[VA] SRC #012b - Then and Now: Root

8th November, 2022, 00:14 **Post: #1**

Valentin Albillo
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

[VA] SRC #012b - Then and Now: Root

Hi, all,

After the many excellent solutions & comments posted for *Problem 1* and once the ***7,100** views* mark has been met and exceeded, now's the time for the 2nd part of my new ***SRC #012 - Then an Now***, where I'll demonstrate that advanced vintage *HP* calcs which were great problem-solvers back **THEN** in the 80's (some 40 years ago !) are **NOW** still perfectly capable of solving recently-proposed tricky problems intended to be tackled by using fast modern *2020*-era personal computers, never mind slow ancient pocket calcs.

In the following weeks I'm proposing **six** increasingly harder such problems for you to try and solve while respectfully abiding by the following ***mandatory rules*** summarized here:

> You must try and solve them using **EXCLUSIVELY** your preferred **VINTAGE HP CALCULATORS** (physical or virtual,) coding in either **RPN** (including **HP-41** *microcode*), **RPL** (any variant existing at the time, including *SysRPL*) or **HP-71B** languages (including *BASIC*, *FORTH* and *Assembler*) **AND NOTHING ELSE**: NO CAS/XCAS, MATLAB, MATHEMATICA, EXCEL, C/C++/C#, PYTHON or the like, AND **NO LENGTHY MATH** SESSIONS/LECTURES. Also, **NO CODE PANELS**, please !
>
> On the positive side, you ***may*** use any official/well-known modules, pacs or libraries which were available at the time, such as the **Math Pac** and **JPC ROM** for the **HP-71B**, the **Advantage Module**, **PPC ROM** and *Extended Memory* for the **HP-41**, and assorted *libraries* for the ***RPL*** models, to name a few.

This ***Problem 2*** deals with *polynomial roots* with a bang, namely:

Problem 2: Root

> Write a program to find the *minimum absolute value* among the roots of the following polynomial:
>
> $$P(x) = 2 + 3x + 5x^2 + 7x^3 + 11x^4 + 13x^5 + \ldots + 104743\,x^{10{,}000}$$
>
> whose coefficients are the *prime numbers* in order: *2, 3, 5, 7, 11, 13, ... , 104743.*

Your program should have no inputs and must output the asked value and automatically end. You should strive for *10-12 correct digits* (gave or take a few *ulp*) depending on your *HP* model, and the faster the running time the better. Also, you must justify in your comments the soundness of your approach, not *"just trying"* or relying on luck. 🙂

Some useful advice is to try and find the correct balance between letting the program do all the work *(i.e. sheer brute force, which could potentially take far too much RAM and running time)* with no help from you, or else use a little bit of insight to help significantly speed up the process. Your choice.

> (As an aside, I wonder if *any* (or even all !) forum members who successfully posted correct solutions for *Problem 1* will be able to follow suit and solve this *Problem 2* as well, for a perfect ***2 for 2*** score ! 😁)

If I see interest I'll post in a few days my *original solution* for the **HP-71B**, a *5-liner* which computes the required absolute value relatively quickly and accurately (it can be done in just *4 lines* albeit at a significantly slower speed).

In the meantime, let's see your very own clever solutions **AND** *remember the above rules*, please.
V.

8th November, 2022, 13:17 **Post: #2**

Werner
Senior Member

Posts: 767
Joined: Dec 2013

RE: [VA] SRC #012b - Then and Now: Root

(second try..)
Well I do have a result now, my previous one was wrong.
It's on Free42, which appears to be permitted (I understood that wrongly), and while I can make it run on a 42S, it would take too long.
Also, I have no idea if my result is the smallest. I'll have to do some thinking instead of mindless coding ;-)
Werner

- the Free42 solution precalculates the matrix of 10001 coefficients - an easy adaptation of a prime number listing routine I wrote years ago. On a 42S, this is not feasible, so the coefficients would have to be calculated over and over - hence the probably long execution time there. (it takes about 4 hours to generate the coefficients once)
- then, a routine to evaluate a polynomial stored as a row vector, also from my archives (really simple)
- and at last, a 'specific' solver routine, which, I think, was inspired by a post of Valentin himself, long ago. 7 iterations are needed to converge to a result, so, no, I'm not going to run it on my 42S.
When I'm reasonably sure the result is correct I'll post some code.

EMAIL PM FIND QUOTE REPORT

9th November, 2022, 10:43 (This post was last modified: 9th November, 2022 16:12 by J-F Garnier.) **Post: #3**

J-F Garnier
Senior Member

Posts: 790
Joined: Dec 2013

RE: [VA] SRC #012b - Then and Now: Root

I was not sure to fully understand the problem. Do you have to search for the smallest (in absolute value) root in the *real* or *complex* domain?
When we are speaking of roots of a polynomial, we are often referring to its complex roots.
But when I read 'absolute value', I tend to link it to a real number, even if the ABS function is often used in HP programming languages to calculate the norm or modulus of a complex number.

So I will assume that we are looking for real roots.
My first thoughts: a real root must be negative, and greater than -1 since the polynomial quickly takes very large values for |X|>1.

My first analysis (using HP calculators), cutting the polynomial to the first 10-100 terms seemed to indicate me that there is no real root.
So if there are real roots, they are close to -1, otherwise the higher X^i terms would be negligible and I would have found the root with the truncated polynomial.
I will try to build a program to support this, but I'm not sure I will be able to propose something useful, even on a fast emulator.

J-F

EMAIL PM WWW FIND QUOTE REPORT

9th November, 2022, 11:02 (This post was last modified: 10th November, 2022 09:04 by Werner.) **Post: #4**

Werner
Senior Member

Posts: 767
Joined: Dec 2013

RE: [VA] SRC #012b - Then and Now: Root

(error corrected in PX)
There are indeed no real roots.
I simply built the polynomial, stored its coefficients in a matrix P, and used my complex solver (explanation and source code here) with starting value 0 to find
(-0.645758096347,0.483177676217), abs value 0.806513599261

here's the polynomial generator routine:

```
00 { 139-Byte Prgm }
01▸LBL "PLST"
02 1
03 10001
04 NEWMAT
05 EDIT
06 1.001
07 STO 00
08 SIGN
```

```
09 STO IND 00
10 2
11 XEQ 14
12 3
13 XEQ 14
14 +
15 XEQ 14
16▸LBL 02
17 2
18 XEQ 03
19 FS? 77
20 GTO 00
21 4
22 XEQ 03
23 FC? 77
24 GTO 02
25▸LBL 00
26 RCLEL
27 EXITALL
28 STO "P"
29 RTN
30▸LBL 03
31 +
32 RCL 00
33 X<>Y
34▸LBL 04 @ ( loop over all stored primes )
35 RCL IND ST Y
36 RCL ST Y
37▸LBL 05 @ ( GCD )
38 MOD
39 LASTX
40 X<>Y
41 X>0?
42 GTO 05
43 +
44 R↓
45 DSE ST T @ ( test whether GCD=1 )
46 RTN
47 ISG ST Y
48 GTO 04
@ ( either the number is prime or it is a perfect square of one )
49 ENTER
50 SQRT
51 IP
52 X^2
53 X<>Y
54 X>Y? @ ( number is prime )
55 GTO 14
56 SQRT @ ( number is a perfect square )
57 STO IND ST Z @ ( store it in next register, just in case )
58 DSE ST Z @ ( will always skip )
59 X>0? @ ( nop )
60 RCL IND ST Z
61 ×
62 SIGN @ ( see if product is too large )
63 LASTX
64 STO+ ST Y
65 X=Y?
66 GTO 00
67 STO IND ST T @ ( if not, save it )
68 RCL ST Z
69 RTN
70▸LBL 00 @ ( else increase array )
71 1ᴇ-3
72 STO+ 00
73 R^
74 RTN
75▸LBL 14 @ ( separate label to easily change )
76 →
77 END
```

and the polynomial evaluator (coefficients [a0 a1 .. an] so Horner scheme has to start at the end)

```
00 { 25-Byte Prgm }
01▸LBL "PX" @ evaluate polynomial, matrix indexed at (1,1)
02 ENTER
03 ENTER
04 ENTER
05 CLX
06 J- @ start from the end
07▸LBL 02
08 ×
09 RCLEL
10 +
11 J-
12 FC? 77
13 GTO 02
14 J+ @ back to (1,1)
15 END
```

There is, however, a much better way, one that singles out the smallest root (thanks Albert!). Working on that ;-)
Cheers, Werner

EMAIL PM FIND QUOTE REPORT

9th November, 2022, 12:19 (This post was last modified: 9th November, 2022 18:00 by C.Ret.) **Post: #5**

C.Ret
Member

Posts: 223
Joined: Dec 2013

RE: [VA] SRC #012b - Then and Now: Root

Ah! Ah!

It is with great pleasure and impatience that I discover this new stage of the challenge!

I rub shoulders with it, but I'm worried. The parity of the polynomial is a problem for me as well as the slowness of my HP-71B. Couldn't things be more complicated than they seem?

As I'm not very proud of my performance and the capabilities of my HP-71B (unless it's the other way around?) and in order not to waste too much time developing my solution, I decided to tackle a much simpler polynomial:

$$P_9(x) = 2 + 3x + 5x^2 + 7x^3 + 11x^4 + 13x^5 + 17x^6 + 19x^7 + 23x^8 + 29x^9$$

This polynomial has same resemblances with the polynomial ***P*** which interests us but has the enormous advantage of being infinitely shorter.

Please note that the derivate of P_9 is easily determinate as:
$$P_9'(x) = 3 + 2\cdot 5x + 3\cdot 7x^2 + 4\cdot 11x^3 + 5\cdot 13x^4 + 6\cdot 17x^5 + 7\cdot 19x^6 + 8\cdot 23x^7 + 9\cdot 29x^8$$

This allows me to code faster; Using Horner's scheme, Newton's root finding method and the prime number facility built in the JPC ROM cartridge, I manage to compose a code adapted to my HP-71B which in a few tens of seconds gives me the absolute value of the only real solution:

```
10  DESTROY  ALL                             !!  SRC#012b V.A. challenge (v.9)
  @  REAL  D,E,K,P,X,Y
  @  X=-2/3  @  E=1.E-12                       !   Initial guess and accuracy
20  REPEAT                                     !!  Computation of Y=P(X) and D=P'(X)
  @        P=29  @  Y=P  @  D=0            !       P  last  prime
30        FOR  K=9  TO  1  STEP  -1           !       Y/D  computation  loop
  @             D=D*X+K*P
  @             P=FPRIM(P-1,2)                 !       FPRIM(P-1,2)  return  previous  prime
  @             Y=Y*X+P
  @        NEXT  K
40        X=X-Y/D                             !   Newton's method X(n+1) = X(n) - P(x)/P'(x)
  @  UNTIL  ABS(Y/D)<E                                                !
  @  DISP  ABS(X)  @  END                                             !!  Display absolute value of
root X
```

$|x_9| = .79462$

To increase speed and easiness, values of P_9 and its derivate P_9' are compute together in the same FOR TO NEXT loop.

My next step was to use this exact algorithm for the next polynomial:

$$P_{10}(x) = 2 + 3x + 5x^2 + 7x^3 + 11x^4 + 13x^5 + 17x^6 + 19x^7 + 23x^8 + 29x^9 + 31x^{10}$$

But I suddenly discover what already discover **Werner** and **J.-F. Garnier**; polynomials with only positive coefficients and even degree have no real root. So my previous version of the code didn't converge at all!

No stress, thanks to his Math module, this HP-71B is much more powerful that any basic calculator. It is very easy to modify the program to run the same algorithm but with complex values:

```
10  DESTROY  ALL                              !!  SRC#012b V.A. challenge (v.10)
  @  REAL  E,K,P
  @  COMPLEX  D,X,Y
  @  X=(.3,.7)  @  E=1.E-12                       !    Initial guess and accuracy
20  REPEAT                                     !!  Computation of Y=P(X) and D=P'(X)
  @       P=31  @  Y=P  @  D=0              !       P  last  prime
30       FOR  K=10  TO  1  STEP  -1            !       Y/D  computation  loop
  @           D=D*X+K*P
  @           P=FPRIM(P-1,2)                   !       FPRIM(P-1,2)  return  previous  prime
  @           Y=Y*X+P
  @       NEXT  K
40       X=X-Y/D                             !    Newton's method X(n+1) = X(n) - P(x)/P'(x)
  @  UNTIL  ABS(Y/D)<E                                                    !
  @  DISP  ABS(X)  @  END                                                 !!  Display absolute value of
root X
```

For P_{10}, I easily get the following absolute minimal value (in fact norm of the complex root).

$|z_{10}| = .734576$ since the closest complex root I found for P_{10} near $(0,0)$ is $z_{10} = (0.297370, 0.671694)$

P.S.: Using this complex valued algorithm with P_9 clearly indicate that I have not found the minimum absolute value among the roots since other complex roots exists that potentially have a smaller absolute norm.
For instance: $|z_9| = 0.71297$ due to root $(-.56933, .42917)$ or conjugate.

Now, I need new batteries and a large bunch of time to run my code for the true challenge for the lengthy polynomial! 10'000 that's really huge!
I urgently need to find a way to determine the correct initial guess at first attempt.

References: Here is one of the documents that inspire me and help me develop this exact solution.
Newton’s method, and the fractal it creates that Newton knew nothing about. At 11:16 start the chapter about the 'Fun Facts', as fun as this Valentin Challenge. But you may watch there why I am stuck at the moment and need another good idea to efficiently start a 3-hour computation on the right initial guess!

PM FIND QUOTE REPORT

9th November, 2022, 18:05 (This post was last modified: 9th November, 2022 18:35 by J-F Garnier.) **Post: #6**

J-F Garnier
Senior Member

Posts: 790
Joined: Dec 2013

RE: [VA] SRC #012b - Then and Now: Root

Thanks Werner and C.Ret to put me back on the right track, so we are looking for complex roots.
But I didn't completely loose my time, just see:

> **Werner Wrote:** (9th November, 2022 11:02)
>
> There are indeed no real roots.

> **C.Ret Wrote:** (9th November, 2022 12:19)
>
> But I suddenly discover what already discover **Werner** and **J.-F. Garnier**; polynomials with only positive coefficients and even degree have no real root.

This is wrong! There are (at least) two real roots !

I used brute force, using HTBasic (1999 version) which is a HP BASIC compatible programming environment on PC, that still runs fine on my W10 computer in 2022.
I know this is slightly outside the rules, but not so much, HTBasic is to the HP-71B what Free42 is to the HP-42S: a native (not emulated) compatible language running on PC, with the notable difference is that HTBasic is not free, actually quite expensive being a professional tool.
So the below program, that searches where the polynomial sign changes, could in principle, be run on a HP-71B:

```
100 ! RE-STORE "src12b"
105 ! SRC12B, MoHPC, 8nov2022
110 ! quite 'force brut' approach
115 !
120 OPTION BASE 0
125 DIM P(10000)
130 !
135 N=10000
140 READ P(*) ! READ P() on the 71B
145 !
150 ! find out where the sign changes:
155 Y1=2 ! polynomial value at X=0
160 FOR X=-.01 TO -1.01 STEP -.001
165   GOSUB Evalpoly
170   IF SGN(Y)<>SGN(Y1) THEN
175      PRINT X1,Y1 ! print the interval where sign changes
180      PRINT X,Y
185      PRINT "------"
190   END IF
195   X1=X ! save X,Y for next iteration
200   Y1=Y
205 NEXT X
210 STOP
215 !
225 Evalpoly: ! evaluate polynomial at X, result in Y
230 Y=0
235 FOR I=0 TO N
240   IF (I*LGT(-X))>-300 THEN Y=Y+X^I*P(I)
245 NEXT I
250 RETURN
255 !
290 !
295 DATA 2 , 3 , 5 , 7 , 11 , 13 , 17 , 19 , 23 , 29
300 DATA 31 , 37 , 41 , 43 , 47 , 53 , 59 , 61 , 67 , 71
305 DATA 73 , 79 , 83 , 89 , 97 , 101 , 103 , 107 , 109 , 113
...
5290 DATA 104677 , 104681 , 104683 , 104693 , 104701 , 104707 , 104711 , 104717 , 104723 , 104729
5295 DATA 104743
5300 END

X, polynomial value:
-.996 .9768...
-.997 -5.9593...
------
-.999 -39.8287...
-1 52726
------
```

So there is a real root between -.996 and -.997 and another one between -.999 and -1, in accordance to my guess that real roots (if existing) would be close to -1.

Now, I will go back to the actual question, using the 71B or a HP *calculator* (promised!), but I thought this unexpected result was worth to be reported here.

J-F

EMAIL PM WWW FIND QUOTE REPORT

9th November, 2022, 18:37 (This post was last modified: 9th November, 2022 19:44 by C.Ret.) **Post: #7**

C.Ret
Member

Posts: 223
Joined: Dec 2013

RE: [VA] SRC #012b - Then and Now: Root

J-F Garnier Wrote: (9th November, 2022 18:05)

```
X, polynomial value:
-.996 .9768...
-.997 -5.9593...
------
-.999 -39.8287...
-1 52726
------
```

So there is a real root between -.996 and -.997 and another one between -.999 and -1, in accordance to my guess that real roots (if existing) would be close to -1.

That's of great interest. I am compiling a short HP-71B -Math Pack JPC ROM HP-BASIC code to confirm your observations. This will also indicate me how slow is my device to compute these only 4 values.

But that won't allay my worries; among the 9998 other complex roots, is there not one or the other closer to the point (0,0)?

PM FIND QUOTE REPORT

9th November, 2022, 22:13 **Post: #8**

PeterP
Member

Posts: 172
Joined: Jul 2015

RE: [VA] SRC #012b - Then and Now: Root

C.Ret Wrote: (9th November, 2022 18:37)

So there is a real root between -.996 and -.997 and another one between -.999 and -1, in accordance to my guess that real roots (if existing) would be close to -1.

That is very perplexing. At first I immediately gave up given then large number of coefficients and powers thinking of the solve algorithm we have for the hp41. But then I decided to just play around a little bit, using simpler polynomials.

I made my way in a somewhat (ok, very) manual sleuthing fashion across the first 30 or so polynomials. And then looked at the real roots. In all cases, I was not able to find a real root for a polynomial which ended in a even power. Which peter-intuitively (ie inferior, wrong, intuition) made sense - the last term has the highest exponent and highest multiplier and as such dominates the prior term. And this is true for all pairs of "even, odd" powers. So each pair creates a positive overhang, making it impossible to converge. Or so my logic went. Clearly, contra factum non est discudandum so I need to think more about this (or get some help from the team).

Using the real roots for the odd-ending polynomials, I saw a really nice curve that looked like a logarithmic curve. And low and behold, a logarithmic fit gives some 92% R^2. And would point to a solution slightly bigger than -1. So I need to think about why that pretty smooth and steady fit for the odd-ending polynomials would have a minimum and turn back up. Only thing I can think of is the space between primes is getting larger, creating bigger gaps between the even-odd pairs.

However, all my hopes sank when I realized that we are looking for the smallest absolute solution. And there is no way I could think of (so far) to find the smallest absolute value directly, out of 5000 pairs of conjugated roots for the full polynomial.

I will think how to do that in the smaller polynomial case.

it is also possible that I can find a way to show that the 10 digit precision of the hp41 only goes to a much smaller polynomial and as such I can just solve a much smaller polynomial as an approximation inside the accuracy of the HP chosen.

PM FIND QUOTE REPORT

10th November, 2022, 00:02 (This post was last modified: 10th November, 2022 00:11 by J-F Garnier.) **Post: #9**

J-F Garnier
Senior Member

Posts: 790
Joined: Dec 2013

RE: [VA] SRC #012b - Then and Now: Root

PeterP Wrote: (9th November, 2022 22:13)

I made my way in a somewhat (ok, very) manual sleuthing fashion across the first 30 or so polynomials. And then looked at the real roots. In all cases, I was not able to find a real root for a polynomial which ended in a even power. Which peter-intuitively (ie inferior, wrong, intuition) made sense - the last term has the highest exponent and highest multiplier and as such dominates the prior term. And this is true for all pairs of "even, odd" powers. So each pair creates a positive overhang, making it impossible to converge. Or so my logic went. Clearly, contra factum non est discudandum so I need to think more about this (or get some help from the team).

Let's take the last two terms:
104729 * X^9999 + 104743 *X^10000
for X=-1 you get +14
but for X=-0.999 you get -0.0041
do the sum of 5000 such pairs , and you can get something quite negative, large enough to overcome the constant 2 factor.
Matter of fact, I found real roots for polynomials of 5000 and 10000 terms, but not with only 2000 terms or less.

J-F

EMAIL PM WWW FIND QUOTE REPORT

10th November, 2022, 01:03 **Post: #10**

Fernando del Rey
Junior Member

Posts: 19
Joined: Dec 2013

RE: [VA] SRC #012b - Then and Now: Root

I've written a short program for the HP-71B that will solve the problem for polynomials of degree up to a few hundred, using brute force with the Math ROM's PROOT function.

What I found out is that, for polynomials of degree 149 and beyond, the roots with the minimum absolute value have always an absolute value of:

0.80651359926

I have tried with polynomials of degree up to several hundred always getting the same minimum absolute value, so I am assuming that this value would be the same for the polynomial of degree 10000 in Valentin's OP. But I don't have a hard proof for it, it's just a guess.

I ran my program using Emu71/Win, where it takes just 3 or 4 seconds to get the result for degree 150. On a real 71B it would take several minutes, but I haven't measured it.

EMAIL PM FIND QUOTE REPORT

10th November, 2022, 10:26 (This post was last modified: 10th November, 2022 11:41 by J-F Garnier.) **Post: #11**

J-F Garnier
Senior Member

Posts: 790
Joined: Dec 2013

RE: [VA] SRC #012b - Then and Now: Root

> **Fernando del Rey Wrote:** (10th November, 2022 01:03)
>
> I've written a short program for the HP-71B that will solve the problem for polynomials of degree up to a few hundred, using brute force with the Math ROM's PROOT function.
>
> What I found out is that, for polynomials of degree 149 and beyond, the roots with the minimum absolute value have always an absolute value of:
>
> **0.80651359926**

I did the same yesterday evening with 200 terms, and found the same minimum root as you and Werner already found:
Zmin = (-0.645758096347,0.483177676217) for an abs value of 0.806513599261

My understanding is that the PROOT method, with 200 terms, proves that the root Zmin is indeed the smallest in abs value for the complete 10000th degree polynomial:
- PROOT finds all the roots of the given polynomial (here the 200th degree one), there is no missing one,
- we can be sure that the root Zmin above is also a root of the 10000th degree polynomial, in the limits of the numerical accuracy, because the terms Pn.Zmin^n for n>200 will have a modulus less than about 2E-16 and will not contribute (again in the limits of the numerical accuracy) to the value of the complete polynomial,
- the complete polynomial can not have a smaller root Zx, because it would also be a root (in the numerical accuracy limits...) of the 200th degree polynomial, for the same reason that the Zx^n terms for n>200 would be negligible.

Here is my HP-71B program using both the MATH and JPC ROMs, for reference, I didn't check but it should run in a few hours on a physical HP-71B. The HP48 and later series that have equivalent PROOT commands may find the solution in less time due to the faster CPU.

```
10 ! SRC12B2
20 OPTION BASE 0
30 !
```

```
40 N=200
50 DIM P(N)
60 COMPLEX Z(N)
70 ! build the polynomial
75 P(N)=2
80 FOR I=N-1 TO 0 STEP -1
90   P(I)=FPRIM(P(I+1)+1,20000)
100 NEXT I
110 !
120 MAT Z=PROOT(P)
125 A=MAXREAL
130 FOR I=0 TO N-1
140   A=MIN(A,ABS(Z(I)))
150 NEXT I
160 DISP A
170 END
```

J-F

EMAIL PM WWW FIND QUOTE REPORT

10th November, 2022, 15:29 (This post was last modified: 10th November, 2022 17:06 by C.Ret.) **Post: #12**

C.Ret
Member

Posts: 223
Joined: Dec 2013

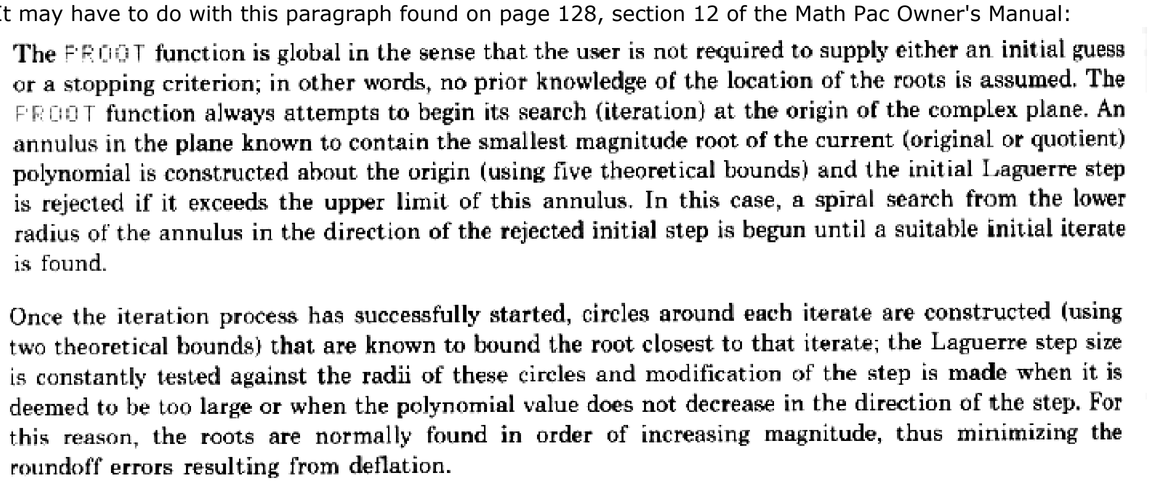
RE: [VA] SRC #012b - Then and Now: Root

> **J-F Garnier Wrote:** (10th November, 2022 10:26)
>
> My understanding is that the PROOT method, with 200 terms, proves that the root Zmin is indeed the smallest in abs value for the complete 10000th degree polynomial:
> - PROOT finds all the roots of the given polynomial (here the 200th degree one), there is no missing one,
> - we can be sure that the root Zmin above is also a root of the 10000th degree polynomial, in the limits of the numerical accuracy, because the terms **Pn.Zmin^n for n>200 will have a modulus less than about 2E-16 and will not contribute (again in the limits of the numerical accuracy) to the value of the complete polynomial**,
> - the complete polynomial can not have a smaller root Zx, because it would also be a root (in the numerical accuracy limits...) of the 200th degree polynomial, for the same reason that **the Zx^n terms for n>200 would be negligible**.

Your explanations make sense and explain why the results no longer evolve when there are enough monômes, the truly high powers of x with |x]<1 becoming negligible at the (limited) 12-figures precision of the machine.
This also explains why using Horner's method as I do waste a lot of computation time because the summation starts with the negligible monômes which will be overwritten by the larger values of the last ones.

Concerning the **PROOT** instruction, I did some tests with about twenty values for *N* ranging from 8 to 66. Each time, I observe that the root of minimal absolute value (of minimal norm) is store at the first position in the result vector containing the roots.
I now short end my code by removing the search loop for the minimum value at the end of the program. I only count of the PROOT program to systematically place the smallest normed root in the first *R(0)* position.

It may have to do with this paragraph found on page 128, section 12 of the Math Pac Owner's Manual:



The PROOT function is global in the sense that the user is not required to supply either an initial guess or a stopping criterion; in other words, no prior knowledge of the location of the roots is assumed. The PROOT function always attempts to begin its search (iteration) at the origin of the complex plane. An annulus in the plane known to contain the smallest magnitude root of the current (original or quotient) polynomial is constructed about the origin (using five theoretical bounds) and the initial Laguerre step is rejected if it exceeds the upper limit of this annulus. In this case, a spiral search from the lower radius of the annulus in the direction of the rejected initial step is begun until a suitable initial iterate is found.

Once the iteration process has successfully started, circles around each iterate are constructed (using two theoretical bounds) that are known to bound the root closest to that iterate; the Laguerre step size is constantly tested against the radii of these circles and modification of the step is made when it is deemed to be too large or when the polynomial value does not decrease in the direction of the step. For this reason, the roots are normally found in order of increasing magnitude, thus minimizing the roundoff errors resulting from deflation.

Having the minimal absolute value (minimal normed or smallest magnitude root) at R(0) **is** not **just** a coincidence...

EDITED: See next post where **Chan** show me that It is effectively just by coïncidence !!

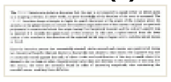
I put a new set of AAA cells in my HP-71B to replace the previous old set that just died during the last endless attempt.

It took nearly two hours to find the minimum norm ***ABS(R(0)) = 0.806513599261*** from a polynomial ***P()*** of degree *N=200* with the following 3-liner directly adapted from **J. -F. Garnier**'s code:

```
10 T0=TIME @ N=200 @ OPTION BASE 0 @ DIM P(N) @ COMPLEX R(N-1)
20 P(N)=2 @ P(N-1)=3 @ FOR K=N-2 TO 0 STEP -1 @ P(K)=FPRIM(2+P(K+1)) @ NEXT K
30 MAT R=PROOT(P) @ T0=TIME-T0 @ DISP T0;ABS(R(0)) @ BEEP

T0
6760.46 (1:52'40.4")

ABS(R(0))
.806513599261

R(0)
(-.645758096347,-.483177676217)

ABS(R(1))
.806513599261

R(1)
(-.645758096347,.483177676217)
```

Attached File(s)

Thumbnail(s)



PM FIND QUOTE REPORT

10th November, 2022, 16:42 **Post: #13**

Albert Chan
Senior Member

Posts: 2,142
Joined: Jul 2018

RE: [VA] SRC #012b - Then and Now: Root

> **C.Ret Wrote:** (10th November, 2022 15:29)
> Concerning the **PROOT** instruction, I did some tests with about twenty values for *N* ranging from 8 to 66. Each time, I observe that the root of minimal absolute value (of minimal norm) is store at the first position in the result vector containing the roots.

Not true. Roots obtained might not be sorted by abs.

Example, hard coded for N=8, last root abs is the minimum, not first.

```
>RUN
 1.81      .735640161749
>MAT DISP R
 (-.379096933597,-.630437913292)     ! abs = 0.73564016175
 (-.379096933597,.630437913292)
 (.125003135678,-.726137429683)      ! abs = 0.736818397379
 (.125003135678,.726137429683)
 (-.677447444085,-.29696080146)      ! abs = 0.739676116352
 (-.677447444085,.29696080146)
 (.518497763744,.521653714169)       ! abs = 0.735501548954
 (.518497763744,-.521653714169)
```

EMAIL PM FIND QUOTE REPORT

10th November, 2022, 17:02 (This post was last modified: 12th November, 2022 07:06 by C.Ret.) **Post: #14**

C.Ret
Member

Posts: 223
Joined: Dec 2013

RE: [VA] SRC #012b - Then and Now: Root

Thank you Chan.

You right, the last two or three roots may or may not have a larger norm than the first one (or first two). The loop to seek for the smallest is still needed.

Here is a corrected version of my code:

```
10 INPUT "n=";N @ T0=TIME @ OPTION BASE 0 @ DIM P(N) @ COMPLEX R(N-1)
20 P(N)=2 @ P(N-1)=3 @ FOR K=N-2 TO 0 STEP -1 @ P(K)=FPRIM(2+P(K+1)) @ NEXT K
30 MAT R=PROOT(P) @ FOR K=1 TO N-1 @ IF ABS(R(K))<ABS(R(0)) THEN VARSWAP R(0),R(K)
40 NEXT K @ T0=TIME-T0 @ DISP T0;ABS(R(0)) @ BEEP
```

I also have to edit my previous post to indicate my mistake...

PM FIND QUOTE REPORT

12th November, 2022, 02:59 **Post: #15**

Valentin Albillo
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

RE: [VA] SRC #012b - Then and Now: Root

.
Hi, all,

Thanks for your interest in this second part of my *SRC#12* and most definitely for your solutions and comments, much appreciated as always.

I'll post my original solution next Sunday circa 23:00 GMT+1 (Spain is physically a *GMT**+0*** country but for some retarded obsolete law we're stuck with *GMT**+1***, which badly shifts the times for everything,) so if anyone wants to have a further say on the subject this is the last chance.

By the way, **C.Ret**, you recently posted this code:

> **C.Ret Lately Wrote:**
> `10 INPUT "n=":N @ T0=TIME @ OPTION BASE 0 @ DIM P(N) @ COMPLEX R(N-1) [...]`

whis is fine save for the fact that I stated this in my *OP*:

> **... but Valentin Albillo Previously Wrote:**
> Your program should have **no inputs** [...]

which disqualifies this code of yours as a valid solution. As *ABBA* said: *"Rules must be obeyed"*.

Also, and it applies to everyone (me included), I think that it's *proper etiquette* to post not just the code *per se*, but also a *run* of it, with *results* and *timings*. This is what I always do as I feel that's the proper way to post a solution: *code, run, results, timings*.

Finally, I feel that few of you heeded my advice about properly *balancing* the work done by the *program* vs. the work done by the *programmer*, with some of you actually doing *a lot* of work to then have the program doing the *very minimum* work necessary, or the other way around.

Enough. As I said, my original solution will be posted next Sunday. Best regards.

V.

PM WWW FIND EDIT QUOTE REPORT

12th November, 2022, 07:21 (This post was last modified: 13th November, 2022 10:22 by C.Ret.) **Post: #16**

C.Ret
Member

Posts: 223
Joined: Dec 2013

RE: [VA] SRC #012b - Then and Now: Root

> **Valentin Albillo Wrote:** (12th November, 2022 02:59)
> By the way, **C.Ret**, you recently posted this code:
>
> > **C.Ret Lately Wrote:**

```
10 INPUT "n=";N @ T0=TIME @ OPTION BASE 0 @ DIM P(N) @ COMPLEX R(N-1) [...]
```

whis is fine save for the fact that I stated this in my *OP*:

> **... but Valentin Albillo Previously Wrote:**
>
> Your program should have **no inputs** [...]

which disqualifies this code of yours as a valid solution. As *ABBA* said: *"Rules must be obeyed"*

That's life, every mistake has to be paid cash!

I copy-paste the wrong version, ...
... Now I am disqualified (but was I able to solve this quizz - not sure really)
... Now I am in big trouble with my internal Q&C, the voll team at the Research Département and HQ directors since I post at a public place a true confidential documentation and badly secured it.

despite I am out of the race now, I still looking for a better solution...

Still have a question, I notice that all the roots found by PROOT are within the unit circle centered at the origin of the complex plane.
Is there any reason for that?
Is there a relation between this all-prime coefficient polynôme and something with a circle or a trigonometry fact?
Isn't there a quick and efficient link between its roots and any characteristic trigonometric fUnction?

hummm.

PM FIND QUOTE REPORT

12th November, 2022, 11:13 (This post was last modified: 12th November, 2022 11:57 by J-F Garnier.) **Post: #17**

J-F Garnier
Senior Member

Posts: 790
Joined: Dec 2013

RE: [VA] SRC #012b - Then and Now: Root

> **C.Ret Wrote:** (12th November, 2022 07:21)
>
> Still have a question, I notice that all the roots found by PROOT are within the unit circle centered at the origin of the complex plane.
> Is there any reason for that?

I believe for the same reason I indicated in my first post above:

> **J-F Garnier Wrote:** (9th November, 2022 10:43)
>
> My first thoughts: a real root must be negative, and greater than -1 since the polynomial quickly takes very large values for |X|>1.

Transposed for complex roots, it means the roots must be |z|<1.
And I believe this is also related to this comment of Valentin:

> **Valentin Albillo Wrote:** (12th November, 2022 02:59)
>
> Finally, I feel that few of you heeded my advice about properly *balancing* the work done by the *program* vs. the work done by the *programmer*, with some of you actually doing *a lot* of work to then have the program doing the *very minimum* work necessary, or the other way around.

This is experimental math for most of us, as Valentin recently pointed it out. Fernando and I (and maybe others silently) tried with PROOT and found a candidate for the minimum root, and the fact that it was *small enough* lead me to my analysis and my solution with 200 terms.

J-F

EMAIL PM WWW FIND QUOTE REPORT

13th November, 2022, 16:15 (This post was last modified: 17th November, 2022 15:26 by Albert Chan.) **Post: #18**

Albert Chan
Senior Member

Posts: 2,142
Joined: Jul 2018

RE: [VA] SRC #012b - Then and Now: Root

For P(x) degree 24 or higher, min abs root is around z = (-2 ± √2*i)/3
We can use Newton's method to zeroed in true root, for N ≥ 24

|z|^2 ≈ (4+2)/9 = 2/3 --> |z|^148 ≈ (2/3)^74 ≈ 9E-14

Set N = 148, we have 12-digits accuracy for N ≥ 148 min abs root.

```
10 DESTROY ALL @ SETTIME 0
20 N=148 @ DIM P(N) @ P(1)=3 ! odd primes
30 FOR K=2 TO N @ P(K)=FPRIM(P(K-1)+2) @ NEXT K
40 COMPLEX Z,F0,F1 @ Z=(-2,SQR(2))/3 @ A1=-1
50 A0=A1 @ A1=ABS(Z) @ DISP TIME,Z,A1 @ IF A1=A1+10*(A1-A0)^2 THEN END
60 F0=0 @ F1=0 @ FOR K=N TO 1 STEP -1 @ F0=F0*Z+P(K) @ F1=F1*Z+K*P(K) @ NEXT K
70 F0=F0*Z+2 @ Z=Z-F0/F1 ! newton's method
80 GOTO 50
```

```
>RUN
 .51       (-.666666666667,.47140452079)       .816496580927
 1.9       (-.645842585444,.480858085479)      .805193854636
 3.22      (-.645737371307,.48318762432)       .806502965276
 4.53      (-.645758096686,.483177673982)      .806513598193
 5.9       (-.645758096347,.483177676218)      .806513599261
```

EMAIL PM FIND QUOTE REPORT

14th November, 2022, 01:23 (This post was last modified: 14th November, 2022 01:57 by PeterP.) **Post: #19**

PeterP
Member

Posts: 172
Joined: Jul 2015

RE: [VA] SRC #012b - Then and Now: Root

This is in homage to a lot of learning that I was allowed to do here about polynomial roots, thanks to the ever generous author and Albert Chan.

I tried Barstow's method as that was something I found on the forum but the hp41 only gets to about 20-30 terms. Not enough.

However, given the 10 digit accuracy, the first 100-128 terms should be enough. Thanks to the teachings from Albert, I implemented a root squaring algorithm, looking for the max abs root of 1/P(x).

I can get it to run with about 100 terms, given the memory limitations of the HP41CX.

Result is 0.806427842 in about 23 seconds. Not very impressive accuracy. More squaring would be required, but I dont have enough registers, and not enough digits of accuracy either I guess.

The code first creates the list of 100 or so primes and stores it into registers. And then successively squares them until we get only 3 elements. And then calculates the max abs root by dividing the third element by the first element, taking the (2*number_of_squaring)th root, for the max abs of Q(x) = 1/P(x). 1/x gives then the min abs root.

The listings are attached as pictures. I will try to type them up as well, but I am worried about typos.

Thank you again for a wonderful learning experience!

Cheers

Peter

Attached File(s)

Thumbnail(s)

PM FIND QUOTE REPORT

14th November, 2022, 01:44 (This post was last modified: 14th November, 2022 01:55 by PeterP.) **Post: #20**

PeterP
Member

Posts: 172
Joined: Jul 2015

RE: [VA] SRC #012b - Then and Now: Root

Ok, here are the listings typed up, please excuse any typos:

It is called with
XEQ “VA2” and stops after about 23 seconds with the result in X.

The elements a[n] to a[n-x] are in the registers sss.eee, with sss > eee. This control number is stored in R00
The calculated elements b[n] from the squaring are in registers bbb.fff with bbb > fff. This control number is stored in R01
Once all b[n] are calculated, the new boundaries are calculated (1/2 of bbb-fff. And bbb-fff is half of sss-eee)
The b[n]’s become the new a[n]s, and the control numbers are created accordingly,

R02 accumulates the values for each b[n-i] as the summation over all j = 1….i takes place.
R03 holds the current j
R04 holds the current i
R06 holds the number of elements to calculate

LBL 00 is the loop summing over the elements j = 1….i
LBL 01 is the loop over all i
LBL 03 is the loop over the successive b[n], b’[n], b’’[n], etc
LBL 05 is the final min abs root value calculation after the last b’’…[n] series calculation which has only 3 elements.

——————————————-

REM - Fill Primes in descending order into registers. Called with a control word of sss.eee

```
LBL “PL”
50
SETCSPD
RDN
STO M
2
STO IND M
DSE M
VIEW X
3
VIEW X
STO IND M
DSE M
LBL 00
INCX
INCX
PRIME?
GTO 01
LASTx
GTO 00
LBL 01
View X
STO IND M
DSE M
GTO 00
END
```
————————-
```
REM….. Main PRogram
LBL “VA2”
109.009
STO 00
XEQ “PL”
RCL 00
159.110
STO 01
X<>Y
-
INT
STO 06
LBL 03
1
STO 04
RCL Ind 00
X^2
```

```
STO IND 01
LBL 01
RCL 04
STO 03
CLX
STO 02
LBL 00
RCL 03
RCL 04
-
RCL 00
+
RCL IND X
STO 05
RCL 03
CHS
RCL 04
-
RCL 00
+
RCL IND X
ST* 05
2
-1
RCL 04
RCL 03
-
Y^x
*
RCL 05
*
ST+02
DSE 03
GTO 00
RCL 00
RCL 04
-
RCL IND X
X^2
-1
RCL 04
Y^X
*
RCL 02
+
RCL 01
RCL 04
-
X<>Y
STO IND Y
RCL 04
INCX
STO 04
Enter
RCL 06
X>Y?
GTO 01
RCL 06
2
/
INT
STO 06
2
X>Y?
GTO 05
RDN
RCL 00
INT
X<>Y
-
1000
```

```
/
RCL 00
INT
+
X<>01
X<>00
GTO 03
LBL 05
-2
RCL 01
RCL IND X
RDN
+
RCL IND X
R^
/
RCL IND 01
LOG
2
LOG
/
2
*
1/x
Y^X
1/x
Beep
CLD
END
```

PM FIND QUOTE REPORT

14th November, 2022, 02:13 **Post: #21**

Albert Chan
Senior Member

Posts: 2,142
Joined: Jul 2018

RE: [VA] SRC #012b - Then and Now: Root

It may help to explain PeterP's root squaring program.
This was a PM I sent to PeterP, and other members.

Albert Chan Wrote:

Instead of min abs P roots, we solve for max abs Q root, Q(x) = P(1/x)

Graeffe's root squaring method (next row roots = previous roots^2)

We only show top 3 coefficients, because we only care for Q max abs root

q = [2,3,5,7,11,13,17,19,23,29, ...] // assumed infinite degree polynomial
→ [2^2, 11, 27]
→ [2^4, 95, 303]
→ [2^8, 671, 7775]
→ [2^16, 3.530559E6, 1.01291839E8]
→ [2^32, 8.11677068927E11, 4.099840909585279E15]
→ [2^64, 3.455854558672141E25, 1.816641529875401E31]
→ [2^128, −5.240706455638273E50, 2.748844184968018E62]
→ [2^256, 8.757340043013178E100, 7.559454552508339E124]
→ [2^512, 9.837400259693430E201, 5.714553957282261E249]
...

2nd column are not "pure squares" (note the negative sign), but 3rd column is.
Thus, roots to seek are complex conjugates.
Assuming roots are well separated now:

max abs Q root = surd(5.714553957282261E249/2^512, 512*2) = 1.2399046971021601
min abs P root = 1 / (max abs Q root) = 0.8065135992606103

[VA012b] is similar to [VA012a], a diffusion problem.
Root Squaring process also based from its neighbor cells.

```
b[n] = a[n]^2
b[n-1] = -a[n-1]^2 + 2*a[n]*a[n-2]
b[n-2] = +a[n-2]^2 - 2*a[n-1]*a[n-3] + 2*a[n]*a[n-4]
...
```

By the time big primes effect diffused to the top, its effects are miniscule.

EMAIL PM FIND QUOTE REPORT

15th November, 2022, 02:51 **Post: #22**

Valentin Albillo
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

RE: [VA] SRC #012b - Then and Now: Root

Hi, all,

Well, a full week has elapsed since I posted my *OP*, which has already passed the *1,300 views* mark (and *Problem 1* has exceeded *9,000 views* already,) and I've got a number of solutions and/or comments, namely by ***Werner***, ***Jean-François Garnier***, ***C.Ret***, ***PeterP***, ***Fernando del Rey*** and ***Albert Chan***. Thank you very much to all of you for your interest and valuable contributions.

Now I'll provide my *original solution* to this ***Problem 2*** but first a couple' comments:

> 1) Some of you provided just *code* but no numerical results, others provided numerical *results* but no code, and most of you provided *comments* but didn't provide *timings*. As I said in my previous post, I'd consider proper etiquette and most useful for everyone if contributors would kindly post *program code*, *a sample run*, *results* and *timings*. Not a rule but it would help. And *comments* are always most welcome, of course.
>
> 2) Again, I'm mildly surprised that *no one* posted *RPL* solutions or *RPL* code of any kind. As for *RP**N*** code, **PeterP** made a most brave attempt to get a reasonably accurate result (as did **Werner** with **Free42**, though using capabilities not available on the vintage **HP-42S**,) but it might be that *RPL* people deem this challenge as too trivial for their powerful vintage *RPL* calcs. On the other hand, perhaps it might be that ... naw, never mind.

That said, these are my own approach and resulting *original solution:*

First of all, the main difficulty is the ***10,000****th*-degree. Were it a mere 100th-degree polynomial, it would be quite trivial, but dealing with the full polynomial using `PROOT`, say, would require allocating 10,000 real elements for the coefficients (they don't fit as integers) plus another 10,000 complex elements for the roots, for a grand total of ~ 30,000 x 8 = **240 Kb**, to which you must add the considerable memory that `PROOT` needs internally (21 x 10.000 + 261 ~ **210 Kb**,) which adds up to ~ **450 Kb**, surely *exceeding* maximum available *RAM*).

Then again, finding the 10,000 roots woud take ~ *(10,000/100)^{2}x* the time required to deal with a 100th-degree polynomial, which on a *physical* **HP-71B** is about *2,100 sec.*, so the big one would take *2,100 x 100^2* ~ ***243 days***. In other words, utterly *unfeasible*. So much for sheer brute force ...

Now, from theoretical considerations it's pretty obvious that none of the 10,000 roots can have an absolute value *(aka modulus, aka magnitude)* ***> 1***, because then the highest term, **104,743 x^{10,000}**, would dominate the sum, making it nonzero. Likewise, if the roots are too close to ***0*** then their powers will quickly tend to zero, thus failing to ever contribute enough negative value to compensate for the lowest coefficient, ***2***. Thus, all the roots must reside in an *annulus* of external radius ***1*** and internal radius to be determined by the minimum magnitude among all the roots, which we can roughly estimate as I'll explain in a moment.

Once we have a rough estimation for said minimum magnitude, we can compute *another estimation*, this time for the *minimum degree* of the truncated polynomial so that its highest term already contributes negligibly to the evaluation of the polynomial according to the precision required, say *12 digits*.

My resulting program is thus this *5-liner: (REPEAT/UNTIL are from the JPC ROM, just for show)*

```
1  DESTROY ALL @ @ OPTION BASE 0 @ M=0 @ W=0 @ N=0 @ FIX 2
2  REPEAT @ W=M @ N=N+10 @ GOSUB 4 @ DISP N;M;P @ UNTIL ABS(M-W)<.01 @ STD
3  N=IP((-12-LGT(P))/LGT(M)) @ DISP "Deg:";N;".." @ GOSUB 4 @ DISP "Min:";M;P @ END

4  DIM A(N) @ COMPLEX R(N) @ P=2 @ A(N)=P @ FOR I=1 TO N @ P=FPRIM(P+1) @ A(N-I)=P
5  NEXT I @ MAT R=PROOT(A) @ M=1 @ FOR I=0 TO N-1 @ M=MIN(M,ABS(R(I))) @ NEXT I @ RETURN
```

Line *1* does some initialization.

Line *2* finds the *minimum absolute value* for polynomials of degrees *10*, *20*, *30*, ... until two consecutive minimum values are within *0.01*, which essentially gives us the result *correct to 2 digits (~0.81)*

Line *3* now uses this 2-digit value to compute an *estimation* to the *minimum necessary degree* to get a fully accurate result, which it then obtains and displays, correct to *12 digits*. It is important to use the minimum-degree truncated polynomial for speed reasons *("the faster, the better")* because **PROOT** finds the roots of a *153th*-degree polynomial about **71%** *faster* than for a *200th*-degree one, say.

Lines *4* and *5* are a subroutine which creates the *Nth*-degree polynomial, fills it up with the prime coefficients, computes all *N* complex roots and returns the minimum magnitude among them. All of it in just *2 lines* of code.

This way, the user doesn't have to guess and provide any particular degree for the truncated polynomial at all, the program finds the necessary degree and then the sought-for minimum absolute value to maximum accuracy (12 digits). In other words, the program does all the work, fast.

Let's do a sample run:

```
>RUN

     Deg   Min   Ncoef
     -----------------
      10   0.73   31
      20   0.80   73
      30   0.81   127
      40   0.81   179
      Deg: 153 ..
      Min: .806513599261    887
```

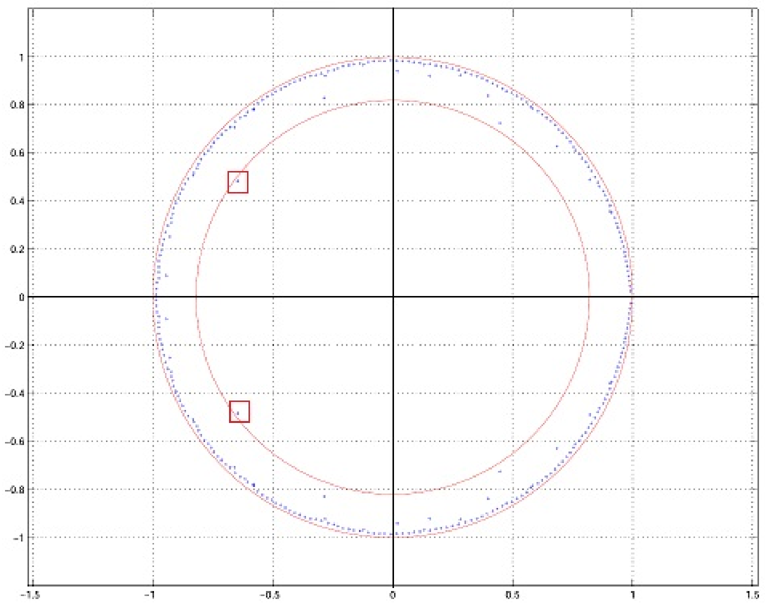
correct to 12 digits in **1 hr 18 min** on a *physical* **HP-71B**. We can check the highest term contribution like this:

```
     >P;M;N;P*M^N  ->  887   .806513599261   153   4.56575832163E-12
```

which indeed contributes negligibly to the value of the polynomial so it and all other higher terms can be safely ignored.

As for using **PROOT** to find all the *N* roots (most of them complex) vs. using ***Newton**'s method* to directly find the root that results in the desired minimum magnitude (as **A.Chan** does in post#18,) the reason is that in general that *might not guarantee* that you get the absolute minimum.

In **A.Chan**'s post it works but in general it could be just *luck*, no guarantee at all without further considerations to try and find a working *initial guess*, and as you can see in the graph below most roots are within a *thin annulus* of radii ***1*** and ***0.82***, i.e. only *0.18* wide, and the conjugated pair we need are extremely near to it, so the roots are quite *clustered* and isolating the one we want might be a *hit-or-miss* affair, while using **PROOT** and then finding the minimum magnitude is *100% guaranteed* to find the correct one, giving us perfect ease of mind with minimum effort on our part.



Well, that will be it for now. Thanks again to all of you who contributed and some of you even solved both *Problem 1* and *Problem 2*, a perfect ***2 for 2*** score so far, but as they were the *easiest* problems of the lot I wonder if you'll manage to also solve incoming ***Problem 3***, which is sure to raise the bar ... slightly ? We'll see ...

V.

PM WWW FIND EDIT QUOTE REPORT

16th November, 2022, 15:28 (This post was last modified: 16th November, 2022 21:04 by Albert Chan.) **Post: #23**

Albert Chan
Senior Member

Posts: 2,142
Joined: Jul 2018

RE: [VA] SRC #012b - Then and Now: Root

If we consider polynomial coefficients with geometric progression:

f(x) = 1 + (r*x) + (r*x)^2 + ... + (r*x)^n = ((r*x)^(n+1) - 1) / ((r*x) - 1)

From RHS numerator, all f roots abs = 1/r

P(x) = 2 + 3x + 5x^2 + 7x^3 + 11x^4 + ...

P roots, sorted in abs: (-0.6458±0.4832i), (0.4472±0.7248i), (-0.2853±0.8292i), ...
With corresponding abs: **0.8065**, 0.8517, 0.8769, ...

Primes does not grow as fast as geometric progression. (sum of reciprocal primes diverges)

P min abs root is due to ratio, (5/2=2.5) > (11/5=2.2)
If the ratios were about the same, we expected f(x) roots pattern, with similar sized roots.

Example, R(x) = P(x)+0.5, 5/(2+0.5) = 2.

R roots, sorted in abs: (-0.6811±0.5122i), (0.4692±0.7114i), (-0.2905±0.8666i), ...
With corresponding abs: **0.8522, 0.8522**, 0.9140, ...

If guess close enough, we can use Newton's method to zeroed in P min abs root.

EMAIL PM FIND QUOTE REPORT

18th November, 2022, 03:44 **Post: #24**

Valentin Albillo
Senior Member

Posts: 958
Joined: Feb 2015
Warning Level: 0%

RE: [VA] SRC #012b - Then and Now: Root

Hi, **Albert Chan**,

Thanks for your recent additional comments, I appreciate it. However, I have a thing or two to comment back, read on ...
(all highlights are mine)

> **Albert Chan Wrote:**
>
> > **Valentin Albillo Wrote:**
> >
> > ***Line 2*** *finds the minimum absolute value for polynomials of degrees 10, 20, 30, ... until two consecutive minimum values are within 0.01, which essentially gives us the result correct to 2 digits (~0.81)*
>
> **Slight error in logic.**

Not at all, see below.

> **Albert Chan Wrote:**
>
> If consecutive minimum abs both ~0.81, we cannot deduce trend apply to higher degree. (**we can assume trend continues, but have to later <u>test</u> validity of assumption**)

And I *<u>did</u>* test, it's just that I didn't want to make an already long post any longer by including unneeded data that most people won't be interested in, as they understand the ***scope*** of my *articles* and *challenges* and trust my results (which they can verify by themselves, if in doubt), but as you seem to like said data, here you are, the checks I did (which I saved but didn't post):

```
     Degree  Min. Magnitude
     -----------------------
       1      .66666 6666667
       2      .63245 5532034
       4      .65224 6975033
       8     .73550 1548954
      16     .76890 4440166
      32     .80252 6072477
      64     .80650 0035750
      96     .80651 362173
     128     .80651 3599285

     140     .80651 3599258
     145     .80651 3599260
     150     .80651 3599261
     160     .80651 3599261
     165     .80651 3599261
     170     .80651 3599261
     175     .80651 3599261
     180     .80651 3599261
     185     .80651 3599261
     190     .80651 3599261
     195     .80651 3599261
     200     .80651 3599261
```
and I was more than satisfied that the trend continued alright and converged to the correct solution, namely ***.806513599261***.

> **Albert Chan Wrote:**
>
> > **Valentin Albillo Wrote:**
> >
> > ***Line 3*** *now uses this 2-digit value to compute an estimation to the minimum necessary degree to get a fully accurate result, which it then obtains and displays, correct to 12 digits.*
>
> **Close**, but not quite.
>
> If this "minimum necessary degree" polynomial also gives abs ≤ 0.81, **we are done**.
> **However, if it gives bigger abs**, we have to repeat again, with even higher degree polynomial.

But it *didn't* give ***"bigger abs"***, it gave ***.806513599261***, which is less than *0.81*, so your comment *doesn't apply* at all and my statement is fully *correct*, not merely *"close"*.

Now a word on the ***scope*** for my *articles* and *challenges*. In a past thread in which you took part (you posted *5 times* no less) and so you surely read my posts there, I said:

> "My articles are intended as just that, articles to be published in a physical fan-made magazine [...] or else on the *WWW (MoHPC)* for *all kinds* of fans, most of them *not scholars*, so my articles do not have the structure nor goals of a formal *peer-reviewed* paper."

In other words, ***my goal*** is first and foremost to provide *entertainment* to the forum members and *HP calc* fans in general, enticing them to think about the challenge and how to use their *vintage HP calc* to solve it, and if additionally they *learn* something new and interesting (and even useful) from my productions then *so much the better*. ***That*** is my goal.

Yours is obviously different, seemingly centered on lecturing, posting *symbolic proofs* and *theoretical* ramblings and lots of data obtained in *Xcas* sessions, *lua*, *Mathematica*, the works. *Good for you* and for the people who like *(lots of)* posts like that. Not my cup of tea *here*.

And please leave aside topics having little or nothing to do with my present *Problem 2* (the references to *integration*, *Borwein integrals* and *Kahan*), save that for your own threads or where it's appropriate. Thanks.

V.