[VA] SRC #012e - Then and Now: Roots Threaded Mode | Linear Mode

7th February, 2023, 22:00 **Post: #1**

Valentin Albillo
Senior Member

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

[VA] SRC #012e - Then and Now: Roots

Hi, all,

Well, today is Feb **7** and my *888th* post here, so **Welcome** to the **5th part** *(next-to-last)* of my ongoing ***SRC #012 - Then and Now***, where I'm showing that advanced vintage *HP* calcs which were great problem-solvers back **THEN** in the 80's are **NOW** still perfectly capable of solving recent, highly non-trivial problems intended to be tackled using modern *PC*s, never mind ancient calcs.

In the next weeks I'm proposing **six** *increasingly harder* such problems for you to try and solve using your vintage *HP* calcs while abiding by the *mandatory rules* summarized here:

> You **must** use **VINTAGE HP CALCS** (physical/virtual,) coding in either **RPN**/*Mcode*, **RPL**/*SysRPL* or **HP-71B BASIC**/*FORTH*/*Assembler*, and **NO CODE PANELS**.

Caveat: the reason for this rule is obvious: this ***Problem 5*** has been tailored for and is challenging *if and only if* you attempt to solve it using a *vintage calc*. Tackling it using modern software (e.g. ***Mathematica***, etc.) is *meaningless* and just succeeds in spoiling the fun, so please don't do it. Now back to business ..

Once ***P1*** *(**Probability**)*, ***P2*** *(**Root**)*, ***P3*** *(**Sum**)* *and* ***P4*** *(**Area**)* are over, now's the turn for ***P5 (Roots)*** which revisits *rootfinding*, only this time the function whose roots are to be found has its own *peculiarities* and the final result is both wholly *unexpected* and utterly ***amazing*** ! ... But first a relevant, hopefully interesting *preamble*:

Preamble

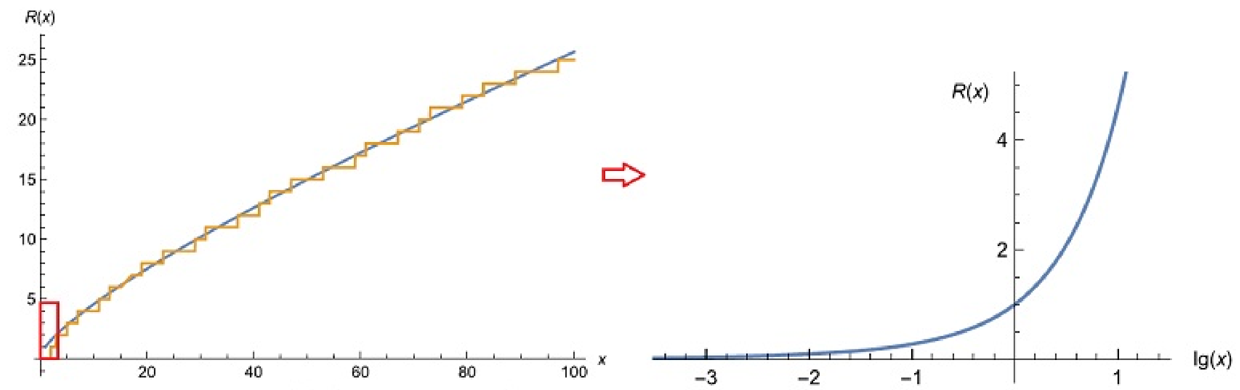
Some of you might remember an article where I computed an approximation to the ***Π(x)*** function, which gives the number of primes up to some given limit (say, the number of primes up to *1,000*, which is *168*) by using an equivalent form of the ***R(x)*** *function*, which is canonically defined as

$$R(x) = \sum_{k=1}^{\infty} \frac{\mu(k)}{k} \operatorname{li}(x^{1/k}) \quad \text{where} \quad \operatorname{li}(x) = \int_0^x dt/\log t$$

and ***μ(k)*** is the ***Möbius*** *function*. However, this form of ***R(x)*** is not convenient for computations because both ***li(x)*** and ***μ(k)*** are time-consuming to evaluate and the convergence is poor, so I used instead a more amenable, equivalent form valid for ***x > 0***, namely:

$$R(x) = 1 + \sum_{k=1}^{\infty} \frac{\log^k x}{k \cdot k!\, \zeta(k+1)} \quad \text{where} \quad \zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \frac{1}{1^s} + \frac{1}{2^s} + \frac{1}{3^s} + \cdots$$

Using this form we easily get *R(10^3) ~ 168* primes up to *1,000* (*err = 0%*), *R(10^6) ~ 78,527* primes up to one million *(err ~ 0.037%)* or *R(10^9) ~ 50,847,455* primes up to one billion *(err ~ 0.00016%)*. Indeed, ***R(x)*** is an *extremely good* approximation, as seen in the figure below (left image) which compares the graphs of ***R(x)*** and ***Π(x)***:



prime counting: $R(x)$ (smooth curve) vs. $\pi(x)$ (step function)

Now you may be wondering where the *"rootfinding"* fits in all this. If we look at the graphics above (notice the ***X***-axis' *logarithmic* scale in the zoomed image), ***R(x)*** seems to be *always positive* for ***x > 0***, as it certainly looks like it will never cross the ***X*** axis and thus will have no positive real roots at all ... but quoting *Pink Floyd: "things are not what they seem"*, and so we have:

Problem 5: Roots

Write a program to compute and output the **7 largest positive real roots** of ***R(x)***, in decreasing order.

You can use *any equivalent form* of ***R(x)*** that suits you best, be it the canonical definition or the one I used in my article and in the examples here, or any other you deem appropriate.

Your program should have *no inputs* and must compute and output the ***7*** *roots* and end. You should strive to get at least ***7*** *correct digits* (give or take a few *ulp*) for each of the **7** roots and the faster the running time the better.

Note: The *main goal* here is to ***compute* the 7 roots**, but as for intermediate values (*e.g.* values of special functions,) you can choose whether to compute them *on the fly* or else to get them *from references* and include them as **DATA** statements or whatever. The former approach will result in a smaller but slower program while the latter will be faster but bigger, or you can mix both approaches as you see fit. Your choice.

If I see interest I'll post my own *original solution* for the **HP-71B**, a *15-line* program which automatically does the job, plus comments. Meanwhile, let's see your very own clever solutions **AND** please remember the above rules.

V.

PM WWW FIND EDIT QUOTE REPORT

10th February, 2023, 20:05 **Post: #2**

J-F Garnier
Senior Member

Posts: 790
Joined: Dec 2013

RE: [VA] SRC #012e - Then and Now: Roots

Here is where I am presently: I implemented the R(X) function and verified that I got the correct values for R(1000), R(1E6) ... R(1E9).
Then I explored how R(x) behaves for X<<1. It constantly decreases (staying positive) down to log(x)=-22 at least, meaning there is no root greater than 1E-10.
Then, unfortunately, my implementation is only giving numeric garbage, so I can't get any conclusion.
Some details: following Valentin's advice, I implemented the zeta function as a table with values from the literature until zeta(16), then computed the following terms.
The R(x) function is evaluated using the Horner algorithm, after identifying the highest term needed.

I'm afraid I will not be able to go further, unless some has a better idea to evaluate R(X).
Below the program I used, if this can be helpful for others.

J-F

```
10 ! SRC12E
20 OPTION BASE 1
30 M=250
40 DIM F(M),Z(M)
50 !
60 DATA 1.64493406685,1.20205690316,1.08232323371,1.03692775514
70 DATA 1.01734306198,1.00834927738,1.00407735620,1.00200839283
80 DATA 1.00099457513,1.00049418860,1.00024608655,1.00012271335
```

```
90 DATA 1.00006124814,1.00003058824,1.00001528226
100 Z(1)=1
110 FOR K=2 TO 16 @ READ Z(K) @ NEXT K
120 FOR K=K TO M
130 N=2 @ S=0
140 A=N^(-K) @ S=S+A
150 IF A>=1.E-13 THEN N=N+1 @ GOTO 140
160 Z(K)=1+S
170 NEXT K
180 FOR K=1 TO M @ F(K)=FACT(K) @ NEXT K
190 !
200 DEF FNR(L)
210 K=8
220 A=L^K/(K*F(K)*Z(K+1))
230 IF ABS(A)>1.E-13 THEN K=K+2 @ GOTO 220
240 R=0
250 FOR K=K TO 1 STEP -1
260 A=1/(K*F(K)*Z(K+1))
270 R=R*L+A
280 NEXT K
290 FNR=R*L+1
300 END DEF
310 !
320 DISP "R(1E3)=";FNR(LOG(1000))
330 DISP "R(1E6)=";FNR(LOG(1.E+6))
340 DISP "R(1E9)=";FNR(LOG(1.E+9))
350 DISP "R(1E12)=";FNR(LOG(1.E+12))
360 DISP "R(4E16)=";FNR(LOG(4.E+16))
370 !
380 DISP "LOG(X)  R(X)"
390 FOR L=0 TO -40 STEP -1
400 DISP L;FNR(L)
410 NEXT L

>RUN

R(1E3)= 168.359446282
R(1E6)= 78527.3994306
R(1E9)= 50847455.4255
R(1E12)= 37607910540.7
R(4E16)= 1.07529277872E15

LOG(X)  R(X)
0 1
-1 .557331425899
-2 .325459335158
-3 .19890131983
-4 .126895395868
-5 .08422535701
[...]
-20 .00260522574
-21 .002335956316
-22 .002172618526
-23 .002356240694
-24 .003210607624
[...]
-31 .013358513742
-32 -.75233322896
-33 -3.53759316048
-34 -10.161375979
-35 -23.8480080257
-36 -54.9637121895
```

EMAIL PM WWW FIND QUOTE REPORT

19th February, 2023, 15:23 (This post was last modified: 19th February, 2023 16:04 by J-F Garnier.) **Post: #3**

J-F Garnier
Senior Member

Posts: 790
Joined: Dec 2013

RE: [VA] SRC #012e - Then and Now: Roots

J-F Garnier Wrote: (10th February, 2023 20:05)

> I'm afraid I will not be able to go further, unless some has a better idea to evaluate R(X).

Well, it took me some efforts but I got an approximation of the R(X) function good enough to locate the roots.
To be fully honest: I took advantage of the clues given in the alternate thread, that helped me to find a paper with the formal solution suited for this problem. I didn't fully understand the underlying math, but was able to decipher the formula and translate it into 71B code, with the help of Wolfram for the math constants, a resource that seems to be allowed here, contrary to my initial understanding. This is all of my contribution in this challenge.

But finally, here are my results:

I first compared the approximation with the direct summation I tried previously. It turns out that this first-order approximation starts to match the direct calculation around log(x)=-20, at the moment where the direct sum starts to fail. Then the approximation decreases smoothly for several decades until 1E-13 where it starts to oscillate around zero. Using a search of sign reversals, then the FNROOT solver provided me an approximation of the roots.

The approximation may be improved, but that's ok for me. I learnt something, this time using Web tools.
It's nice to be able to reproduce results got elsewhere with Mathematica-like software, but the negative point is that it's pure math, you have to trust complicate (and obscure, for me) math formula without any other way to check it.

```
10 ! SRC12E2
20 OPTION BASE 1
30 M=250
40 DIM Z(M)
50 ! Table of zeta(x)-1
60 DATA 0.644934066848,0.202056903160,0.0823232337111,0.0369277551434
70 DATA 0.0173430619844,0.00834927738192,0.00407735619794,0.00200839282608
80 DATA 0.000994575127818,0.000494188604119,0.000246086553308,0.000122713347578
90 DATA 0.0000612481350587,0.0000305882363070,0.0000152822594087
100 FOR K=2 TO 16 @ READ Z(K) @ NEXT K
110 FOR K=K TO M @ N=2 @ S=0
120 A=N^(-K) @ S=S+A @ IF A>=1.E-13*S THEN N=N+1 @ GOTO 120
130 Z(K)=S @ NEXT K
140 !
150 ! direct summation
160 DEF FNR(L)
170 K=8 @ R=0
180 A=L^K/(K*FACT(K)*(1+Z(K+1))) @ IF A>1.E-13 THEN K=K+2 @ GOTO 180
190 FOR K=K TO 1 STEP -1 @ R=R*L+1/(K*FACT(K)*(1+Z(K+1))) @ NEXT K
200 FNR=R*L+1 @ END DEF
210 !
220 ! 1st order approximation, for small enough X
230 COMPLEX R1,Z1,G1
240 R1=(.5,14.1347251417) ! rho1=first non-trivial zero of zeta function
250 Z1=(.783296511867,.124699829748) ! zeta'(rho1)
260 Z2=-.0304484570584 ! zeta'(-2)
270 G1=(-1.44555144882,5.52278808182)*1.E-10 ! gamma(1-rho1)
280 DEF FNR1(L)=-2*L^(-3)/3/Z2+2*REPT(G1*L^(R1-1)/(R1-1)/Z1)
290 !
300 ! compare sum/approx.
310 DISP "LOG(X) sum R(X) : approx. R1(X)"
320 FOR L=2 TO 40 STEP 2
330 DISP -L;FNR(-L);1+FNR1(L)-1
340 NEXT L
350 DISP
360 ! search for sign reversals
370 S=1
380 FOR L=100 TO 200000 STEP 100
390 R=FNR1(L) @ IF SGN(R)=S THEN 430
400 S=SGN(R)
410 X=FNROOT(L-100,L,FNR1(FVAR))
420 X=-X/LOG(10) @ DISP "LOG10(X)=";X;" X=";IROUND(10^(X-INT(X)));"x 10^";INT(X)
430 NEXT L
440 DISP
```

```
LOG(X) sum R(X) : approx. R1(X)
-2 .325459335158 2.73686555517
-4 .126895395868 .34210819444
[..]
-18 .003465662878 .00375427371
-20 .00260522574 .00273686556
-22 .002172618526 .00205624762
```

```
-24 .003210607624 .00158383424
-26 .00668381081 .00124572851
[..]
-38 -394.875761543 .00039901817
-40 -1051.95605513 .00034210819

LOG10(X)=-14827.7100594 X= 2 x 10^-14828
LOG10(X)=-15300.7353704 X= 2 x 10^-15301
LOG10(X)=-21381.4705128 X= 3 x 10^-21382
LOG10(X)=-25461.6898751 X= 2 x 10^-25462
LOG10(X)=-32711.8833976 X= 1 x 10^-32712
LOG10(X)=-40219.6322135 X= 2 x 10^-40220
LOG10(X)=-50689.8146055 X= 2 x 10^-50690
LOG10(X)=-62979.7778881 X= 2 x 10^-62980
LOG10(X)=-78890.2152906 X= 6 x 10^-78891
```

J-F

EMAIL PM WWW FIND QUOTE REPORT

19th February, 2023, 19:37 **Post: #4**

Fernando del Rey
Junior Member

Posts: 19
Joined: Dec 2013

RE: [VA] SRC #012e - Then and Now: Roots

Wow!, J-F, I'm impressed!

Even with all the clues given in the alternative thead, I have failed miserably with this problem. It's way beyond my reach!

Out of curiosity, I entered your code in Emu71/Win and reproduced the same results, as expected. I haven't tried it on the physical 71B, but in the emulator it runs in no time!

It would be interesting to get an idea of timing on a physical 71B. After all, one purpose of Valentin's challenges is to demonstrate that our old vintage calcs are capable of solving recently posted problems aimed at today's computing hardware. Getting the solutions in a reasonable time would be meaningful to this demonstration.

I'm eager to see Valentin's original solution. I hope he will post it here!

EMAIL PM FIND QUOTE REPORT

19th February, 2023, 19:57 **Post: #5**

Valentin Albillo
Senior Member

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

RE: [VA] SRC #012e - Then and Now: Roots

.
Hi, **Fernando**,

> **Fernando del Rey Wrote:** (19th February, 2023 19:37)
> It would be interesting to get an idea of **timing on a physical 71B**. After all, one purpose of Valentin's challenges is to demonstrate that our old vintage calcs are capable of solving recently posted problems aimed at today's computing hardware. **Getting the solutions in a reasonable time would be meaningful to this demonstration**.

My original solution produces all seven roots with good accuracy in just **8' 50"** on a physical **HP-71B**, which seems to me to be a perfectly reasonable time for such difficult problem when tackled using a *~40 year-old* handheld calculator *cum* "pocket computer".

> **Quote:**
> **I'm eager to see Valentin's original solution. I hope he will post it here!**

Thanks for you enthusiastic interest, **Fernando**. I'll wait for a few more days just in case someone else wants to post something but afterwards you can count on it.

Best regards.
V.

23rd February, 2023, 23:51 **Post: #6**

Fernando del Rey
Junior Member

Posts: 19
Joined: Dec 2013

RE: [VA] SRC #012e - Then and Now: Roots

I have entered J-F's code in a physical 71B and timed its execution. It takes around 18 minutes to find all the 9 roots, still a very reasonable time for a 40 year-old hardware.

As the thread has been quiet for several days, I suppose Valentín will soon post his original solution. Let's see!

EMAIL PM FIND QUOTE REPORT

26th February, 2023, 02:34 **Post: #7**

Valentin Albillo
Senior Member

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

RE: [VA] SRC #012e - Then and Now: Roots

Hi, all,

Caveat lector: this post is VERY *long*, so you might want to read it only when you can allocate enough free time for it.

Well, almost three weeks have elapsed since I posted ***Problem 5*** and, as expected, its difficulty and advanced math subject matter resulted in very few posts and just a single solution *(*)* other than my original one. Back in *ye goode olde* forum, problems like this would've been made short work of ... Ah, those were the days ! ...

Again, no *RPL* (or *RPN*) solutions at all, and this time the tireless contributors were ***Fernando del Rey*** and ***J-F Garnier*** *(*)*, plus additional posts in a parallel thread by ***EdS2***, ***J-F Garnier***, ***Fernando del Rey*** and ***PeterP***. Thank you very much to all of you for your interest and valuable contributions.

Now, this is my detailed ***sleuthing*** *process* and resulting *original* ***solution***, plus *additional* ***comments:***

My sleuthing process

First of all, we must decide which equivalent formula for ***R(x)*** is the most suitable for *rootfinding*. The *canonical formula*, namely:

$$R(x) = \sum_{k=1}^{\infty} \frac{\mu(k)}{k} \operatorname{li}(x^{1/k}) \quad \text{where} \quad \operatorname{li}(x) = \int_0^x dt/\log t$$

is absolutely *useless* for the purpose, as ***μ(k)***, the ***Möbius*** *function*, is hard to compute for large integer ***k***, while ***li(x)*** is time-consuming to evaluate accurately and last but not least, the convergence of the summation is extremely *slow*, thus this canonical formula is out of the question.

As for the second formula, *aka* the *Gram series*, which I used in my article, namely

$$R(x) = 1 + \sum_{k=1}^{\infty} \frac{\log^k x}{k \cdot k!\,\zeta(k+1)} \quad \text{where} \quad \zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \frac{1}{1^s} + \frac{1}{2^s} + \frac{1}{3^s} + \cdots$$

it does converge very nicely for large arguments ***x*** but not so much for arguments ***x << 1*** because then ***ln(x)*** is large and *negative* and this causes the summation's terms to *oscillate widely* between positive and negative while initially increasing in magnitude enormously, as can be seen by executing this code from the command prompt (after running my solution program below at least once, to initialize all code and data):

```
>INPUT X @ SCI 4 @ FOR M=1 TO 10 @ LN(X)^M/(M*FACT(M)*FNZ(M+1)); @ NEXT M @@ STD

    X         Term 1    Term 2    Term 3    Term 4     Term 5     ...  Term 9     Term 10
---------------------------------------------------------------------------------------
? 1E-12   -> -1.6798E1 1.5878E2 -1.0828E3 5.8556E3  -2.6386E4 ...   -2.8717E6   7.1448E6
? 1E-100  -> -1.3998E2 1.1027E4 -6.2664E5 2.8239E7  -1.0604E9  ... -5.5655E14 1.1539E16
? 1E-499  -> -6.9850E2 2.7457E5 -7.7861E7 1.7508E10 -3.2807E12 ... -1.0676E21 1.1045E23
```

The terms ultimately tend to zero as the *factorial* in the denominator dominates over the powers of the (large) negative *logarithm* in the numerator, but adding and subtracting the initial very large terms to eventually get a very small result incurs in enormous rounding errors and/or would require heavy multiprecision computations, so ultimately *Gram series* is no good either.

What to do ? We must search for *another*, better suited equivalent formula, that's what, and we can do it in just three logical steps, as follows:

1) In my *OP*, I mention an old article of mine which includes a *prime-counting function*, so a quick, easy search at my website section *HP Calculator Articles*, reveals it to be ***HP Article VA027 - Small Fry - Primes A'counting***.

2) In the article, the **FNR** prime-counting function uses **FNZ**, which is identified there as the ***Riemann Zeta*** function, so doing a *Google* search on *"Prime counting function Riemann zeta function"* brings as one of the very first hits the link *"**Prime-counting function**" - Wikipedia*.

3) Clicking on that link, we find under the section ***Formulas for prime-counting functions*** the paragraph:

> *"Folkmar Bornemann proved [...] that*
>
> $$R(e^{-2\pi t}) = \frac{1}{\pi}\sum_{k=1}^{\infty}\frac{(-1)^{k-1}t^{-2k-1}}{(2k+1)\,\zeta(2k+1)} + \frac{1}{2}\sum_{\rho}\frac{t^{-\rho}}{\rho\cos(\pi\rho/2)\,\zeta'(\rho)}$$
>
> *where **ρ** runs over the non-trivial zeros of the Riemann zeta function and t>0."*

which includes a formula that's also *equivalent* to the previous ones but whose argument is $e^{-2\pi t}$ and consisting of two infinite summations in terms of ***t***, valid for ***t > 0***. This means that ***t*** essentially acts as a *logarithm* and so the formula can be used to evaluate ***R(x)*** for *incredibly small* values of ***x***, as needed.

> **Note**: I noticed that the second summation, where ***ρ*** runs over *all* non-trivial complex zeros of ***ζ*** (which so far appear in *conjugate* pairs,) can be shortened to use just *one* zero from each pair, as the imaginary parts of both terms would cancel out, while the real parts are identical and would double in value, thus cancelling the ***1/2*** factor. This cuts the number of terms in half (thus doubling the speed) and the second summation gets simplified to this (the big ***R*** stands for *real part* and the big ***I*** stands for *imaginary part*):
>
> $$\sum_{\rho:\Im\rho>0}\Re\left(\frac{t^{-\rho}}{\rho\cos(\pi\rho/2)\,\zeta'(\rho)}\right)$$

Once we have this formula, solving ***Problem 5*** is now just a matter of implementing it in any of the languages available in *HP* vintage calcs, which might seem difficult at first sight but it merely takes some work to either evaluate the *special functions* appearing in the formula or getting their values from references, without computation.

The special functions and values appearing in the formula are ***Riemann's ζ*** function for *integer* arguments, its *non-trivial complex zeros* and the *complex* values of its first derivative ***ζ'*** at those zeros. After a little experimentation, it turns out that to achieve *12-digit* accuracy only the *real* values ***ζ(2)*** to ***ζ(8)***, the *non-trivial complex zeros* ρ_1 - ρ_6 *of* ***ζ*** and the corresponding *complex* values of ***ζ'*** at those zeros are needed and can be taken from references. Assuming your vintage *HP* calc has decent *complex-math* capabilities, the rest is easily computed.

My original solution

My *original solution* is this *15-line, 897-byte* **HP-71B** program:

```
10 DESTROY ALL @ OPTION BASE 1 @ DIM Z(8),Z0(6),K,L,P,R,S,T,U,V,W,Y @ COMPLEX Z1(6),C

20 DATA 0,PI^2/6,1.20205690316,PI^4/90,1.03692775514,PI^6/945,1.00834927738,PI^8/9450
30 DATA 14.1347251417,21.0220396388,25.0108575801,30.4248761259,32.9350615877,37.5861781588
40 DATA (.783296511867,.124699829748),(1.10929556346,-.248729788516)
50 DATA (1.29579560501,.450036709438),(1.12013084524,-.667509469349)
60 DATA (1.16057006749,.750554150342),(1.85346624998,-.561004420496)

70 READ Z,Z0,Z1 @ A=2 @ M=SGN(FNR(A)) @ FOR B=500 TO 20000 STEP 500 @ N=SGN(FNR(B))
80 IF N#M THEN FIX 8 @ DISP "t:";FNROOT(A,B,FNR(FVAR));"-> X=";FNX$(RES,8);", R=";FVALUE @ M=N
90 A=B @ NEXT B

100 DEF FNR(T) @ S=0 @ K=0 @ REPEAT @ V=S @ L=2*K+3 @ S=S+(-1)^K*T^(-L)/(L*FNZ(L))
110 K=K+1 @ UNTIL S=V @ R=S/PI @ S=0 @ K=0 @ REPEAT @ K=K+1 @ C=(.5,Z0(K)) @ V=S
120 S=S+REPT(T^(-C)/(C*COS(PI/2*C)*Z1(K))) @ UNTIL S=V @ FNR=R+S

130 DEF FNZ(N) @ IF N<9 THEN FNZ=Z(N) @ END ELSE IF N>37 THEN FNZ=1 @ END
```

```
140 Y=0 @ P=1 @ REPEAT @ P=P+1 @ W=Y @ Y=Y+P^(-N) @ UNTIL Y=W @ FNZ=Y+1

150 DEF FNX$(T,N) @ STD @ T=-2*PI*T/LN(10) @ FNX$=STR$(10^(FP(T)+1))[1,N]&"E"&STR$(INT(T))
```

Note: keywords **FNROOT**, **FVAR**, **FVALUE**, **COMPLEX** and **REPT** are from the ***Math ROM***; **REPEAT** and **UNTIL** are from the ***JPC ROM***.

Line 10 performs some initialization and defines all arrays and certain variables. In particular, **DIM ...,K,L,...,W,Y** and **COMPLEX ...,C** are necessary to create those variables *here* and not inside the **DEF** function definitions because of a known system bug. See this post for further details.

Lines 20-60 define all necessary data, which includes an initial dummy *0* and the values of ***ζ(2)*** .. ***ζ(8)*** in line *20*, the imaginary parts of the first six zeros of ***ζ(z)*** in line *30*, and the six complex values of ***ζ'(z)*** at those zeros in lines *40-60*.

Lines 70-90 read all data and compute and output the seven roots x_1 .. x_7, as well as the corresponding ***t*** parameters and resulting values of ***R(x)*** at the computed roots, which are suitably near *0*.

The roots are located by sweeping the interval ***t*** = *[2, 500 .. 20000]* using steps of *500* and when a sign change is detected, **FNROOT** is used to accurately compute the root using the subinterval's extremes as initial approximations, for fast convergence.

Lines 100-120 define ***R(x)***, computing each summation until the sums stop changing, then returns the final sum of both. The first summation calls **FNZ** to compute ***ζ(n)***, while the second summation uses the complex values of ***ζ'(z)*** from the array where they were stored and various complex functions and complex arithmetic operations. **FNR** can be called from the command prompt after the program is run.

Lines 130-140 define ***ζ(n)*** for integer ***n ≥ 2***, returning the real values from the array where they were stored for ***2 ≤ n ≤ 8***, the constant ***1*** for ***n > 37***, and otherwise it computes the summation until it stops changing and returns the final sum. **FNZ** can be called from the command prompt after the program is run and will quickly return values accurate to 12 digits for all integer arguments ***n ≥ 2***.

Line 150 defines an utility string-valued function which converts the value of the parameter ***t*** to the corresponding value of ***x*** and returns it as an *N*-character mantissa plus its corresponding exponent, which can be >> ***499***.

Notes:

○ There's no need to specify **RADIANS** in the initialization because all complex trig functions always use radians regardless on the current angular mode.

○ It is possible to save *30* bytes of *RAM* and slightly reduce array **Z**'s size by including just the values of ***ζ(n)*** which are actually used, so line *20* would look like this:

```
20 DATA 0,0,1.20205690316,0,1.03692775514,0,1.00834927738
```

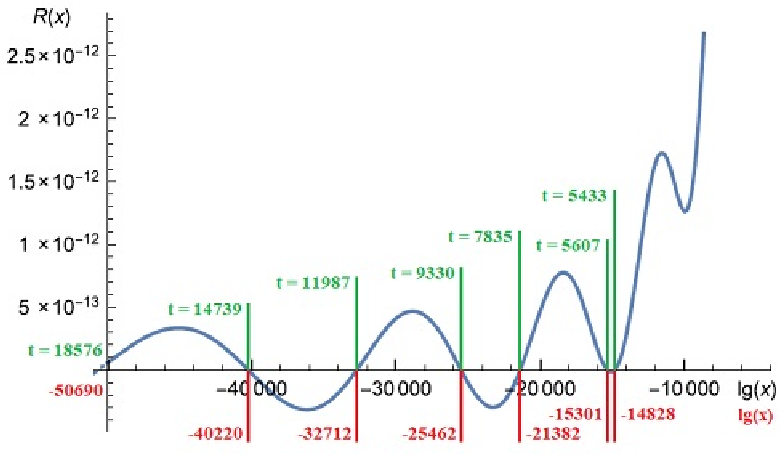
but this would break the ability to call **FNZ** from the prompt for certain values *(2, 4, 6, 8)* and would require index remapping so it isn't worth the trouble.

○ The **INT** function at line *150* can't be changed to **IP** lest you'd get *E-**14827*** instead of the correct *E-**14828***.

Let's run it !

```
>RUN

t: 5433.88846830 -> X=1.828642E-14828, R= 0
t: 5607.21036604 -> X=2.039534E-15301, R=-1.E-24
t: 7835.62497997 -> X=3.289421E-21382, R=-5.E-24
t: 9330.89271531 -> X=2.000957E-25462, R= 3.E-24
t: 11987.8440717 -> X=1.374136E-32712, R= 2.92E-23
t: 14739.1970154 -> X=2.378127E-40220, R= 2.48E-23
t: 18576.1969026 -> X=1.420375E-50690, R= 1.69E-23
```



For the first, largest root, the *46-digit* value is:

1.828643269752522610409732527318069320008652918 * 10^{-14828}

so we've got ***7*** correct mantissa digits *(save 1 ulp)* plus the correct ***5***-digit exponent.

As the **HP-71B** is a 12-digit machine we can do no better (*7d* mantissa + *5d* exponent = *12d* in all). The other six roots should have about ***6-7*** correct mantissa digits as well (give or take a few *ulp*,) plus the correct ***5***-digit exponent, of course.

For timing, execute instead:

```
>SETTIME 0 @ CALL @ TIME
```

which tells us that all 7 roots are obtained in *4.14"* in ***go71b*** at *128x*, *0.55"* in ***Emu71/Win*** at *972x*, or just ***8' 50"*** on a physical **HP-71B**.

Additional comments

● This problem is easily solved directly from the command line using recent versions of ***Mathematica***, e.g. to find the first root correct to 17 digits, simply execute:

```
10^-t /. FindRoot[RiemannR[10^-t], {t, 14000}, PrecisionGoal → 15, WorkingPrecision → 21]
```

```
1.8286432697525226 x 10^-14828
```

Then again, ***Mathematica*** is a multi-*Gb* software intended to run on powerful modern hardware, so it's quite remarkable that vintage *HP* calcs can stand their ground, evaluating the same formula which *Mathematica* also uses.

● All seven roots' exponents are so small that they can't be represented in ***Free42 Decimal*** (which uses the *Intel Decimal Floating-Point Math Library*, i.e. it uses *IEEE 754-2008* quadruple precision decimal floating-point, which has min. exponent *-6,143*,) but notice that the *7th* root's exponent, ***-50,690***, exceeds even the lower limit of the **HP-71B**'s internal *15-form* representation's exponent, which is *-50,000*, see *HP Journal July 1984*, *p.34*.

Further roots have even smaller exponents. e.g. beyond the 7 roots just computed there are 10 even smaller additional roots going down to *10$^{-500,000}$*, as seen in the graph below.

● ***Problem 5*** asked for the first *seven* roots but it's very easy to compute additional roots (albeit with diminished accuracy), e.g. to compute all roots having *5-digit-long* exponents (i.e. up to **E-99999**), we just need to raise the upper limit of the **FOR** loop in line *70* like this:

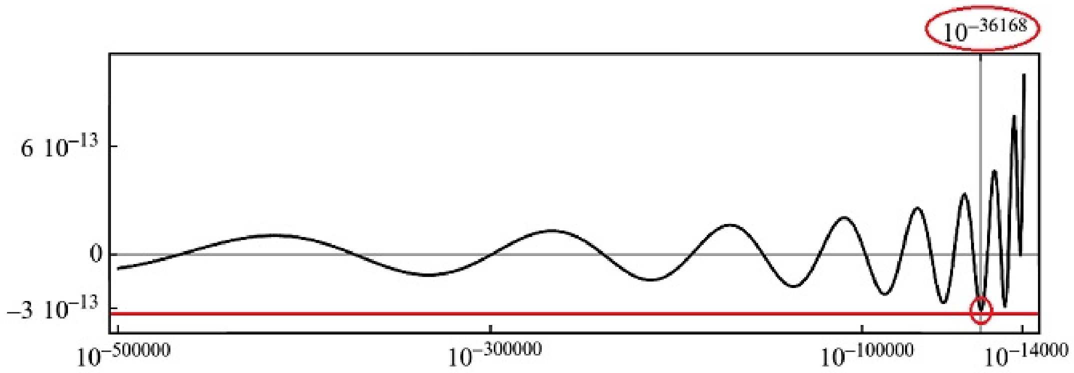
```
70 ... @ FOR B=500 TO 37000 STEP 500 @ ...

>RUN
      (... the first seven roots, and then ...)
      t: 23080.0653626 -> X=1.618632E-62980, R= 1.156E-23
      t: 28910.7052546 -> X=6.835264E-78891, R=-1.176E-23
      t: 36044.9662268 -> X=1.587852E-98358, R= 8.28E-24
```

which runs in *5.99"* in ***go71b*** at *128x*, *0.79"* in ***Emu71/Win*** at *972x*, or just ***12' 47"*** on a physical **HP-71B**.

When computing roots beyond the *3rd* one, the **STEP 500** at line *70* can be raised to **1000** for faster execution and even further as the spacing between then roots decreases more and more. For instance, using **STEP 1000** the timings for the 10 roots above are *5.32"*, *0.70"* and just ***11' 21"*** on a physical **HP-71B**.

• Once the roots have been computed, we can easily obtain additional interesting results, such as for instance the *unique* real positive value of ***x*** where ***R(x)*** tentatively attains its *global minimum* value, as seen in this graph:



To compute the value of such ***x*** and the *global minimum*, simply insert the following program line and run it *(now the program will be 967 bytes long)*:

```
95  D=FNROOT(11987,14739,FNR(FVAR+.5)-FNR(FVAR-.5)) @ DISP "Min: ";FNR(D);"at ";FNX$(D,6)

    >RUN 95
              Min: -3.14748471571E-13 at 1.1088E-36168
```

Notes:

○ The expression computing ***x*** must be run as a *program line* because **FNROOT** can't be executed from the command prompt if it calls a user-defined function, **FNR** in this case. Attempting to do so results in an error issued by the ***Math ROM***, namely *Error 6*, `Kybd FN in FNROOT/INTEGRAL`.

○ The global minimum is located by finding a root of ***R'(x)*** between the limits discussed below, which is computed with good accuracy by using a very simple numerical approximation to the *derivative*. The resulting global minimum is accurate to *10-12* digits while the corresponding ***x*** is accurate to about *5* digits plus the fully correct *5*-digit exponent.

○ From the above graph (or a tabulation of values) we notice that the *global minimum* is located between the *5th* and *6th* roots, so we use the integer parts of their ***t*** parameters *(11987 and 14739, respectively)* as the initial approximations for faster convergence.

Well, I hope you enjoyed my solution and comments to ***Problem 5***, even if you didn't enjoy the problem itself, which I very much did.

Actually, I think it's indeed a problem with a *most amazing, awesome result*, where a simple-looking function in its canonical *(and Gram series, too)* form, with no *ad-hoc* constants or contrivances and whose graph is a dull logarithmic-like curve for *macroscopic* arguments, suddenly turns out *wiggly* and *crosses* the *X*-axis by the *tiniest* amount when evaluated at *"nanoscopic"* arguments *(0.00000...{14,000+ zeros}...0000018286...)*, completely *invisible* and utterly *unexpected*.

In my humble opinion, this ***does*** add a measure of *"magic"* and *mystery* to ***Mathematics***, and one wonders what other as-yet-undiscovered marvels might be lurking out there... :-)

Next will be ***Problem 6***, which will conclude my months-long six-pronged ***SCR #12 - Then and Now*** thread once its main point has been thoroughly made.

V.

26th February, 2023, 22:58 **Post: #8**

Dave Frederickson

Senior Member

Posts: 2,139
Joined: Dec 2013

RE: [VA] SRC #012e - Then and Now: Roots

Fernando del Rey Wrote: (19th February, 2023 19:37)

It would be interesting to get an idea of timing on a physical 71B.

In the Emu71 Settings, check Authentic Calculator Speed.

EMAIL PM FIND QUOTE REPORT

26th February, 2023, 23:54 **Post: #9**

EdS2
Senior Member

Posts: 517
Joined: Apr 2014

RE: [VA] SRC #012e - Then and Now: Roots

Bravo Valentin for the challenge, and thanks for the writeup. As you note, a challenging challenge, and not many visible attempts.

"nanoscopic" is a nice word!

Thanks also for the pointer to the representation used for PROOT in the Math Pac - very interesting.

EMAIL PM FIND QUOTE REPORT

28th February, 2023, 11:08 **Post: #10**

J-F Garnier
Senior Member

Posts: 790
Joined: Dec 2013

RE: [VA] SRC #012e - Then and Now: Roots

This has been a difficult problem for me (at different levels) that seems so simple to solve when reading Valentin's solution.
As I often noticed, carefully reading the problem statement is already a part of the solution. And indeed, contrary to most of Valentin's problems, using the Web resources was not explicitly prohibited here. I missed that aspect at the beginning.
This was the clue that we really needed: don't limit our attempts to our own (limited) resources, think *wider*.

> **Valentin Albillo Wrote:** (26th February, 2023 02:34)
>
> Actually, I think it's indeed a problem with a *most amazing, awesome result*, where a simple-looking function in its canonical *(and Gram series, too)* form, with no *ad-hoc* constants or contrivances and whose graph is a dull logarithmic-like curve for *macroscopic* arguments, suddenly turns out *wiggly* and *crosses* the *X*-axis by the *tiniest* amount when evaluated at *"nanoscopic"* arguments *(0.00000...{14,000+ zeros}...0000018286...)*, completely *invisible* and utterly *unexpected*.

In my first investigations, I tried to evaluate the summation (the Gram series) without the zeta factors, considering that they quickly tend to 1 and should have little impact, so I may have a first estimation.
Unfortunately, that summation doesn't tend to zero anymore, actually not even to a finite limit when X tend to zero. So my attempt failed.
This was a major surprise to me, much more than the tiny roots: the limit to zero is due to these zeta terms. It may have a deep mathematic meaning but it's out of my understanding.

J-F

EMAIL PM WWW FIND QUOTE REPORT

1st March, 2023, 02:36 (This post was last modified: 2nd March, 2023 03:03 by Valentin Albillo.) **Post: #11**

Valentin Albillo
Senior Member

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

RE: [VA] SRC #012e - Then and Now: Roots

.
Hi, **Jean-François**,

> **J-F Garnier Wrote:** (28th February, 2023 11:08)
>
> This has been a **difficult** problem for me (at different levels) that **seems so simple** to solve when reading Valentin's solution.

It wasn't nearly as *"impossible"* as some came to believe, and the suggested equivalent formula was readily found in *Wikipedia*, among other places. But I concede that it *seemed* difficult and the subject matter was probably deemed heavy math and thus unappealing to most current denizens of the forum. As I said, we're not in the old forum anymore.

But at least *you did succeed* in providing a correct solution, the only one who did (as the usual suspects stayed mum this time,) and I thank you for it because I hate it when one of my challenges gets unanswered, a dry run, as it's happened a very few times.

Incoming ***Problem* 6** will be the hardest of the lot but boasting a very *different* kind of "hardness" as compared to this previous *Problem 5*: it'll be appreciably *lighter* on the math but *harder* on the programming, for variety.

If you *also* succeed in solving it, I'll officially declare you the *"winner"* of this one-of-a-kind **SRC #12** ! *(not that it was a contest of sorts ...* 😃 😃 *)*

Best regards.
V.