HP Forum Archive 20

[Return to Index | Top of Index]

HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #1 Posted by **Valentin Albillo** on 11 Nov 2011, 2:30 p.m.*

Hi all:

The recent release of the **HP-15C LE** is a momentous event which truly deserves a *great celebration* on my part, being as I am a life-long sworn fan of this wonderful model (most especially today which is the one-and-only **11-11-11** date you'll get to see for the next 100 years or so).

However, as I'm currently not in the mood for any great efforts and I'm feeling a little lazy today, methinks you'll have to make do with this *two-bit*

HP-15C (& LE!) 11-11-11 Mini-Challenge

We all know that *numerical integration* is all the rage these days in the forum with plenty of good algorithms being discussed and offered for consideration to the WP34S team and such. The best of those integration algorithms are powerful and complex, a many-byte affair.

However, on the other hand, the ugly duckling would be **numerical derivation** which, unlike integration, can usually be done by numerically evaluating a symbolically-obtained derived function y'(x) at any specified X ordinate. Regrettably, that requires some advanced symbolic manipulation which is simply unavailable in the HP-15C (&LE!) and thus numerical approximations is the only way to go.

So the challenge is:

You're asked to write a program for the **HP-15C (& LE!)** which must be as short as possible, and which can evaluate the **derivative** at a given ordinate X of a user-supplied function **y(x)**, subject to the following *mandatory* requirements.

The exact requirements your program must meet are:

1. first and foremost, it must be *as short as possible*, anything more than a dozen steps or two means back to the drawing board for you. The program must begin with **LBL A** and end with **RTN**.
2. it must accept a given real ordinate X in the display and produce the value of the *derivative* of **y(x)** at that ordinate, where **y(x)** is a user-supplied function which must be accessible under **LBL E**.
3. it must *not* depend on or require any particular setting or any particular memory allocation to be previously stablished.
4. it must *not* require any pre-stored constants in the registers or stack contents except for the ordinate X being initially in the display at runtime.
5. and last but not least, it must strive to produce *full 10-digit accuracy* (except for the usual cases of particular troublesome points or ranges, etc, of course).

Sample of use:

1. given $\mathbf{y(x) = e^x/Sqrt(sin(x)^3+cos(x)^3)}$, find **y'(1.5)**

```
define y(x) -> LBL E, e^X, LAST X, LAST X, SIN, 3, y^x, X<>Y, COS, 3, y^x, +, SQR, /, RTN

1.5, GOSUB A ->  4.053427894, which is the correct result to all 10 digits
```

2. given $\mathbf{y(x) = 7*e^{(2.1*X)}}$, find **y'(15)**

```
define y(x) -> LBL E, 2.1, *, e^X, 7, *, RTN

15, GOSUB A -> 7.040338081 E14, which is the correct result to all 12 digits
```

3. given $\mathbf{y(x) = sin(x)/e^x}$, find **y'(1)**

```
define y(x) -> LBL E, SIN, LAST X, e^X, /, RTN

1, GOSUB A -> -0.1107937653, whis is correct to all 10 digits
```

Within a few days I'll post my solution, which is just **11 steps** long, meets all five conditions above, and produces the sample results above correct to all digits shown. Relaxing a condition or two could shorten it even further, to just 6 steps or even just 5, but let's first keep all five requirements above and see what **YOU** can produce ... :)

Enjoy !

Best regards from V.

.

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #2 Posted by **Gerson W. Barbosa** on 11 Nov 2011, 5:11 p.m.,*
in response to message #1 by Valentin Albillo

Hello Valentin,

I think the following meets all your requirements:

```
001- f LBL A
002-   9
003-   CHS
004-   10^x
005-   STO 0
006-   Re<>Im
007-   +
008-   GSB E
009-   Re<>Im
010-   RCL/ 0
011- g RTN
```

Best regards,

Gerson.

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #3 Posted by **peacecalc** on 12 Nov 2011, 2:44 a.m.,*
in response to message #2 by Gerson W. Barbosa

Hello Gerson,

I'm sorry but I think your code doesn't function in the way you want.

I miss the "-" operation, which is nessecary for calculating: "f(x+h)/h - f(x)/h"

But your idea is genial, **parallel computing** with the hp 15c!

sincerely peacecalc

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #4 Posted by **Thomas Radtke** on 12 Nov 2011, 6:09 a.m.,*
in response to message #3 by peacecalc

Looks like Gerson ingeniously implemented this.

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #5 Posted by **Gerson W. Barbosa** on 12 Nov 2011, 8:17 a.m.,*
in response to message #4 by Thomas Radtke

My source, actually. That's what Wikipedia and Google are for :-) Technically we should get rid of the imaginary part in the end, either by setting the complex mode off or by clearing the register X before the second Re<>Im instruction. This takes one extra step, however.

Here is an HP-42S solution:

```
00 { 27-Byte Prgm }
01 LBL "DRV"
02 1E-9
03 RCL* ST Y
04 STO 00
05 COMPLEX
06 XEQ "FN"
07 COMPLEX
08 RCL/ 00
09 END
```

Example:

```
00 { 8-Byte Program }
01 LBL "FN"
```

```
02 LN
03 RTN


1E-490 XEQ DRV -> 1.E490
       XEQ DRV -> 1.E-490
```

You might want to insert X<>Y and Rv between the steps 08 and 09. The answer will be valid only if FN returns a real result for the given X. A few more steps might handle this.

Gerson.

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #6 Posted by **Werner** on 12 Nov 2011, 10:13 a.m.,*
in response to message #5 by Gerson W. Barbosa

Or,

```
00 { 25-Byte Prgm }
01 LBL"DER"
02 1E-9
03 COMPLEX
04 ASTO L
05 XEQ IND ST L
06 COMPLEX
07 1E9
08 *
09 END
```

It does not need a predefined alpha label for the function
It does not use registers, numbered or other. Just the alpha register to pass the function name

Cheers, Werner

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #7 Posted by **Gerson W. Barbosa** on 12 Nov 2011, 10:44 a.m.,*
in response to message #6 by Werner

Thanks for the tip, Werner. It doesn't give correct results when |x| >= 1e4 for my example (LN), unless a slight modification is made:

```
00 { 27-Byte Prgm }
01 LBL "DER"
02 1E-9
03 RCL* ST Y
04 STO 00
05 COMPLEX
06 ASTO ST L
07 XEQ IND ST L
08 COMPLEX
09 RCL/ 00
10 END
```

Cheers,

Gerson.

Edited: 12 Nov 2011, 10:48 a.m.

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #8 Posted by **Dieter** on 12 Nov 2011, 12:13 p.m.,*
in response to message #7 by Gerson W. Barbosa

What about x = 0? This currently means that h = 0 as well, throwing a division by zero error. Setting h = 1E-9 in this case is not really an option. ;-)

Dieter

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #9 Posted by **Gerson W. Barbosa** on 12 Nov 2011, 12:29 p.m.,*
in response to message #8 by Dieter

Ooops!

```
00 { 28-Byte Prgm }
01 LBL "DER"
02 1E-9
03 X<Y?
04 RCL* ST Y
...
```

This should handle practical cases. Of course the result is not valid for LN(x) because LN(0) is not defined.

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #10 Posted by **Dieter** on 12 Nov 2011, 12:48 p.m.,*
in response to message #9 by Gerson W. Barbosa

Are negative values no practical cases ?-)
What about x = -1 000 000? This would set h = 1E-9 as well.

Dieter

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #11 Posted by **Gerson W. Barbosa** on 12 Nov 2011, 1:45 p.m.,*
in response to message #10 by Dieter

Back from a nap... which might explain it (don't program when you're sleepy :-)

Back to the drawing board:

```
00 { 29-Byte Prgm }
01 LBL "DER"
02 1E-9
03 RCL* ST Y
04 X=0?
05 LASTX
06 STO 00
...
```

Or on the HP-15C:

```
001- f LBL B
002-   ENTER
003-   ENTER
004-   EEX
005-   CHS
006-   9
007-   *
008- g x=0
```

```
009- g LSTx
010-   STO 0
011- f Re<>Im
012-   +
013-   GSB E
014- f Re<>Im
015-   RCL/ 0
016- g CF 8       ; leave complex mode
017- g RTN


018- f LBL E
019- g LN
020- g RTN


 keystrokes       display


EEX 90 GSB B -> 1.000000-90
       GSB B -> 1.000000 90
```

Any idea to save a step or two?

Edited: 12 Nov 2011, 7:57 p.m.

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #12 Posted by **Dieter** on 13 Nov 2011, 8:00 a.m.,*
in response to message #11 by Gerson W. Barbosa

Gerson,
I think we are discussing a problem that actually does not exist. ;-) There is no need to make sure that h is a certain fraction of x. And x=0 does not require any special handling either. The method should work for essentially any h, regardless of the magnitude of x. Consider your ln(x) example. It doesn't matter if x is 1 or 50000 or 10^9 - in all cases h = 1E-9 will do. Remember, h is *not* added to x, it's just used for its imaginary part.

Example:

```
   x = 1234567890
   h = 1E-9
   ln (1234567890 + 1E-9 i)
     = 2,9339868592 + 8,10000007371E-19 i
So
```

```
f' = 8,10000007371E-19 / 1E-9
   = 8,10000007371E-10
   = 1/x
```

It even works down to h = x * 1E-99 resp. x * 1E-499.
Or, in other words: your very first solution was fine already. ;-)

Dieter

Edited: 13 Nov 2011, 8:09 a.m.

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #13 Posted by **Gerson W. Barbosa** on 13 Nov 2011, 8:44 a.m.,*
in response to message #12 by Dieter

Dieter,

I am just trying to avoid the results below. The second program gets them all right.

```
   keystrokes          display


.1 GSB A             10.00000000
.01 GSB A            100.0000000
.001 GSB A           1000.000000
.00001 GSB A         10000.00000
EEX CHS 5 GSB A      99999.99967
EEX CHS 6 GSB A      999999.9967
EEX CHS 7 GSB A      9999999.987
EEX CHS 8 GSB A      99668652.49
EEX CHS 9 GSB A      785398163.4
EEX CHS 10 GSB A     1471127674.
EEX CHS 20 GSB A     1570796327.
EEX CHS 30 GSB A     1570796327.
EEX CHS 40 GSB A     1570796327.
...
```

Regards,

Gerson.

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #14 Posted by **Dieter** on 13 Nov 2011, 9:06 a.m.,*
in response to message #13 by Gerson W. Barbosa

I see, so the problem arises for values close to zero. On the other hand you previously said *"...it doesn't give correct results when |x| >= 1e4 for my example (LN)"*, so I took a closer look at *large* values. #-)

This leads to the question if there is a clever way to determine a suitable h that's small enough to ensure sufficient precision as well as large enough to prevent underflow. But I fear this cannot be done in 11 steps or less. ;-)

Dieter

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #15 Posted by **Gerson W. Barbosa** on 13 Nov 2011, 9:44 a.m.,*
in response to message #14 by Dieter

> Quote:
>
> But I fear this cannot be done in 11 steps or less. ;-)

Agreed! The steps 11 and 12 in the program above can be replaced by [f] [I] per Thomas Klemm's suggestion. Now it's only 16 steps long and decounting :-)

Gerson.

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #16 Posted by **Thomas Klemm** on 13 Nov 2011, 10:47 a.m.,*
in response to message #14 by Dieter

From The Complex-Step Derivative Approximation:

> Quote:
>
> Taking the imaginary parts of both sides of this Taylor series expansion (7) and dividing it by h yields:

Invalid Equation

Similarly, for the truncation error of the derivative estimate to vanish, we require that:

Invalid Equation

For this example we have:

$$f(x) = \log(x)$$

The third derivative is:

Invalid Equation

This leads us to:

$$h^2 < \varepsilon|3x^2|$$

or

$$h < \sqrt{3\varepsilon}|x|$$

So with a relative working precision of 1e-10 we should be happy with h = 1e-5 |x|. It turns out this isn't exactly enough but it seems h = 1e-6 |x| is.

> Quote:
>
> This leads to the question if there is a clever way to determine a suitable h that's small enough to ensure sufficient precision as well as large enough to prevent underflow.

I doubt that this can be achieved in general. OTOH we probably run only into problems in the neighborhood of singularities but Valentin stated in requirement 5:

> Quote:
>
> (except for the usual cases of particular troublesome points or ranges, etc, of course)

Therefore I guess we can live with h = 1e-9 in most of the cases.

Cheers
Thomas

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #17 Posted by **Werner** on 13 Nov 2011, 9:17 a.m.,*
in response to message #12 by Dieter

I thought so too, at first, but: take f(x)= LN(x) and x = 1e-10 then (with h=1e-9):

```
f(1e-10,1e-9) = (-20.7182906715,1.4711276743)
```

and the calculated derivative is 1471127674.3 instead of 1e10.
Making h smaller will work, so you might think, let's take the smallest possible h=1e-499. But then,
Taking h=1e-499 returns f'(1e499)=0, as underflow occurs in the calculation of LN(x,h).
Scaling h by x, as Gerson did, works for the whole range.

Cheers, Werner

OT: Google et al. [Re: HP-15C (& LE!) 11-11-11 Mini-Challenge]

*Message #18 Posted by **Thomas Radtke** on 12 Nov 2011, 11:16 a.m.,*
in response to message #5 by Gerson W. Barbosa

> Quote:
>
> That's what Wikipedia and Google are for :-)

I still smile about it, too, but while I think wikipedia will serve us well now and in the long run, I'm waiting for the day Google renders itself useless for anything off mainstream. It already act at its own authority too much.

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #19 Posted by **Marcus von Cube, Germany** on 12 Nov 2011, 10:33 a.m.,*
in response to message #4 by Thomas Radtke

This may help to shorten the f' function in WP 34S which is implemented in user code. Pauli?

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #20 Posted by **Paul Dale** on 13 Nov 2011, 2:01 a.m.,*
in response to message #19 by Marcus von Cube, Germany

This method isn't really suitable for the 34s. It requires a complex evaluation of the user supplied function which isn't seamless unfortunately.

The 34s differentiation routine uses 4, 6 and 10 point estimates and returns the 10 point one if all evaluations succeed.

I've not attempted this challenge on the 34S but the code is nice and short:

```
LBL A
f'(x) D
RTN
```

Using label D instead of E. Shorter again to not bother coding this and just enter f'(x) D :-)

- Pauli

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #21 Posted by **Gerson W. Barbosa** on 12 Nov 2011, 8:25 a.m.,*
in response to message #3 by peacecalc

It does work. That was Valentin's idea, though. I've only figured out which it was, I think.

Regards,

Gerson.

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #22 Posted by **peacecalc** on 12 Nov 2011, 9:36 a.m.,*
in response to message #21 by Gerson W. Barbosa

Hello Gerson,

I beg your pardon Gerson, my lessons in complex functional analysis are 30 years left. I forgot them.

Now, it's a good reason for me repeating that stuff, because it's new for me to use result from this for numerical purposes.

sincerely peacecalc

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #23 Posted by **Gerson W. Barbosa** on 12 Nov 2011, 9:48 a.m.,*
in response to message #22 by peacecalc

Hello,

Same here. As I said, I've only implemented the formula in the Wikipedia article above. The following paper provides a detailed explanation of the method:

The Complex-Step Derivative Approximation

Regards,

Gerson.

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #24 Posted by **Thomas Klemm** on 13 Nov 2011, 8:00 a.m.,*
in response to message #23 by Gerson W. Barbosa

It appears that Valentin took the first example from this paper:

2.2 A Simple Numerical Example

Since the complex-step approximation does not involve a difference operation, we can choose extremely small step sizes with no loss of accuracy.

To illustrate this point, consider the following analytic function:

$$f(x) = \frac{e^x}{\sqrt{\sin^3 x + \cos^3 x}}. \qquad (12)$$

The exact derivative at $x = 1.5$ is computed analytically to 16 digits and then compared to the results given by the complex-step formula (6) and the forward and central finite-difference approximations.

Kind regards
Thomas

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #25 Posted by **Thomas Klemm** on 13 Nov 2011, 7:35 a.m.,*
in response to message #2 by Gerson W. Barbosa

Saved one step:

```
001 - 42,21,11   LBL A
002 -       26   EEX
003 -        9   9
004 -       16   CHS
005 -    44  0   STO 0
006 -   42  25   I
007 -   32  15   GSB E
008 -   42  30   Re<>Im
009 - 45,10, 0   RCL ÷ 0
010 -   43  32   RTN
```

Nice solution!

Cheers
Thomas

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #26 Posted by **Gerson W. Barbosa** on 13 Nov 2011, 9:24 a.m.,*
in response to message #25 by Thomas Klemm

> Quote:
>
> Saved one step

Great!

This should be closer to Valentin's original solution. I guess his solution includes

```
10- 53, 5, 8  CF 8
```

This is required to get rid of the imaginary part. Try, for instance, squaring f'(pi/6) when f(x) = sin(x). The correct result is 3/4 not 1/2 + sqrt(3)/2*i. When I saw your listing, I imagined it would not work if the complex mode had not been previously set. I had tried a similar program, except that I included SF 8 in the beginning (I didn't remember [f] [I] activates the complex mode, like Re<>Im does).

Cheers,

Gerson.

Edited: 13 Nov 2011, 9:32 a.m.

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #27 Posted by **J-F Garnier** on 13 Nov 2011, 1:36 p.m.,*
in response to message #25 by Thomas Klemm

I learnt something today, thanks to Valentin.

For the fun, here is a solution of the last two problems for the HP32SII. At least I found a use of the HP32SII complex mode!

```
LBL A
1E-9
x<>y
XEQ E
x<>y
1E-9
/
RTN


LBL E  ( 2nd problem )
0
2.1
CMPLX*
CMPLXe^x
0
7
CMPLX*
RTN


LBL E  ( 3rd problem )
0
```

```
ENTER
CMPLX+
CMPLXSIN
Rv
Rv
CMPLXe^x
CMPLX/
RTN
```

I didn't succeed to write it on the HP35s...

Re: HP-15C (& LE!) 11-11-11 Mini-Challenge

*Message #28 Posted by **Gerson W. Barbosa** on 13 Nov 2011, 6:17 p.m.,*
in response to message #27 by J-F Garnier

The first problem is possible as well:

```
A01 LBL A
A02 1E-9
A03 STO B
A04 x<>y
A05 XEQ E
A06 x<>y
A07 RCL/ B
A08 RTN


E01 LBL E
E02 STO A
E03 RCL B
E04 x<>y
E05 CMPLXCOS
E06 RCL B
E07 RCL A
E08 CMPLXSIN
E09 CMPLX+
E10 0
E11 3
E12 CMPLX*
E13 3
E14 RCL* B
E15 3
E16 RCL* A
```

```
E17 CMPLXCOS
E18 CMPLX+
E19 3
E20 RCL* B
E21 3
E22 RCL* A
E23 CMPLXSIN
E24 CMPLX-
E25 0
E26 4
E27 CMPLX/
E28 2
E29 RCL* B
E30 2
E31 RCL* A
E32 CMPLXe^x
E33 CMPLX/
E34 CMPLX1/x
E35 0
E36 ENTER
E37 0.5
E38 CMPLXy^x
E39 RTN


1.5 XEQ A -> 4.05342789391
```

Better use the original expression and save the intermediate results to registers. Or use the HP-28S symbolic differentiator instead :-)

Regards,

Gerson.

HP-15C (& LE!) 11-11-11 Mini-Challenge ORIGINAL SOLUTION & COMMENTS

*Message #29 Posted by **Valentin Albillo** on 14 Nov 2011, 8:12 a.m.,*
in response to message #1 by Valentin Albillo

Hi, all:

Thanks for your interest in my HP-15C (&LE!) Mini-challenge, I'm glad many of you enjoyed it and produced a number of correct solutions, some of which virtually duplicated my original one, as well as producing solutions for other models. That's the spirit of these challenges and mini-challenges and I thank you for joining in.

As for my original solution, this is it:

```
01  LBL A
02  EEX
03   6
04  CHS
05  STO I
06   I
07  GSB E
08  Re<>Im
09  CF 8
10  RCL/ I
11  RTN
```

which is 11 steps and meets all 5 requirements. I prefer to use h=1E-6 instead of smaller values and I think that CF 8 must be included lest you'll leave the machine in complex mode, which would be utterly unexpected to the user as both the input and the output are assumed and expected to be real.

Partially relaxing condition (1), i.e., supressing LBL and RTN, would save two steps, while relaxing condition (4), i.e., pre-storing 1E-6 in a register, would save three additional steps, leaving the grand total at a mere 6 steps or so. An additional step could be saved by supressing CF 8 but as stated above, I think this would seriously detract from the usability.

As for the register used to store the constant, I prefer using permanent register RI instead of R0 or R1 (also permanent) though, as the constant is less than one, the best solution is using register RAN#, which frees RI for indexing purposes and R0-R1 for general use or matrix operations. However, as there's no RCL\ RAN# instruction this means an additional step to perform the division.

Finally, I knew that for X arguments near 0 accuracy would get worse and so require extra steps to cater for that case, thus this is why I included the exception at condition (5) in order to keep it short and sweet. After all, this was intended as a Mini-challenge, not a full-fledged article on numerical derivation complete with industrial-strength code ! ... :D

When I was busy concocting this mini-challenge, I first implemented it for the HP-71B, where the code is as simple as

```
IMPT(FNF((X,H)))/H
```

where FNF is the user-defined function f(x) and H is a suitably small constat (1E-7, say). I conducted some experiments to see which H would be best, for instance:

```
10 DEF FNF(X)=5*COS(10*X)+((X-2)*X-6)*X+10
20 DEF FND(X)=-50*SIN(10*X)+3*X*X-4*X-6
30 FOR I=1 TO 9 @ H=10^(-I) @ DISP I;IMPT(FNF((1,H)))/H,FND(1) @ NEXT I


>RUN
```

```
 1  24.9567129442     20.2010555444
 2  20.24631331       20.2010555444
 3  20.2015078976     20.2010555444
 4  20.201060068      20.2010555444
 5  20.2010555897     20.2010555444
 6  20.2010555449     20.2010555444
 7  20.2010555444     20.2010555444  << H=1E-7 first nails the exact derivative
 8  20.2010555444     20.2010555444
 9  20.2010555444     20.2010555444
```

and in virtually all cases H = 1E-7 was optimal (except for X near 0, of course, where a smaller H is needed). For the HP-15C, which only carries 10 (13) digits instead of 12 (15), H=1E-6 does the trick.

That's all. I've got (hopefully) interesting ideas for some dozen challenges and mini-challenges as well as for another dozen full-fledged articles but regrettably next to no or very little time to write them down & post them hera, and actually no suitable place to publish the long articles so it might be a while till I make another lengthy contribution ... :(

Thanks again for your interest and best regards from V.

[Return to Index | Top of Index]

Go back to the main exhibit hall