HP Forum Archive 21

Riemann's Zeta Function (HP 50g)

Message #1 Posted by ***Gerson W. Barbosa*** *on 4 Dec 2012, 3:59 p.m.*

This is a follow-up of a recent thread on the Zeta function . On the occasion Thomas Ritschel wrote an RPL program based on Euler-MacLaurin summation, which was much faster and efficient than my previous attempts. The following is based on the same method, as described in the book *Riemann's Zeta Function*, by H. M. Edwards (pages 114 through 118).

```
Zeta:


%%HP: T(3)A(D)F(.);
\<< RCLF SWAP -22. SF RAD 1. CF DUP RE 0. <
  IF
  THEN 1. SF NEG 1. +
  END 10. 8. 0. \-> s n m b
  \<< '\GS(k=1.,n-1.,k^-s)' EVAL n 1. s - ^ s 1. - / + n s NEG ^ 2. / + BNL 1. 2. m
    FOR i GETI s i + 1. - GAMMA * n 1. i - s - ^ * s GAMMA / i ! / 'b' STO+ 2.
    STEP b NIP NIP + 1. FS?
    IF
    THEN 1. s - DUP 2. / \pi * SIN 2. ROT ^ * * s DUP GAMMA \pi ROT NEG ^ * *
    END
  \>> SWAP STOF
\>>


BNL:


%%HP: T(3)A(D)F(.);
{ .166666666667 -3.33333333333E-2 2.38095238095E-2 -3.33333333333E-2 7.57575757576E-2
  -.253113553114 1.16666666667 -7.09215686275 54.9711779449 -529.124242424
  6192.12318841 -86580.2531136 1425517.16667 -27298231.0678 601580873.901
  -15116315767.1 429614643062. -1.37116552051E13 4.88332318973E14 -1.92965793419E16
  8.41693047575E17 -4.03380718541E19 2.11507486381E21 -1.20866265223E23 7.50086674608E24 }
```

```
%%HP: T(3)A(D)F(.);
\<< { } 2. 50.
  FOR n n IBERNOULLI \->NUM + 2.
  STEP 'BNL' STO
\>>
```

This produces the list with the first twenty-five even-order Bernoulli numbers above. The Zeta program above uses only the first four (n=8), which is suffices for the examples below. However complex arguments will require more terms as their modules grow larger, as we'll see in other examples.

```
Values of Zeta(s) for some real and complex arguments:


    s    |        Zeta(s)        |  time(s) |    exact value
---------+-----------------------+----------+-----------------
  8.     |    1.00407735620      |   0.487  |     pi^8/9450
  7.     |    1.00834927741      |   0.489  |
  6.     |    1.01734306199      |   0.493  |     pi^6/945
  5.     |    1.03692775515      |   0.484  |
  4.     |    1.08232323370      |   0.488  |     pi^4/90
  3.     |    1.20205690316      |   0.481  |
  2.     |    1.64493406685      |   0.479  |     pi^2/6
  1.     |    9.99999999999E499  |   0.476  |     infinite
  0.5    |   -1.46035450882      |   0.701  |
  0      |   -0.500000000000     |   0.388  |    -1/2
 -0.5    |   -0.207886224977     |   0.785  |
 -1.     |   -8.33333333331      |   0.521  |    -1/12
 -2      |    4.00788420761E-15  |   0.523  |     0
 -3      |    8.33333333323      |   0.535  |     1/120
 -4      |   -2.10179397707E-15  |   0.537  |     0
 -5      |   -3.96825396829E-3   |   0.543  |    -1/252
 -6      |   -2.3297903572E-15   |   0.537  |     0
 -7      |    4.16666666665E-3   |   0.540  |     1/240
 (1.,2.) |   (0.598165569758,    |   3.512  |
         |   -0.351854745216)    |          |
 (3.,4.) |   (0.890554906967,    |   3.375  |
         |   -8.07594542632E-3)  |          |
(-5.,6.) |   (1.42843389347,     |   3.504  |
         |    0.184238641590)   |          |
```

Let us experiment with different numbers of zeta terms (n) and B_n terms (m/2) of the Euler-MacLaurin summation for a few arguments in the positive complex half-plane and observe their influence on the overall accuracy using the program below.

```
%%HP: T(3)A(D)F(.);
\<< 0. \-> s n m b
```

```
  \<< '\GS(k=1.,n-1.,k^-s)' EVAL n 1. s - ^ s 1. - / + n s NEG ^ 2. / + BNL 1. 2. m
    FOR i GETI s i + 1. - GAMMA * n 1. i - s - ^ * s GAMMA / i ! / 'b' STO+ 2.
    STEP b NIP NIP +
  \>>
\>>


        s         |  n |  m |            Zeta(s)                 |  time(s)
------------------+----+----+------------------------------------+----------
        2         |  5 |  4 |           1.64493377778            |   0.330
        2         |  5 |  8 |           1.64493406547            |   0.438
        2         |  7 |  4 |           1.64493403873            |   0.371
        2         |  7 |  8 |           1.64493406681            |   0.483
        2         |  8 | 10 |           1.64493406685            |   0.521
        2         | 10 |  8 |           1.64493406685            |   0.464
       0.5        |  4 |  4 |          -1.46035496134            |   0.394
       0.5        |  4 |  6 |          -1.46035444845            |   0.491
       0.5        |  4 |  8 |          -1.46035451114            |   0.586
       0.5        |  8 | 10 |          -1.46035450881            |   0.743
       0.5        | 10 |  8 |          -1.46035450881            |   0.684
    (0.5,18)      |  6 |  8 | (2.32919185455,-0.188602491388)    |   3.545
    (0.5,18)      | 12 | 16 | (2.32915487303,-0.188866005798)    |   7.075
    (0.5,18)      | 20 | 20 | (2.32915487307,-0.188866005802)    |   9.247
   (2.5,-100)     | 12 | 16 | (1.02952895989,5.76320966153E-2)   |   6.753
   (2.5,-100)     | 25 | 22 | (1.13221432860,3.98766176236E-2)   |   9.704
   (2.5,-100)     | 36 | 28 | (1.13221432259,3.98766168104E-2)   |  12.049
(0.5,14.13472514) | 20 | 16 | (2.185000000E-10,-1.35802000E-9)   |  12.049
```

Here is the equivalent HP-71B BASIC program. Because GAMMA() on the HP-71B doesn't handle complex arguments, it will work only on the real domain.

```
10 DESTROY ALL @ OPTION BASE 1
12 REAL S,Z,B(15)
14 INTEGER I,N,M
16 INPUT S,N,M
18 FOR I=1 TO 15
20   READ B(I)
22 NEXT I
24 Z=0
26 FOR I=1 TO N-1
28   Z=Z+I^(-S)
30 NEXT I
32 Z=Z+N^(1-S)/(S-1)+N^(-S)/2
34 FOR I=2 TO M STEP 2
36   Z=Z+B(I DIV 2)*GAMMA(S+I-1)*N^(-S-I+1)/(GAMMA(S)*FACT(I))
38 NEXT I
```

```
40 PRINT Z
42 END
44 DATA 1/6,-1/30,1/42,-1/30,5/66,-691/2730,7/6,-3617/510,43867/798,-174611/33
46 DATA 854513/138,-236364091/2730,855303/6,-23749461029/870,601580873.901


>run
? 2,5,4
 1.64493377777
>run
? 0.5,4,4
-1.46035496134
>run
? 2,5,8
 1.64493406546
>run
? 0.5,4,8
-1.46035451114
```

The following program HP-71B program will accept both real and complex arguments:

```
10 DESTROY ALL @ OPTION BASE 1
12 COMPLEX P,S,Z
14 REAL B(15)
16 INTEGER I,N,M
18 INPUT S,N,M
20 FOR I=1 TO 15
22   READ B(I)
24 NEXT I
26 Z=0
28 FOR I=1 TO N-1 @ Z=Z+I^(-S) @ NEXT I
30 Z=Z+N^(1-S)/(S-1)+N^(-S)/2
32 P=1/(S-1)
34 FOR I=2 TO M STEP 2
36   P=P*(S+I-3)*(S+I-2)
38   Z=Z+B(I DIV 2)*P*N^(-S-I+1)/FACT(I)
40 NEXT I
42 IF IMPT(S)=0 THEN PRINT REPT(Z) ELSE PRINT Z
46 END
48 DATA 1/6,-1/30,1/42,-1/30,5/66,-691/2730,7/6,-3617/510,43867/798,-174611/33
50 DATA 854513/138,-236364091/2730,855303/6,-23749461029/870,601580873.901


>run
? 2,5,4
 1.64493377777
>run
```

```
? 0.5,4,4
-1.46035496134
>run
? (0.5,18),6,8
 (2.32919185455,-.188602491387)
```

Both programs require the Math Module. I haven't tested them on the real hardware, but the running time should be fast enough.

Edited: 4 Dec 2012, 4:14 p.m.

Re: Riemann's Zeta Function (HP 50g)

Message #2 Posted by ***Paul Dale*** *on 5 Dec 2012, 2:57 a.m.,*
in response to message #1 by Gerson W. Barbosa

Interesting work.

Has anyone here tried the algorithms from Unified algorithms for polylogarithm, L-series, and zeta variants?

- Pauli

Re: Riemann's Zeta Function (HP 50g)

Message #3 Posted by ***Valentin Albillo*** *on 5 Dec 2012, 4:35 a.m.,*
in response to message #1 by Gerson W. Barbosa

Hi, Gerson:

Interesting work, I'll try and get some free time one of these next days (4 vacational days ahead) to try your routine, but in the meantime some well-meant, efficiency-related comments re your HP-71B versions of the code (not criticism, of course):

1) line 16 INTEGER I,N,M

INTEGER variables are slower to process and operate with than REAL variables in HP-71B BASIC, so unless memory must be saved at all costs (and with escalar, non-array variables that's never the case) using INTEGER instead of REAL is counterproductive in terms of speed.

2) lines 20 FOR I=1 TO 15, 22 READ B(I), 24 NEXT I

A FOR-NEXT loop isn't necessary to read the elements of an array in HP-71B BASIC, a simple 20 READ B will do and it will be both neater and much faster.

3) in the FOR-NEXT loops that follow, there are many invariant expressions within the loops. They should be computed and saved into variables before the loop, then use the variable inside the loop. It will be faster in general.

4) line 42 IF IMPT(S)=0 THEN PRINT REPT(Z) ELSE PRINT Z

The comparison to 0 isn't necessary, it would be more efficient and faster like this:

42 IF IMPT(S) THEN PRINT Z ELSE PRINT REPT(Z)

5) line 46 END

Not actually needed. Same with line 26 Z=0, which isn't needed after the DESTROY ALL that precedes it.

Best regards from V.

Re: Riemann's Zeta Function (HP 50g)

Message #4 Posted by ***Gerson W. Barbosa*** *on 5 Dec 2012, 8:44 a.m.,*
in response to message #3 by Valentin Albillo

Hello Valentin,

Thank you very much for your suggestions. I have only skin deep knowledge of the HP-71B system. I decided to use it before writing the HP 50g program because testing algorithms is easier in BASIC than in RPL, at least for me. I have an HP-71B with MATH ROM, 32K module and card reader, but I haven't tested the programs on it yet. I suspect the two calls to the Gamma function in the last loop takes longer than the alternative two multiplication operations, even for a large number of iteration, but I choose it for compactness. Also, pre-computed Bernoulli numbers is mandatory on the HP-71B, but when only four of them are needed perhaps the built-in BERNOULLI function on the HP 50g would be faster. Anyway, half a second running time for arguments in the limited range most users would be interested in fine for me. That's about the time trigs took on the HP-35 :-) I have the book in the reference since 2009, but I never read past the first chapter (The math is too complicated for me).

Any improvement you want to try and share, such as automatic choice of parameter for the maximum possible range, is much appreciated.

Best regards,

Gerson.

Re: Riemann's Zeta Function (HP 50g)

*Message #5 Posted by **Thomas Ritschel** on 5 Dec 2012, 9:39 a.m.,*
in response to message #4 by Gerson W. Barbosa

Hello Gerson,

thanks again for sharing your work. It's interesting to see that you got the algorithm much faster by precomputing the Bernoulli numbers.

Meanwhile I tried the Borwein algorithm. There is also a related paper by Xavier Gourdon and Pascal Sebah.

For some initial tests I prepared this little Python script:

```
#!/usr/bin/python
# computes the zeta function for complex arguments
# using Borwein's "Algorithm 2"


import math, cmath, sys


s = complex(float(sys.argv[1]),float(sys.argv[2]))


def d_k(k,n):
    y = 0.0
    for i in range(0,k+1):
        y += math.factorial(n+i-1)*(4.0**i)/(math.factorial(n-i)*math.factorial(2*i))
    y *= n
    return y


def zeta2(s,n):
    y = 0.0
    for k in range(0,n):
        y += ((-1)**k) * (d_k(k,n)-d_k(n,n)) / (k+1)**s
    y *= (-1)/(d_k(n,n) * (1-2**(1-s)))
    return y


for n in range(1,25):
    print n, zeta2(s,n)
```

It turns out that for 12 digits precision one needs about n=12-15. This is in accordance with n=1.3d+0.9t suggested by Gourdon and Sebah ("d" is the number of digits, "t" is the absolute value of the imaginary part of the argument).

On the HP50g the same algorithm as given above takes roughly half a minute for n=14, which is quite slow. Perhaps, one could try some optimization by unrolling the double loop, or at least reducing the number of "d_k" evaluations.

Borwein presents in his paper also a "simpler" algorithm (dubbed "algorithm 3") using binomial coefficents. But there are still double sums, which makes it quite slow for n>10.

Perhaps I should consider this as a challenge to dive a little deeper into SysRPL, in which I have almost no experience so far...

Best regards,

Thomas.

Re: Riemann's Zeta Function (HP 50g)

Message #6 Posted by ***Paul Dale*** *on 5 Dec 2012, 4:09 p.m.,*
in response to message #5 by Thomas Ritschel

The 34S uses the Borwein algorithm. We found that the error bounds he provides are too tight and more iterations are actually required. However it is still the fastest algorithm I uncovered at the time. The other problem is this algorithm takes O(y) iterations to converge for a complex argument x + *i*y. This is why the 34S doesn't support complex zeta even though I did code it.

- Pauli

Re: Riemann's Zeta Function (HP 50g)

Message #7 Posted by ***Gerson W. Barbosa*** *on 5 Dec 2012, 5:37 p.m.,*
in response to message #6 by Paul Dale

> Quote:
>
> This is why the 34S doesn't support complex zeta even though I did code it.

This is the answer to the question I was going to ask you. Thank you!

Gerson.

Re: Riemann's Zeta Function (HP 50g)

Message #8 Posted by ***Paul Dale*** *on 5 Dec 2012, 10:49 p.m.,*
in response to message #7 by Gerson W. Barbosa

Knowing that the function will execute in reasonable time is kind of nice. I couldn't guarantee this with complex zeta unfortunately.

- Pauli

Re: Riemann's Zeta Function (HP 50g)

Message #9 Posted by ***Angel Martin*** *on 6 Dec 2012, 1:39 a.m.,*
in response to message #8 by Paul Dale

Yes but even if slow it's a pleasure to obtain those results. The 41Z has it implemented in ZZETA, it works like a charm on the CL.

Re: Riemann's Zeta Function (HP 50g)

Message #10 Posted by ***Paul Dale*** *on 6 Dec 2012, 1:52 a.m.,*
in response to message #9 by Angel Martin

Someone will try to evaluate 3 + i 1e25...

- Pauli

Re: Riemann's Zeta Function (HP 50g)

Message #11 Posted by ***Angel Martin*** *on 6 Dec 2012, 3:02 p.m.,*
in response to message #10 by Paul Dale

easy: "DATA ERROR" obviously ZZETA is not up to this challenge :-)

http://www.wolframalpha.com/input/?i=polylog%283+%2B+i+1e25%29

Re: Riemann's Zeta Function (HP 50g)

Message #12 Posted by ***Gerson W. Barbosa*** *on 6 Dec 2012, 3:15 p.m.,*
in response to message #11 by Angel Martin

Neither is Zeta above, which cannot properly handle 0.5+18i :-)

Re: Riemann's Zeta Function (HP 50g)

*Message #13 Posted by **Angel Martin** on 7 Dec 2012, 12:59 a.m.,*
in response to message #12 by Gerson W. Barbosa

ZZETA obtains in a blitz (on the CL) for that argument:

18, ENTER, .5, ZZETA --> 2,329-J0,189

and using FIX9 settings:

Im = -0,188866015
Re = 2,329154846

not bad for a venerable platform.

Re: Riemann's Zeta Function (HP 50g)

*Message #14 Posted by **Gerson W. Barbosa** on 7 Dec 2012, 5:20 a.m.,*
in response to message #13 by Angel Martin

The first HP 50g program above, with fix parameters n=10 and m=8, gets the first 5 or 6 digits right:

```
(2.32915540368,-0.188867886498)
```

This takes about 4 seconds. As Im(s) grows larger, the accuracy degrades significantly. The actual 12-digit result is

```
(2.32915487305,-0.18886600580),
```

which means your best HP-41 result is really not bad:

```
(2.329154846,-0.188866015)
```

Re: Riemann's Zeta Function (HP 50g)

*Message #15 Posted by **Thomas Ritschel** on 7 Dec 2012, 7:29 a.m.,*
in response to message #14 by Gerson W. Barbosa

Using n=30 the Borwein algorithm (see the program in message #18) yields:

```
zeta(0.5+18i) -> (2.32915487306,-0.188866005808)
```

This took about 4 seconds on a HP 49g+.

Re: Riemann's Zeta Function (HP 50g)

Message #16 Posted by ***Gerson W. Barbosa*** *on 7 Dec 2012, 7:53 a.m.,*
in response to message #15 by Thomas Ritschel

That's pretty good! I think it will be the only Zeta program in my 50g. Thanks again!

Re: Riemann's Zeta Function (HP 50g)

Message #17 Posted by ***Gerson W. Barbosa*** *on 5 Dec 2012, 5:34 p.m.,*
in response to message #5 by Thomas Ritschel

Hello Thomas,

> Quote:
>
> thanks again for sharing your work.

It's me who should thank you again. This is essentially the work you have started. I only recoded it following the text and examples in my reference.

> Quote:
>
> It's interesting to see that you got the algorithm much faster by precomputing the Bernoulli numbers.

Probably the idea wouldn't have occurred to me if I hadn't decided to test the algorithm on the HP-71B first. When doing it I noticed the computation of B_2, B_4,...B_{10} alone would take about one minute.

> Quote:
>
> Meanwhile I tried the Borwein algorithm. There is also a related paper by Xavier Gourdon and Pascal Sebah.

That's the way to go. This is the same method used in SandMath, I'm told. Please share your HP 50g when it's finished.

> Quote:

> Perhaps I should consider this as a challenge to dive a little deeper into SysRPL, in which I have almost no experience so far...

Besides speed another advantage of SysRPL is the ability of doing all calculations with 15 significant digits, I think. I don't know anything about SysRPL, though.

Best regards,

Gerson.

Re: Riemann's Zeta Function (HP 50g)

*Message #18 Posted by **Paul Dale** on 5 Dec 2012, 10:40 p.m.,*
in response to message #17 by Gerson W. Barbosa

> Quote:
>
> Besides speed another advantage of SysRPL is the ability of doing all calculations with 15 significant digits, I think.

SYSRPL has the long real and long complex types which do carry extra digits -- I think fifteen but could be mistaken.

\- Pauli

Re: Riemann's Zeta Function (HP 50g)

*Message #19 Posted by **Thomas Ritschel** on 6 Dec 2012, 4:27 a.m.,*
in response to message #17 by Gerson W. Barbosa

Hello Gerson,

here comes a (UserRPL) implementation of the Borwein algorithm for the HP 50g. Actually it's the Gourdon-Sebah variant ("Proposition 1" in their paper). Note that their definition of "d_k" is slightly different to Borwein's.

```
DK:


%%HP: T(3)A(R)F(.);
\<< \-> k n
  \<< 0 k n
    FOR i n i + 1 - !
```

```
4 i ^ * n i - ! 2 i *
! * / +
    NEXT n *
  \>>
\>>
```

```
Zeta:


%%HP: T(3)A(R)F(.);
\<< \-> s n
  \<< 0 1 n
    FOR k -1 k 1 - ^
k n DK * k s ^ / +
    NEXT 1 0 n DK 1 2
1 s - ^ - * / *
  \>> EVAL \->NUM
\>>
```

The above version is more or less just a proof of concept. For n=10 or more iterations it is rather slow.

However, it can be improved by precomputing the "d_k". The following routine generates the list of "d_k" for a given n. Just enter the number of iterations, e.g. n=20, and run "DKI":

```
DKI:


%%HP: T(3)A(R)F(.);
\<< \-> n
  \<< { } 0 n
    FOR i i n DK EVAL
+
    NEXT
  \>> 'DKL' STO
\>>
```

The list is stored in "DKL" (for "d_k list") and is then used by the following program. It automatically takes the number of iterations from the size of the "DKL" list.

```
Zeta2:


%%HP: T(3)A(R)F(.);
\<< PUSH -105 SF DKL
SIZE 1 - \-> s n
  \<< DKL 1 GETI 1 2 1
```

```
s - ^ - * INV NEG 3
ROLLD 1 n
    FOR k GETI -1 k ^
* k s ^ / 3 ROLLD
    NEXT DROP2 2 n
    FOR i +
    NEXT *
  \>> EVAL \->NUM POP
\>>
```

It should work for real and complex arguments and is reasonable fast. I did no timings, but it "feels" like less than 2 seconds for 20 iterations. Most probably the EVAL and \->NUM steps could be removed.

Best regards,

Thomas

Re: Riemann's Zeta Function (HP 50g)

Message #20 Posted by ***Gerson W. Barbosa*** *on 8 Dec 2012, 12:16 p.m.,*
in response to message #3 by Valentin Albillo

Hello Valentin,

Quoting myself above:

> Quote:
>
> The following is based on the same method, as described in the book *Riemann's Zeta Function*, by H. M. Edwards (pages 114 through 118)

Dover Publications, Inc. has kindly granted me permission to reproduce the following excerpt for use by forum members of this museum:

http://i918.photobucket.com/albums/ad30/gwbarbosa/Temp/Zeta_E-McL_Sum_1.png
http://i918.photobucket.com/albums/ad30/gwbarbosa/Temp/Zeta_E-McL_Sum_2.png
http://i918.photobucket.com/albums/ad30/gwbarbosa/Temp/Zeta_E-McL_Sum_3.png
http://i918.photobucket.com/albums/ad30/gwbarbosa/Temp/Zeta_E-McL_Sum_4.png
http://i918.photobucket.com/albums/ad30/gwbarbosa/Temp/Zeta_E-McL_Sum_5.png

```
Riemann's Zeta Function, by H. M. Edwards. Dover Publications (2001), ISBN 10: 0486417409, pages 114 through 118.
```

I think this book might be useful to you and other readers who have stronger math backgrounds than I do.

Best regards,

Gerson.

Re: Riemann's Zeta Function (HP-28S & HP-48)

Message #21 Posted by ***Gerson W. Barbosa*** *on 17 Dec 2012, 8:49 p.m.,*
in response to message #20 by Gerson W. Barbosa

Implementation of the above method on the HP-28S and HP-48 calculators:

```
Zeta (HP-48)
%%HP: T(3)A(D)F(.);
\<< DUP 1 - INV 10 8 0 \-> s p n m b
  \<< 0 1 n 1 -
    FOR k k s NEG ^ +
    NEXT n 1 s - ^ s 1 - / + n s NEG ^ 2 / + BNL 1 2 m
    FOR i s i + 3 - DUP 1 + * 'p' STO* GETI p * n 1 i - s - ^ * i ! / 'b' STO+ 2
    STEP b ROT ROT DROP2 +
  \>>
\>>


Zeta (HP-28S)
\<< DUP 1 - INV 10 8 0 \-> s p n m b
  \<< 0 1 n 1 -
    FOR k k s NEG ^ +
    NEXT n 1 s - ^ s 1 - / + n s NEG ^ 2 / + BNL 1 2 m
    FOR i s i + 3 - DUP 1 + * p * 'p' STO GETI p * n 1 i - s - ^ * i FACT / b + 'b' STO 2
    STEP b ROT ROT DROP2 +
  \>>
\>>


BNL
{ .166666666667 -3.33333333333E-2 2.38095238095E-2 -3.33333333333E-2 7.57575757576E-2
  -.253113553114 1.16666666667 -7.09215686275 54.9711779449 -529.124242424
  6192.12318841 -86580.2531136 1425517.16667 -27298231.0678 601580873.901
  -15116315767.1 429614643062. -1.37116552051E13 4.88332318973E14 -1.92965793419E16
  8.41693047575E17 -4.03380718541E19 2.11507486381E21 -1.20866265223E23 7.50086674608E24 }


Re(s) >= -1; small Im(s)
```

For larger complex arguments, edit the program and change the n and m parameters as needed (up to m=50). The default values (n=10, m=8) should give maximum accuracy for real arguments (x>=-1).

P.S.: Running time on the HP-28S

```
(0.5 18) Zeta  --> (2.32915540368, -0.188867886498)  in 4.8610 seconds  (average of three timings using # 4554d SYSEVAL)
```

By replacing

```
FOR i s i + 3 - DUP 1 + * p * 'p' STO
```

with

```
FOR i s i + 3 - DUP SQ + p * 'p' STO
```

the running time gets down to 4.7005 seconds.

Other such minor improvements can be done, but what would probably halve the running time would be using the stack only.

Edited: 19 Dec 2012, 12:17 p.m.

[Return to Index | Top of Index]

GTO Go back to the main exhibit hall