HP Forum Archive 14

[Return to Index | Top of Index]

Re: Feigenbaum number

*Message #1 Posted by **Valentin Albillo** on 3 June 2004, 6:12 a.m.*

Hi all:

A few days ago, Csaba posted:

"what was the program, what used by Feigenbaum originally(!), to calculate this number on his HP65? [...] The greatest problem, how to isolate the bifurcation points on the diagram..."

Actually, it's quite simple. Have a look at this small 4-line, 174-byte *ad-hoc* program I've written for the HP-71B (which can be very easily ported to most any other HP calc, Feigenbaum's own HP-65 included):

```
10 DESTROY ALL @ DELAY 0,0 @ F=3.2 @ M=0 @ N=1
20 FOR K=2 TO 9 @ P=N+(N-M)/F @ X=FNROOT(P,P,FNB(2^K,FVAR)) @ F=(N-M)/(X-N)
30 PRINT "K=";K;"F=";F @ M=N @ N=X @ NEXT K @ END
40 DEF FNB(K,A) @ B=0 @ FOR I=1 TO K @ B=A-B*B @ NEXT I @ FNB=B @ END DEF
```

When run, it will perform 9 iterations of a procedure designed to calculate an increasingly refined value of Feigenbaum's constant, starting from a rough initial approximation (3.2). Just press [RUN] and you'll get these results (K = #iteration, F = estimated value for Feigenbaum's constant):

```
    K           F
-------------------------
    2      3.21851142201
    3      4.38567759894
    4      4.60094927695
    5      4.65513048088
    6      4.66611150124
    7      4.66855134603
    8      4.66905077160
    9      4.66921465563
```

After some 200 seconds, the last result at the end of the 9th iteration is **F = 4.66921+**, correct to 6 digits within 1 ulp (err = 0.00001+). For comparison purposes, the value of Feigenbaum's constant is *4.66920 16091029906718+*.

Within the constraints of HP-71B's 12-digit precision and speed, this is more or less the best result you can get. A further 10th iteration would take much, much longer for no real improvement, as the running time grows exponentially with every iteration (there are a number of easy ways of speeding my 4-liner, of course, but I prefer to keep it at its simplest for didactic purposes, if nothing else).

By the time it reaches the 9th iteration, it is finding a root of a *512th-degree* (!) polynomial, which takes a lot of time and incurs in growing cumulative rounding errors when evaluating it a number of times as part of the search for its root. A 10th iteration would mean solving (i.e.: evaluating repeatedly) a 1,024th-degree (!!) polynomial, and cumulative rounding errors mean 6 correct digits is the best you can do in 12-digit arithmetic.

However, if you have access to an arbitrary precision package either for a PC (Maple, Mathematica, etc) or even your HP-71B, you can easily get more correct digits using this algorithm, albeit at a heavy time penalty.

Best regards from V.

Edited: 3 June 2004, 6:17 a.m.

[OT] Re: Feigenbaum number

*Message #2 Posted by **Olivier De Smet** on 3 June 2004, 8:57 a.m.,*
in response to message #1 by Valentin Albillo

Hi Valentin,

Btw I search actually as much as possible information on the HP8x series (processor, memory controller, ...).

I have an HP85A, an HP86B some 91{3/5}3 HD and a 9121 floppy. To use the 91xx I need an EMS rom, but can't find one. So I think about dis-assembling the rom, patching to allowed relocation and re-assemble sources to allow them to be loaded as binary. I think it's feasible but I need some more information on the 8x series. Actually I have a python programm who can de-assemble an HP8x binary in a quite usefull form. The assembler will be easier to write.

For example after searching the web I still have some questions:

- What does SAD and PAD opcodes exactly ? (octal 232 and 237)

- Is there an opcode for octal 326 and 336 ?

- What are the vectors for interrupts in the capricorn processor ?

- How the rom and ram paging works in an HP86 (preciselly)?

- I partially understood the structure of a binary program, but I miss informations for the rom structure ...

I get some informations from HP journals, from HP75C programs and from assemblers globals (HP85 and HP86)

Thanks in advance if you could help me.

Olivier

Re: [OT] Re: Feigenbaum number

*Message #3 Posted by **Valentin Albillo** on 3 June 2004, 9:34 a.m.,*
in response to message #2 by Olivier De Smet

Hi, Olivier:

Olivier wrote:

"Btw I search actually as much as possible information on the HP8x series (processor, memory controller, ...)."

I did read your request the first time you posted it. Unfortunately, it's been more than 15-18 years since I last had anything to do with an HP86 and all materials I once owned (manuals, ROMs, listings, binaries, etc) are now irretrievably missing. And, regrettably, my memory is of no help for such accurate details after so long a time lapse.

Anyway, should I find any materials of information, by sheer luck, I'll let you know promptly.

Best of lucks in your plausible search and

Best regards from Valentin Albillo

Feigenbaum for the HP-15C [LONG]

*Message #4 Posted by **Valentin Albillo** on 4 June 2004, 6:48 a.m.,*
in response to message #1 by Valentin Albillo

Hi,

Just to prove the point, here's a conversion of my 4-liner HP-71B program to the HP-15C !

Despite the obvious (and large) differences in language level (low-level RPN vs. high-level BASIC), numerical precision (10-digit vs. 12-digit) and above all, speed (15-30x slower), the HP-15C is perfectly up to the task, delivering Feigenbaum's constant correct to 4 digits in 18 min., which fares well compared to the 71B results (6 digits in 3.3 min.).

Also, the HP-15C program is much more compact, at 60+ bytes (47 steps) versus 71B's 174 bytes (4 multi-statement lines), i.e. just 1/3rd the size (RPN strikes again!). Listing follows with a few pertinent comments:

```
01  LBL A      14  LBL 0      25  RCL- 1     36  GTO 0
02  MATRIX 1   15    2        26  RCL 4      37  RTN
03  CLX        16  STOx 5     27  RCL- 0     38  LBL 1
04  STO 1      17  RCL 0      28    /        39  RCL 5
05   7         18  RCL- 1     29  STO 2      40  STO 6
06  STO 3      19  RCL/ 2     30   R/S       41  CLX
07   2         20  RCL+ 0     31  RCL 0      42  LBL 2
08  STO 5      21  ENTER      32  STO 1      43   X^2
09  3.2        22  SOLVE 1    33  RCL 4      44    -
12  STO 2      23  STO 4      34  STO 0      45  DSE 6
13  FIX 6      24  RCL 0      35  DSE 3      46  GTO 2
                                              47   RND
```

To use, simply press **RUN** with no inputs. The program will compute and display the increasingly accurate value of Feigenbaum's constant for each iteration, stopping for you to see it. Press **R/S** to go on with the next iteration. After completing all 7 iterations, the program will stop displaying the last computed superstable *a* value, **a = 1.401146+**, which is the SOLVEd root of the corresponding 256th-degree polynomial.

Results:

```
Iteration   F value   Iteration time
-------------------------------------
    1      3.218510        17 sec.
    2      4.385687        50
    3      4.600945        65
    4      4.655086        83
    5      4.666256       162
    6      4.667830       234
    7      4.669803       477

    Last result is 4.669+, accurate to 4 digits within 1 ulp.
```

Notes:

- As the HP-15C uses 10-digit precision instead of HP-71B's 12-digit, we can't get a result accurate to more than 4-5 significant digits, so just 7 iterations will suffice. Besides, we don't need to find the high-degree polynomials' roots to full 10-digit precision, thus steps **13 FIX 6** and **47 RND** provide early termination for **22 SOLVE 1**, saving *much time* at no cost in final accuracy.

- The high-degree polynomial that is being SOLVEd is computed at steps 38-47. Notice the very tight loop 42-46 which is once again a *superb* practical demonstration of classic RPN's stack capabilities at its best: no matter how high the degree of the polynomial, the repeated subtraction at step 44 *never* exhausts the provision of *a* values, thanks to top-level replication, where T is replicated continuously as the stack drops and drops. If literally coded in RPL this loop would collapse the stack in no time !

- The ever bigger polynomials' roots are found using the powerful **SOLVE** instruction at step 22. In a running program, SOLVE behaves like a test: if a root is found, the next step is executed, else it is skipped. Normally this must be taken into account, but for this particular program, the root always exists and SOLVE always finds it for the number of iterations considered, so there's no need to code the possibility of it failing.

- Last but not least, notice how every advanced feature of the HP-15C instruction set helps to reduce step count and increase speed, from the matrix instruction at step 02, to recall arithmetic everywhere, to being able to use any numbered register for loop indexing purposes.

In conclusion, I hope you'll agree that the mere fact that this kind of computation is feasible in the HP-15C in reasonable times and with such a small, simple program, is but another proof of the outstanding quality of this wonderful machine. As "Moulin Rouge!"'s Zidler himself would say on behalf of the HP-15C: "A magnificent, opulent, tremendous, stupendous, gargantuan success"

Best regards from V.

Edited: 4 June 2004, 7:55 a.m.

Re: Feigenbaum number

*Message #5 Posted by **helge gabert** on 4 June 2004, 11:38 p.m.,*
in response to message #1 by Valentin Albillo

I ported Valentin's Feigenbaum program to the HP49G+, and it runs a little faster (as it should, compared to the 71B), i.e., 9 iterations in 1 min 35 sec.

However, it seems to be less accurate (more prone to rounding error).

After 7 iterations, F= 4.66854950126, after 8 iterations, F= 4.6690698451, after 9 iterations, F= 4.66915720392. (only accurate to 4 digits)!

Even after 10 iterations, F=4.66903301492. The CAS mode (approximate or exact, Real or Complex) doesn't matter.

Strange, I thought the 49g+ was also using 12 digit precision internally, so the results should be identitical to those obtained on the 71B.

Here is the 49G+ program:

FEIG: << CLLCD TIME 'T' STO 3.2 'F' STO 0. 'M' STO 1. 'N' STO 2. 9. FOR k N M - F / N + 'P' STO << 2. k ^ 'X' FNB EVAL >> 'X' P ROOT 'X' STO N M - X N - / 'F' STO "K = " k + 1. DISP "F = " F + 2. DISP "Time = " TIME T HMS- 100. * + 3. DISP N 'M' STO X 'N' STO NEXT 7. FREEZE >>

FNB: << -> K A << 0 1 K START DUPDUP * NEG A + NIP NEXT >> >>

Any suggestions on how to improve the accuracy?

Re: :)

*Message #6 Posted by **Tizedes Csaba [Hungary]** on 7 June 2004, 4:23 p.m.,*
in response to message #1 by Valentin Albillo

Thank you!

I think I changing my mind about programming...

Csaba

[Return to Index | Top of Index]

Go back to the main exhibit hall