex for forum formatting, you can send me HTML and I can insert it into an article post.

David, any chance ?

*Message #6 Posted by **Valentin Albillo** on 17 Oct 2003, 4:24 a.m.,*
in response to message #5 by Dave Hicks

David posted:

"My first thought is the articles forum. If your post is too complex for forum formatting, you can send me HTML and I can insert it into an article post. "

Any chance you would reconsider admitting articles in PDF format ? :-)

Best regards from V.

Edited: 17 Oct 2003, 5:24 a.m.

Re: Reverse Engineering HP Solve [long]

*Message #7 Posted by **Herman** on 15 Oct 2003, 8:35 a.m.,*
in response to message #1 by Patrick

Can the spreadsheet solve f(x) = cos(nx), where n is an integer, given initial guesses a = 0 and b = 2PI/n ?

Re: Reverse Engineering HP Solve [long]

*Message #8 Posted by **Patrick** on 15 Oct 2003, 4:56 p.m.,*
in response to message #7 by Herman

Well, no and ... yes.

Of course, as Kahan points out in his Solve article, any algorithm which depends upon function sampling can be led astray by a careful selection of the function.

Your example, though, is problematic to the HP Solve algorithm due to the mulitples of pi/n that the algorithm tends to generate. The culprit here is the number 100 in the part of the algorithm which makes the restriction:

```
|x3-x2| <= 100 |x2-x1|
```

See Kahan's article for a reference to this. The number 100 is problematic for your example mostly because it is an integer. If I change my spreadsheet to use the number 99.67 instead, it brackets a root and, therefore, eventually converges to it.

I just want to point out that my intent is to reverse engineer the existing HP Solve algorithm, as it is found on the HP-15C. My intent is not to build a better Solver (that would be considered "Phase 2" ;-).

Re: Reverse Engineering HP Solve [long]

*Message #9 Posted by **Patrick** on 15 Oct 2003, 4:59 p.m.,*
in response to message #8 by Patrick

BTW... it occurs to me that precisely the sort of circumstance exhibited by your example is what might have led HP to introduce the randomization I referred to in my initial post.

Solve in HP48

*Message #10 Posted by **Mike (Stgt)** on 15 Oct 2003, 9:20 a.m.,*
in response to message #1 by Patrick

Pls allow me a subject drift to the solver of HP-48GX. A colleague just asked me if the algorithm is available. He has written one in C that up to now never failed where the one from the HP48 did. (Well, I'll give him some patholotic functions that are known to fool every solver like

```
y = 3*SQ(x) + LN(SQ(PI - x))/SQ(SQ(PI)) + 1
```

) Now he wants to see how HP did implement SOLVE on the HP48.

TIA.....Mike

Re: Reverse Engineering HP Solve [long]

*Message #11 Posted by **David Brunell** on 15 Oct 2003, 10:44 a.m.,*
in response to message #1 by Patrick

Why not look at a 41C Advantage or Math Pac listing? I think the algorithm is pretty close to the 15C.

Re: Reverse Engineering HP Solve [long]

*Message #12 Posted by **Mike (Stgt)** on 15 Oct 2003, 11:12 a.m.,*
in response to message #11 by David Brunell

IFAIK, in the Advantage it's written in MCode. PPCROM would be a good idea as it comes with a very complete manual, no - documentation!

But I assume the algorithms in other HP calculators vary in some respects.

Re: Reverse Engineering HP Solve [long]

*Message #13 Posted by **Patrick** on 15 Oct 2003, 4:31 p.m.,*
in response to message #11 by David Brunell

I have the Advantage Pac, but the Solve algorithm is written in assembler. It is not listable as a normal HP-41C program. Of course, if I had such a listing, my reverse engineering job would be somewhat moot! I don't know if that would be a good thing or a bad one...!

Re: Reverse Engineering HP Solve [long]

*Message #14 Posted by **J-F Garnier** on 16 Oct 2003, 8:00 a.m.,*
in response to message #13 by Patrick

You can have a look at the Advantage Solve code with any emulator that supports ROM modules and have a disassembly function (for instance, my Emu41 :-)

J-F

Re: Reverse Engineering HP Solve [long]

*Message #15 Posted by **Valentin Albillo** on 15 Oct 2003, 11:41 a.m.,*
in response to message #1 by Patrick

Patrick wrote:

"There is a feature, though, which Kahan describes as bending the secant. This happens when a root has been bracketed between two guesses but the secant method, used blindly, would land you outside of those brackets. Alas, none of the problems he gives in the article need to bend the secant in order to find a root. Once I found a problem which did have to bend the secant, my spreadsheet results diverged from those produced by my calculators."

Please post *details*, i.e: exact "problem which did have to bend the secant", your exact and complete spreadsheet's results for said example, same for the actual calculator's results.

If you've got more than one "secant-bending" example available, the more the merrier.

Best regards from V.

Re: Reverse Engineering HP Solve [long]

*Message #16 Posted by **Patrick** on 15 Oct 2003, 3:13 p.m.,*
in response to message #15 by Valentin Albillo

To see the effect of the bent secant characteristic of HP Solve, try the following experiment. Using an HP-34C or an HP-15C (other calculators may use the same algorithm, I have not verified any others yet), key in a program which consists of a single instruction, R/S, sandwiched between a label and a RTN:

```
LBL A
R/S
RTN
```

This program forms my lab bench upon which I perform experiments on Solve. If you run "Solve A" the program will stop each time it calls your "function". The program simply waits for you to look at the x value in the display and type in the y value you want to correspond to it. You then press R/S to continue the Solve algorithm.

Using this technique, you can present the algorithm with any sort of situation you like, without having to craft some sort of mathematical function that does it for you.

With this setup, start Solve running with initial guesses x1=1 and x2=-1 (1, ENTER, CHS, SOLVE A). When your program stops to get y values, suppy the values y1=1 and y2=-1 (i.e., just press R/S each time). Of course the secant line between these two points lands you smack dab at the origin, so the next x point you'll be asked about is x3=0. Well, not quite. There is a certain amount of what can only be described as *randomization* inserted by HP Solve, so what you get is an x3 value *near* zero. For this x value, whatever it is, supply the y value y3=-0.8.

So far, you've given the algorithm three points to consider: (1,1), (-1,-1), and (0, -0.8). Note that the first two x values already *bracket* the root, since the corresponding y values are of opposite signs. The brackets become *tighter* once you know that the value of the function at zero is negative. Now the root should lie between 0 and 1. However, if you try to draw the secant line between the second and third points, the resulting x value is x4=4, outside of the brackets. This is where the bending of secant should take place, so as to guarantee x4 stays between the brackets: $0 < x4 < 1$.

My spreadsheet uses the following algorithm: If the secant method lands you outside the brackets, pick the midpoint of the brackets for the next trial. In this case, my spreadsheet would choose x4=0.5. The HP Solve algorithm on my HP-15C doesn't do this. It selects x4=0.6997201114.

My question is, how is that strange number selected?

In fact, after an initial period of experimentation, I believe that the HP Solve algorithm has computed:

```
x4 = base + randomizer
```

where base=0.7 and the randomizer is on the order of (5e-4 * base). I have an inkling as to why the randomization is included, but it is still pretty foggy to me.

Anyhow, why 0.7? That is my question.

Using this technique, one can generate any number of examples of when the bent secant algorithm is needed. Using a series of experiments, I have determined that:

- The x4 value scales linearly in x1 and x2 (i.e., replace x1 by c*x1 and x2 by c*x2 and the x4 value changes to c*x4)
- The x4 value scales linearly in y1 and y2.

- The x4 value does not depend on a value of x and y previous to (x1,y1).

I have computed the value of x4 for a host of values of y3 and have plotted the results, but don't have a functional relationship there yet. Still trying...

Re: Reverse Engineering HP Solve [long]

*Message #17 Posted by **Mike (Stgt)** on 15 Oct 2003, 3:50 p.m.,*
in response to message #16 by Patrick

Aha! Now I've got the point. Alas I can not tell you the solution right away.

Ciao.....Mike

Re: Reverse Engineering HP Solve [long]

*Message #18 Posted by **hugh** on 15 Oct 2003, 5:46 p.m.,*
in response to message #16 by Patrick

im wondering if its fitting some parabola to the points. according to the manual, it sometimes fits a parabola then takes the x value where this curve is min or max.

for (-1,-1),(0,-0.8),(1,1) i get y = 0.8x^2+x-0.8 using this directly for y = 0, gives x = 0.5542 and it mimimises at x = -0.625, neither of which give 0.7

inverse quadratic interpolation (brents method) gives x4 = 2.222 (out of range) and ridders method gives x4 = 0.6247

ridders method is pretty good btw, i'd recommend it as a possible improved algorithm.

Mystery Solved

*Message #19 Posted by **Patrick** on 23 Oct 2003, 1:45 p.m.,*
in response to message #16 by Patrick

Just wanted to let people know that I've solved this little mystery. I am now quite certain the "true" value in the example I illustrated was, in fact, 0.7. Referring to figure 6 in Kahan's article on the HP-34C Solve algorithm, I believe the following alogorithm is used to bend the secant (I write a, b,c,... instead of the Greek letters alpha, beta, gamma, ...):

Let

```
r = (a-c)/(s-c)
```

Where s is the place line 1 in that figure intersects the x axis. Thus, s is the place where the secant method would bring us to if we *didn't* bend the secant. Set r=0 if the line never intersects the x axis. We see that r is the ratio between the size of the best known brackets for the root and the distance the secant method wants to travel from the previous best estimate for the root (which is c). The HP-34C and HP-15C use the following formula for where the secant should be bent:

```
d = c + (a-c)*t
```

where

```
t = (2-r)/(3-2*r)
```

Applying this to the example I gave, c=0, a=1, s=4, so r=0.25 and t=0.7, as advertized.

As I mentioned in my previous post, this is not exact. There is some randomization added in. I doubt that I'll ever be able to figure out the formula for that since the effects are quite small.

Edited: 23 Oct 2003, 1:48 p.m.

[Return to Index | Top of Index]

Go back to the main exhibit hall