[VA] SRC #010 - Pi Day 2022 Special Threaded Mode | Linear Mode

03-14-2022, 09:03 PM **Post: #1**

Valentin Albillo
Senior Member

Posts: 792
Joined: Feb 2015
Warning Level: 0%

[VA] SRC #010 - Pi Day 2022 Special

Today it's ***March, 14*** *aka* ***π Day***, so ***Happy π Day to all of you !***, and

Welcome to my ***SRC #010 - π Day 2022 Special***, intended to once again commemorate this most famous of constants, *π*.

After posting many threads over the years about *π*, it would seem difficult to find *new*, interesting appearances of it but actually that's not the case at all, *π* is *inexhaustible* and to prove the point let me introduce a new appearance for your enjoyment. Here you are !:

> **Note**: Unlike some previous *SRCs*, this one is **not** intended as a *challenge* to the readers to produce results on their own. Instead, this will be ***article****-like*: Here I give outright all the details and my commented results (which were obtained by my own *original research*, so they've never been available anywhere on the Internet so far), and you can read and enjoy them at once without delay. Come to that, you might try to reproduce and even *expand* my results using your own calcs if you feel like it. But you saw them here first ! 🙂

An unexpected infinite product for *π* and related questions.

Consider this infinite product ***P(x)***, for $0 \le x < \infty$:

$$P(x) = x^{3/2}\prod_{n=2}^{\infty} x\left(1-\frac{1}{n^2}\right)^{n^2}$$

Now let me answer the following ***4 Questions 4*** by first writing a little bit of *RPN* code for a programmable *HP* calc, and then using it to do some sleuthing. For speed and accuracy considerations I'll use an ancient version *2.2 (2019)* of **Free42** without using any of its extended instruction set, so that the code will run unmodified in a physical **HP-42S** (albeit at reduced speed and accuracy).

First of all, I need to write code to evaluate ***P(x)***, and as I can't use an infinite number of terms, I'll store the number of terms to use, ***N***, in variable ***"N"*** so that I'll be able to see how it does affect the accuracy of the results obtained, which will prove useful for the sleuthing afterwards.

Thus, this *46-byte* program will evaluate ***P(x)*** for a given ***x***, assuming that the number ***N*** of terms to use has been previously stored in variable ***"N"***

```
LBL "PX"
STO 02
RCL "N"
STO 00
STO 04
1
STO- 04
STO 03
LBL 00
1
RCL 00
X^2
STO 01
1/X
-
RCL 01
Y^X
RCLx 02
STOx 03
DSE 00
DSE 04
GTO 00
RCL 02
SQRT
RCLx 02
RCLx 03
END
```

Let's try computing a few values with ***PX*** using just ***N = 10*** terms, for speed. We get, for instance:

```
FIX 6, 10, STO "N", 1, XEQ "PX" -> 0.000091 ... etc., ...
```

N	x = 1	x = 2	x = 3
10	0.000091	0.131395	9.279782

The Four Questions

a. Is there a value of *x* for which *P(x)* equals π ?

To answer this question I'll first create a wrapper program and then use the **[SOLVER]** to solve the equation:

$$P(x) = \pi$$

The wrapper program is this trivial *23-byte* piece of code:

```
01  LBL "PXEQ"       05  XEQ "PX"
02  MVAR "N"         06  PI
03  MVAR "X"         07  -
04  RCL "X"          08  END
```

and solving for increasing values of ***N*** = *10, 100, 1000, 10000* , we get:

```
FIX 8, SOLVER -> Select Solve Program, [PXEQ],
10, [N][X] -> 2.70596645, 100, [N][X] -> ... etc., ...
```

N	x
10	2.70596645
100	2.71814726
1000	2.71828047
10000	2.71828181

and it's fairly obvious to any math-inclined person that ***the root is clearly converging to*** e = *2.718281828...*, the base of the *natural logarithms* and its inverse, the *exponential function*, which is thus the answer to the first ***Question***, and so we have the promised **unexpected infinite product for** π announced in the title, the *awesome* expression:

$$\pi = e^{3/2} \prod_{n=2}^{\infty} e\left(1 - \frac{1}{n^2}\right)^{n^2}$$

which beautifully relates π and e. Now, if you remember my last year's ***SRC #009***, I gave there a "trick" expression for π as a function of e, namely:

$$\pi = 4 * (\text{Arctan } \mathbf{e} - \text{Arctan } \tfrac{e-1}{e+1})$$

the *catch* being that e isn't necessary here *at all*, an *infinity* of other values will do, e.g. your age, or your phone number, or your friend's. Can't it be the same case here, that the above infinite product ***P(x)*** will evaluate to π for arguments ***x*** *other* than e ? This leads me to the second ***Question*** ...

b. We know now that *P(e)* = π. Are there any other such arguments or is *e* unique ?

In order to answer this question, I'll do some sleuthing after reformulating it as two other related questions, namely:

- What's the value of ***P(x*** < e***)*** ?
- What's the value of ***P(x*** > e***)*** ?

Well, using the **PX** program above for various increasing values of ***N*** = *10, 100, 1000, 10000*, we get the following, in **SCI 4**:

N	x = 1	x = 2	x = 2.7	x = e	x = 2.8	x = 3
10	9.0733E-5	1.3140E-1	3.0696E0	3.2950E0	4.4970E0	9.2798E0
100	7.1239E-44	1.2771E-13	1.6024E0	3.1573E0	6.1958E1	6.3592E4
1000	9.6769E-435	1.4664E-133	3.6744E-3	3.1432E0	2.3300E13	2.2159E43

10000	2.1637**E-4343**	6.1049**E-1333**	1.5436**E-29**	3.1417**E0**	1.3775**E129**	6.1138**E428**

and we can clearly see that for arguments ***x*** < *e* the value of ***P(x)*** goes to ***0***, while for arguments ***x*** > *e* it goes to **∞**, so the answer to the second question is:

> ***e*** **is indeed the *only* argument which makes this infinite product evaluate to** π.

Just for fun, if you own some ***HP*** calc which has graphics capabilities, try plotting ***P(x)*** for ***x*** = ***0*** to ***2*e*** in steps of ***e/10***, for various values of ***N*** (say, *10*, *100*, *1000*, ...). Post a *screen capture* of the plot, if possible. That said, time for third ***Question*** ...

c. Now, *fixing x* as *e*, the question is: How many correct digits of π (give or take a few *ulps*) do we get when using *N = 10, 100, 1000, ...*, terms ?

To help answer this question, and to speed the computation, now that ***x*** is not an argument anymore because it's fixed as *e*, I've written a new version of **PX**, a *59-byte* program now called **PN** because it only depends on the number of terms, ***N***, and also optimized for speed by displaying a *Wait...* message while the program runs (avoids the scrolling symbol) and using stack registers, not as easy to understand as **PX** but faster:

```
01 LBL "PN"
02 "Wait..."
03 AVIEW
04 STO 03
05 STO 02
06 2
07 1
08 STO 00
09 STO- 02
10 E^X
11 ENTER
12 SQRT
13 RCLx ST Y
14 STO 01
15 Rv
16 LBL 00
17 X<>Y
18 ENTER
19 X^2
20 ENTER
21 1/X
22 RCL- 00
23 +/-
24 X<>Y
25 Y^X
26 RCLx ST T
27 STOx 01
28 X<> ST T
29 ISG ST Y
30 LBL 00
31 DSE 02
32 GTO 00
33 CLST
34 CLD
35 RCL 01
36 END
```

Let's use it to obtain these data, in **FIX 5** :

FIX 5, *10*, **XEQ** *"PN"* **->** **3.29501** ... *etc.*, ...

N	*PN*
10	**3.2**9501
100	**3.1**5726
1000	**3.143**16
10000	**3.1417**5
100000	**3.14161**
1000000	**3.14159**

My program benefits from using **Free42** *Decimal*'s 34-digit precision, which helps cater for any cumulative rounding errors, and as can be seen ***PN*** does indeed converge *very slowly* to π as ***N*** goes to **∞**, and its value for ***N*** = *1,000,000* comes out as:

[SHOW] (and hold) **-> 3.14159 42243 85727 33446 22511 05879 403**

computed accurately to at least 27 digits or better, but giving an estimated value of π accurate to only ***7*** *correct digits*, save 2 units in the last place *(ulp)*.

Thus, we have that this infinite product computed to ***N*** terms does indeed converge to π but at an *excruciatingly slow* speed, needing a ***million*** terms to get just about ***7 digits***. And this brings us to the fourth and final ***Question***: *Can we do something about it ?*

d. Finally, the sleuthing part: Can we do something about the extremely slow convergence ?

> **Note**: I did my own *original research* so don't search the Internet for any of this 'cause it's not there, this is the first time this info appears on the Internet. It's not *rocket science* but no one published any of this before.

To try and speed the convergence, first of all I went on to estimate the error, using ***PN*** to compute the following values and then subtracting π from each to obtain the errors, which I recognized as being *very close* to π/2 = *1.57079632679...* divided by *a million* (i.e., ***N***):

N	*PN(N)*	*PN(N) -* π	*Observations*

```
    999998    3.14159 42243 88868 93182 82237 27239 246   1.57079907569 E-6   = π/2 * 10.00001750003 E-7
    999999    3.14159 42243 87298 13157 44454 09443 986   1.57079750489 E-6   = π/2 * 10.00000750001 E-7
   1000000    3.14159 42243 85727 33446 22511 05879 403   1.57079593410 E-6   = π/2 * 9.99999750000 E-7
   1000001    3.14159 42243 84156 54049 16628 07700 354   1.57079436330 E-6   = π/2 * 9.99998750002 E-7
   1000002    3.14159 42243 82585 74966 25768 41450 776   1.57079279251 E-6   = π/2 * 9.99997750005 E-7
```

thus, all the errors seem to be very close to π/2 * 1E-6, with the smallest one occurring for ***N*** = *1000000*, where the error is π/2 * 9.99999750000 E-7. Noticing this, I then used π/2 * 1E-6 as a correction term to be applied to the computed ***PN(****1000000****)***, obtaining the following:

> ***PN(****1000000****)*** - π/2 * 1E-6 = 3.14159 26535 89400 53956 56318 74557 711

and subtracting π, the absolute error now is ~ *3.92699 E-13*, which means we've got about ***14*** *correct digits* (save 4 *ulp*), where previously we had just ***7*** *correct digits* (save 2 *ulp*). In other words, applying this extremely simple correction term, π/2 * 1/***N***, essentially ***duplicates*** the number of correct digits.

Can we do better ? Yes, we can. Observing the errors using just ***PN(N)*** above for ***N*** = *999998* to ***N*** = *1000002*, we notice that not only are they of the form

> π/2 * 1E-6 ~ π/2 * 1/***N*** ,

but the actual differences with respect to *that* value also have a very regular form:

> *..750003, ..750001, ..750000, ..750002, ..750005*,

which suggests a **second** correction term to cater for the *..75* difference. To cut to the chase, after a few trivial arithmetic operations the second correction term is immediately found to be in absolute value equal to π/2 * 1/(4*N^2), and the corrected evaluation is now:

> π ~ ***PN(N)*** - π/2 * (1/***N*** - 1/(4*N^2))

and as π appears on both the *LHS* and the *RHS*, we proceed to isolate π at the *LHS*, which gives:

> π ~ ***PN(N)*** / (1 + 1/(2****N***) - 1/(8*N^2))

which, if desired, could be easily converted to the form π ~ ***PN(N)*** * (1 - 1/(2****N***) + 3/(8*N^2) + ...) by *polynomial division*, but the above expression will do for now, as we do not have enough additional terms to do an accurate polynomial division anyway.

This short additional code applies both *correction terms* to the output of ***PN(N)***. First, change **36 END** to **36 STOP** and then include after it the following lines:

```
37  RCL 03     41  X^2     45  -     49  END
38  RCL+ 03    42  8       46  1
39  1/X        43  x       47  +
40  RCL 03     44  1/X     48  /
```

This adds just *17 bytes* to ***PN*** and executes instantly but as we'll see in a moment, it ***greatly*** increases the number of correct digits. To use it, simply:

> ***N*** *(number of terms)*, **XEQ** ***"PN"*** **->** *(shows computed* ***PN(N)*** *and pauses)*, **R/S ->** *(shows corrected value)*

When particularized for ***N = 1,000,000***, the corrected evaluation gives:

```
    PN(N) = 3.14159 42243 85727 33446 22511 05879 403  ( 7 correct digits save 2 ulp )
Corrected = 3.14159 26535 89793 23864 73305...
        π = 3.14159 26535 89793 23846 26433...
    Error ~ 1.84687 E-19  ( i.e. 20 correct digits save ~ 2 ulp )
```

This means we've got essentially ***20*** *correct digits* using just two simple, inexpensive correction terms, while the original uncorrected ***PN(****1000000****)*** gave us only about ***7*** *correct digits*. Let's check the results for other values of ***N***, for instance:

- For ***N = 100,000***:

```
        PN(N) = 3.14160 83615 13791 56287 28512 11516 805  ( 6 correct digits save 2 ulp )
    Corrected = 3.14159 26535 89793 52207...
            π = 3.14159 26535 89793 23846...
        Error ~ 2.83612 E-16  ( 17 correct digits save 2 ulp )
```

- For ***N = 10,000***:

```
        PN(N) = 3.14174 97292 95765 50614 37729 49086 661  ( 5 correct digits save 2 ulp )
    Corrected = 3.14159 26535 90076 819...
            π = 3.14159 26535 89793 238...
        Error ~ 2.83581 E-13  ( 14 correct digits save 3 ulp )
```

And it seems that using the two correction terms we've obtained, we empirically have:

New *#correct digits =* ***3 * Old*** *#correct digits - 1*

Thus, while using just ***one*** correction term *duplicates* the number of correct digits, using ***two*** correction terms essentially ***triples*** the precision obtained, i.e. :

```
N = 10,000      (5 correct digits)  -> 3 * 5 - 1 = 14 correct digits
N = 100,000     (6 correct digits)  -> 3 * 6 - 1 = 17 correct digits
N = 1,000,000   (7 correct digits)  -> 3 * 7 - 1 = 20 correct digits
```

and of course the number of correct digits could be increased even further by simply obtaining additional correction terms, either empirically as I've done here or better yet, analytically.

Matter of fact, I've managed to obtain two additional terms empirically, but giving details here would make this already *humongous* exposition even much longer, so that's left as an exercise for the interested reader. 🙂

That's all. **Any and all *constructive* and *on-topic* comments are most welcome and appreciated**.

V.

All My Articles & other Materials here: **Valentin Albillo's HP Collection**

PM WWW FIND EDIT QUOTE REPORT

03-14-2022, 11:29 PM **Post: #2**

EdS2
Senior Member

Posts: 358
Joined: Apr 2014

RE: [VA] SRC #010 - Pi Day 2022 Special

.
Thank you Valentin, that is both unexpected and interesting. Indeed, thank you also for the reminder of that previous post.

My questions would be on the lines of
- why is this so? (It seems another unexpected connection between e and pi)
- how did you find it?

I'm also interested in this process of intuiting the correction terms. How sure can we be that what seem to be correct terms are in fact correct?

EMAIL PM FIND QUOTE REPORT

03-15-2022, 04:11 PM (This post was last modified: 03-18-2022 09:57 AM by Ángel Martin.) **Post: #3**

Ángel Martin
Senior Member

Posts: 1,276
Joined: Dec 2013

RE: [VA] SRC #010 - Pi Day 2022 Special

Valentín, many thanks for the very interesting contribution, you've done it again !

As you guys know I'm "stuck" in the 41 world, which means can't really duplicate Valentín's results due to its "venerable" (read: severely limited) data precision/accuracy design: a 10-digit mantissa in user code definitely ain't going to cut it, and sure enough my FOCAL routines did not work at all.

I decided to give MCODE a chance to see how much of an improvement 3 additional digits would make, and interestingly enough it works, well sort of works because again, the benchmark is always up against the same barrier. The ink is still fresh, I *think* it's all correct but there may be errors...

For anyone who may care about the details, in the **attached pdf** you can see the MCODE listing for **PPIE**, based on Valentín's product formula plus adding the two correction factors. Cutting to the chase the final results show that the ***sweet spot appears for n=495 terms***, which gives a 10-digit value of 3.141592703 , i.e. a delta of 1.55972E-08 (in percent absolute value) versus the 10-digit native pi value.

So on one hand the restrictions won't allow going much further, but at least it doesn't take an exorbitant number of terms to reach such "local optima", for the lack of a better definition (yes, I know: poor man's consolation at play...)

Here's the complete table with all logged results - note how things go south quite rapidly for N>500, which I can only attribute to the inadequate platform for this type of exercise - unless someone can spot other flaws to my reasoning?

Code:

```
n       result          |Delta|
10      3.157699001     0.005126809
100     3.141749935     5.00641E-05
200     3.14163147      1.23555E-05
300     3.141608481     5.03789E-06
400     3.141598986     2.01554E-06
450     3.141595562     9.25645E-07
475     3.141593955     4.14121E-07
480     3.141593627     3.09716E-07
490     3.141592941     9.13549E-08
494     3.141592753     3.15127E-08
```

Anyway, thanks to Valentín for the opportunity to play around with this, I'm really enjoying it. It goes without saying that **PPIE** will make its way into the forthcoming PIE ROM, soon to be released.

Best,
ÁM

PM FIND QUOTE REPORT

03-15-2022, 07:01 PM **Post: #4**

Fernando del Rey

Junior Member

Posts: 6
Joined: Dec 2013

RE: [VA] SRC #010 - Pi Day 2022 Special

Thanks, Valentín, I have found your post rather interesting and entertaining.

Perhaps many members of this forum would prefer that you give them a challenge rather than an article, but your challenges are typically way out of my reach, so I liked this article-style SRC.

I entered all your code in Free42 Decimal on my iPhone 11 and ran all the cases in your post, and a few more cases. As expected, everything worked fine and the program executions times were extremely fast, even for N=1,000,000.

Then I decided to try to run all the code in a physical HP-42S. What I found is that you can run PX and PN up to N=100 in relatively short times (seconds, not minutes). Even when using the solver with the wrapper program PXEQ, you can start with a small N value (say, N=10) to get a first estimate of X, and then gradually progress to higher N values (N=20, 50, 100), letting the resulting X value from the previous iteration be the initial guess of the next iteration. In that way, you get a relatively fast (in time) convergence, even for N=100.

All the results are still meaningful with N=100 on a physical HP-42S and the trends can be distinguished (results approaching e or Pi), even if the convergence of the infinite product function is very slow. The result of program PN with the correction terms added is surprisingly good for N=100, with an error of only 2.9e-7!

Now, I wonder if you would have been able to derive this function and the corrections terms, and to write a similar article back in 1988, using no computer and just a physical HP-42. Or do you absolutely need the speed and increased accuracy of Free42?

My guess is that you would have managed to make the same discovery in 1988 with a physical HP-42S, intuition, and a lot of patience. What do you think?

EMAIL PM FIND QUOTE REPORT

03-15-2022, 11:04 PM **Post: #5**

Albert Chan

Posts: 1,696

Senior Member Joined: Jul 2018

RE: [VA] SRC #010 - Pi Day 2022 Special

> **Valentin Albillo Wrote:** (03-14-2022 09:03 PM)
>
> the *awesome* expression:
>
> $$\pi = e^{3/2} \prod_{n=2}^{\infty} e \left(1 - \frac{1}{n^2}\right)^{n^2}$$
>
> which beautifully relates π and e.

Below confirmed expression numerically, by turning sum to integral.

ln(pi)
= 3/2 + sum(1 + n^2 * ln(1 - 1/n^2), n = 2 .. inf)
= 3/2 + sum(1 - n^2 * ((1/n)^2 + (1/n)^4/2 + (1/n)^6/3 + (1/n)^8/4 + ...), n=2 .. inf)
= (1-ζ(0)) + (1-ζ(2))/2 + (1-ζ(4))/3 + (1-ζ(6))/4 + ... // note: ζ(0) = -1/2

Zeta even integer generating function:

$$\sum_{n=0}^{\infty} \zeta(2n) x^{2n} = -\frac{\pi x}{2} \cot(\pi x) = -\frac{1}{2} + \frac{\pi^2}{6} x^2 + \frac{\pi^4}{90} x^4 + \frac{\pi^6}{945} x^6 + \cdots$$

Replacing x by √x, and integrate from 0 to 1, we matched zeta terms.
We also need to add a function, to match fraction terms.

1/(1-x) = 1 + x + x^2 + ...
∫(1/(1-x), x=0..1) = 1 + 1/2 + 1/3 + ...

$$\ln(\pi) = \int_0^1 \left(\frac{1}{1-x} + \frac{\pi\sqrt{x}}{2\ \tan(\pi\sqrt{x})} \right) dx$$

Note that integrand is inaccurate when x approach 1. P cannot be set too small.

```
10 P=.000001
20 DEF FNF(X,Y)=1/(1-X)+.5*Y/TAN(Y)
30 DISP INTEGRAL(0,1,P,FNF(IVAR,PI*SQRT(IVAR))), EXP(RES)
>
>RUN
1.14472988295      3.14159264448
```

EMAIL PM FIND QUOTE REPORT

03-16-2022, 09:50 AM **Post: #6**

J-F Garnier
Senior Member

Posts: 588
Joined: Dec 2013

RE: [VA] SRC #010 - Pi Day 2022 Special

Thanks Valentin for this interesting reading. Relations between pi and e always intrigued me.

> **Ángel Martin Wrote:** (03-15-2022 04:11 PM)
>
> As you guys know I'm "stuck" in the 41 world [...] and sure enough my FOCAL routines did not work at all.

Really? HP-41 user code can't do it?

Let's see:
Maybe it's better to transform Valentin's expression with log and then compute the exponential at the end.
Using pseudo algebraic language (I'm not comfortable with graphic equation editors), with ln as the natural log:

```
PN = exp(3/2) * Prod(n=2,N,e*(1-1/n²)^n²) / (1+1/(2*N)-1/(8*N²))
```
becomes
```
ln(PN) = 3/2 + Sum(n=2,N,1+n²*ln(1-1/n²)) - ln(1+1/(2*N)-1/(8*N²))
```

e doesn't appear explicitly any more, but of course it does at the end when computing exp(ln(PN)).

Here is the corresponding HP-41/42 program, using the here highly useful LN1+X function that preserves the accuracy for small $1/n^2$ quantities to some extend:

```
01*LBL "PN2"
02 "RUNNING"
03 AVIEW
04 STO 00 ; N
05 1
06 -
07 STO 01 ; control loop 1..N-1
08 0
09*LBL 00 ; sum loop <---
10 RCL 01
11 1
12 + ; n=2..N
13 X^2
14 ENTER^
15 1/X
16 CHS
17 LN1+X
18 *
19 1
20 +
21 +
22 DSE 01
23 GTO 00 ; sum endloop --^
24 RCL 00
25 2
26 *
27 1/X
28 RCL 00
29 X^2
30 8
31 *
32 1/X
33 -
34 LN1+X ; correction factor
35 -
36*LBL 01 ; final result
37 1.5
38 +
39 E^X
40 CLD
41 END
```

and results for the HP-41:

```
10.00000000 RUN
RUNNING
3.141844397 ***

100.0000000 RUN
RUNNING
3.141592946 ***

200.0000000 RUN
RUNNING
3.141592701 ***

300.0000000 RUN
RUNNING
3.141592670 ***

400.0000000 RUN
RUNNING
3.141592651 *** best result

500.0000000 RUN
RUNNING
3.141592685 ***
```

Not bad for this ancient 10-digit machine, isn't it?

So thanks again Valentin for this contribution to the fascinating pi, and Ángel for giving me the opportunity to write a HP-41 code, something I'm rarely doing - I'm a HP-71B man - but the HP-41 language is so deeply buried in my mind since 40 years that it was very natural.

J-F

EMAIL PM WWW FIND QUOTE REPORT

03-16-2022, 11:30 AM (This post was last modified: 03-16-2022 11:37 AM by Ángel Martin.) **Post: #7**

Ángel Martin
Senior Member

Posts: 1,276
Joined: Dec 2013

RE: [VA] SRC #010 - Pi Day 2022 Special

> **J-F Garnier Wrote:** (03-16-2022 09:50 AM)
> Thanks Valentin for this interesting reading. Relations between pi and e always intrigued me.
>
> > **Ángel Martin Wrote:** (03-15-2022 04:11 PM)
> > As you guys know I'm "stuck" in the 41 world [...] and sure enough my FOCAL routines did not work at all.
>
> Really? HP-41 user code can't do it?

Well, that's not quite what I said - I stated that "**my**" routines didn't work, as I was slavishly porting Valentín's HP-42 code directly - a big *booooo* to me ;-)

So many thanks for your new routine, very clever and good example of the platform capabilities when you know what you're doing with it.
I'm however curious: how does it respond for higher number of terms, say 10,000 or even 100,000? That's where the rubber meets the road, methinks.

Honestly I've come to the point that it's easier for me to go straight into MCODE than sleuthing around the FOCAL dustbin in search for better games, I confess.

Best,
ÁM

PM FIND QUOTE REPORT

03-16-2022, 12:49 PM (This post was last modified: 03-16-2022 12:53 PM by J-F Garnier.) **Post: #8**

J-F Garnier
Senior Member

Posts: 588
Joined: Dec 2013

RE: [VA] SRC #010 - Pi Day 2022 Special

> **Ángel Martin Wrote:** (03-16-2022 11:30 AM)
> I'm however curious: how does it respond for higher number of terms, say 10,000 or even 100,000? That's where the rubber meets the road, methinks.

With the HP41, the best results are obtained with N around 400. The accuracy starts to decline after 500, as you noted too.
I guess the reason is the accuracy of the $\ln(1-1/n^2)$ quantity even with the LN1+X function.
$\ln(1+x)$ is about $x-x^2/2+\ldots$ and with 10 digits the term $(1/n^2)^2$ starts to get inaccurate (relative to $1/n^2$) when n is in range of 1000 or so.
When switching to the Free42 platform, I got similar (and not identical) results to Valentin's program, however without improving the accuracy of the corrected value.

For instance:
N=1E5, w/o correction:
VA : 3.14160 83615 13791 56287 28512 11516 805
JFG: 3.14160 83615 13791 56287 28668 95754 789

N=1E5, w/ correction:
VA : 3.14159 26535 89793 52207..
JFG: 3.14159 26535 89793 52207..

I guess that at N=1E5, we are still far from the limits of Free42 accuracy.

J-F

EMAIL PM WWW FIND QUOTE REPORT

03-16-2022, 09:49 PM (This post was last modified: 03-17-2022 01:58 PM by Albert Chan.) **Post: #9**

Albert Chan
Senior Member

Posts: 1,696
Joined: Jul 2018

RE: [VA] SRC #010 - Pi Day 2022 Special

> **Albert Chan Wrote:** (03-15-2022 11:04 PM)
>
> $$\ln(\pi) = \int_0^1 \left(\frac{1}{1-x} + \frac{\pi\sqrt{x}}{2\ \tan(\pi\sqrt{x})} \right) dx$$
>
> Note that integrand is inaccurate when x approach 1. P cannot be set too small.

To improve accuracy, lets remove square roots, x = y^2, dx = 2y dy

g(y) = (1/(1-y^2) + pi*y/2/tan(pi*y)) * 2y
       = -1/(y+1) - 1/(y-1) + pi*y^2/tan(pi*y)

Since tan(pi*(1-y)) = -tan(pi*y), we might as well fold the area.

∫(g(y), y=0..1) = ∫(g(y) + g(1-y), y=0..1/2)

$$\ln(\pi) = \int_0^{1/2} \left(\frac{-1}{y+1} + \frac{1}{y-2} + \frac{-1}{y-1} + \frac{1}{y} + \frac{\pi(2y-1)}{\tan(\pi y)} \right) dy$$

H(y) = ∫(-1/(y+1) + 1/(y-2) - 1/(y-1) dy = -ln|y+1| + ln|y-2| - ln|y-1|

H(1/2) = -ln(3/2) + ln(3/2) - ln(1/2) = ln(2)
H(0) = -ln(1) + ln(2) - ln(1) = ln(2)

With H(1/2) - H(0) = 0, we can remove integrand first 3 terms. 🙂

$$\ln(\pi) = \int_0^{1/2} \left(\frac{1}{y} + \frac{\pi(2y-1)}{\tan(\pi y)} \right) dy$$

Let's compare the 2 versions.

```
10 P=1E-6
20 DEF FNF(X,Y)=1/(1-X)+.5*Y/TAN(Y)
30 DISP INTEGRAL(0,1,P,FNF(IVAR,PI*SQRT(IVAR))),EXP(RES)
40 DEF FNG(Y)=1/Y+PI*(2*Y-1)/TAN(PI*Y)
50 DISP INTEGRAL(0,.5,P,FNG(IVAR)),EXP(RES)
>
>RUN
1.14472988295      3.14159264448
1.14472988584      3.14159265356
>
>LOG(PI), PI
1.14472988585      3.14159265359
```

EMAIL PM FIND QUOTE REPORT

03-17-2022, 03:54 PM (This post was last modified: 03-17-2022 04:24 PM by Albert Chan.) **Post: #10**

Albert Chan
Senior Member

Posts: 1,696
Joined: Jul 2018

RE: [VA] SRC #010 - Pi Day 2022 Special

> **Albert Chan Wrote:** (03-16-2022 09:49 PM)
>
> $$\ln(\pi) = \int_0^{1/2} \left(\frac{1}{y} + \frac{\pi(2y-1)}{\tan(\pi y)} \right) dy$$

Wolfram Alpha proved this, with closed-form anti-derivative !

Code:

```
def G(y):
    k = pi*j
    z = exp(2*k*y)
    return ln(y) - ln(sin(pi*y)) + y*(2*log1p(-z)-k*y) + polylog(2,z)/k
```

>>> from mpmath import *
>>> limit(G,1/2) - limit(G,0)
mpc(real='1.1447298858494002', imag='0.0')
>>> exp(_)
mpc(real='3.1415926535897931', imag='0.0')

OP product form, which integral was derived from, is thus proved.

$$\pi = e^{3/2} \prod_{n=2}^{\infty} e \left(1 - \frac{1}{n^2}\right)^{n^2}$$

EMAIL PM FIND QUOTE REPORT

03-17-2022, 07:22 PM (This post was last modified: 03-17-2022 10:24 PM by Albert Chan.) **Post: #11**

Albert Chan
Senior Member

Posts: 1,696
Joined: Jul 2018

RE: [VA] SRC #010 - Pi Day 2022 Special

> **EdS2 Wrote:** (03-14-2022 11:29 PM)
> I'm also interested in this process of intuiting the correction terms.
> How sure can we be that what seem to be correct terms are in fact correct?

We can get correction term symbolically, to be "sure"

Correction term = product(e*(1-1/k^2)^(k^2), k=n+1 .. inf)
Instead of doing products, we sum the log's instead.

We estimate the size of correction using Euler-Maclaurin formula

XCAS> f := 1 + ln(1-1/x^2)*x^2
XCAS> c := int(f) - f/2 + f'/12 - f'''/720 :;

f = -x^-2/2 - x^-4/3 - x^-6/4 + ... ⇒ c(x = inf) = 0. No need to eval upper limit.

XCAS> C := exp(c)(x=n+1) :;   // PN/C ≈ pi
XCAS> series(C, n=inf, 7)

$$1 + \frac{1/2}{n} - \frac{1/8}{n^2} + \frac{13/144}{n^3} - \frac{77/1152}{n^4} + \frac{547/11520}{n^5} - \frac{13529/414720}{n^6} + O\left(\frac{1}{n^7}\right)$$

Continued fraction with taylor series (n=inf) that matches C coefs: (N = 2n+1)

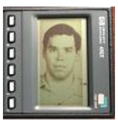
$$C = 1 + \cfrac{1}{(N - \frac{1}{2}) - \cfrac{1}{\frac{36}{17} N + \cfrac{1}{\frac{1445}{419} N + \ldots}}}$$

Or, based from continued fraction approximation of little c: (again, N = 2n+1)

$$\ln(C) = \cfrac{1}{N - \cfrac{1}{\frac{9}{5} N + \cfrac{1}{\frac{125}{8} N + \ldots}}}$$

EMAIL PM FIND QUOTE REPORT

03-17-2022, 09:10 PM (This post was last modified: 03-17-2022 09:11 PM by Gerson W. Barbosa.) **Post: #12**



Gerson W. Barbosa
Senior Member

Posts: 1,447
Joined: Dec 2013

RE: [VA] SRC #010 - Pi Day 2022 Special

Albert Chan Wrote: (03-17-2022 07:22 PM)

Continued fraction with taylor series (n=inf) that matches above coefs:

$$1+\cfrac{1}{(2n+\frac{1}{2})-\cfrac{1}{\frac{36}{17}*(2n+1)+\cfrac{1}{\frac{1445}{419}*(2n+1)}}}$$

The following corrects the result to 3.1415926535 for n=500:

1+1/(2n+1/(2+1/(n+17/(18n+1/2))))

Too few terms to deduce a pattern if any, though.

EMAIL PM FIND QUOTE REPORT

03-18-2022, 01:25 AM **Post: #13**

Valentin Albillo
Senior Member

Posts: 792
Joined: Feb 2015
Warning Level: 0%

RE: [VA] SRC #010 - Pi Day 2022 Special

.
Hi, **all**,

Thanks to **EdS2**, **Ángel Martín**, **Fernando del Rey**, **Albert Chan**, **J-F Garnier** and **Gerson W. Barbosa** for your interest in my ***SRC #10***. Here I'll address some of your questions, plus assorted additional comments:

EdS2 Wrote:

I'm also interested in this process of intuiting the correction terms. How sure can we be that what seem to be correct terms are in fact correct ?

Using an empirical approach to find them, as I did here, you can never be completely sure that what you found is 100% correct because that would require a theoretical approach. It's the same with ***Pi*** itself: no matter how many digits you compute, you can never be 100% sure that ***Pi*** is a *normal* number, *that* requires theory to stablish.

However, after analyzing *the first 16 trillion bits of* ***Pi***, the result is that the decision *"**Pi** is not normal"* has credibility *4.3497.10*$^{\boldsymbol{-3064}}$, which makes it all but *impossible*, so ***Pi*** is all but *certainly* normal. Same here, after computing these correction factors using high enough precisión they're highly certain to be correct.

Ángel Martín Wrote:

I decided to give *MCODE* a chance to see how much of an improvement 3 additional digits would make, and interestingly enough it works, well sort of works because again, the benchmark is always up against the same barrier.

It's quite brave of you to attempt the feat with the limited precision allowed by the **HP-41** platform *(10 digits to the user, 13 digits internally)*, but I see you pretty much succeeded within the unavoidable constraints of precision and speed.

Also, experimenting first with *RPN* versions (I refuse to call it the *"F"-word*, i.e. *FOCAL*) and then implementing it as an *MCODE* routine in such a short time span (*a few hours* from my *OP*), plus additionally creating high-quality documentation for the *MCODE* source code, is utterly ***unbelievable***, **you're *incredible* !** ... Wish you had *"sticked" (archaic, I know* *)* with the *HP-71B* platform instead ...

As I said at the end of my OP, I self-quote:

> *"Matter of fact, **I've managed to obtain two additional terms empirically**, but giving details here would make this already humongous exposition even much longer, so that's left as an exercise for the interested reader."*

and for your benefit and anyone interested's, the additional two correction factors I found were:

$c_3 = 13/(144 * N^3)$ and $c_4 = -77/(1152 * N^4)$

and the resulting ***Pi*** approximation becomes:

$Pi \sim PN(N) / (1 + 1/(2 * N) - 1/(8 * N^2) + 13/(144 * N^3) - 77/(1152 * N^4))$

which, when updating my ***PN*** program to include them and running it on ***Free42*** *Decimal*, gets me the following assorted results:

```
              with 3 factors      with 4 factors
```

```
                ________________________________________________________________
N terms   :            100                 100
PN(N)     :  3.15726162848       3.15726162848
Corrected :  3.14159265152       3.14159265360
Error     : -2.07468621147E-9    1.47418338373E-11
Digits    :             ~ 10                ~ 12
                ________________________________________________________________
N terms   :          1,000               1,000
PN(N)     :  3.14316305750         3.143163058
Corrected :  3.14159265359         3.141592654
Error     : -2.09731017621E-13     1.489942199E-16
Digits    :             ~ 14                ~ 17
                ________________________________________________________________
N terms   :         10,000              10,000
PN(N)     :  3.14174972930        3.14174972930
Corrected :  3.14159265359        3.14159265359
Error     : -2.09959513255E-17    1.49139164910E-21
Digits    :             ~ 18                 ~ 22      (vs ~ 14 digits w/ 2 c.f.)
                ________________________________________________________________
N terms   :              -               20,000
PN(N)     :              -        3.14167119242
Corrected :              -        3.14159265359
Error     :              -        4.46023309170E-23
Digits    :              -                 ~ 24
                ________________________________________________________________
N terms   :              -               30,000
PN(N)     :              -        3.14164501303
Corrected :              -        3.14159265359
Error     :              -        1.70263469640E-23
Digits    :              -                 ~ 24
                ________________________________________________________________
N terms   :        100,000             100,000
PN(N)     :  3.14160836151        3.14160836151
Corrected :  3.14159265359        3.14159265359
Error     : -2.11550799992E-21   -1.56692427780E-23
Digits    :             ~ 22                 ~ 24      ( vs ~ 17 digits w/ 2 c.f.)
                ________________________________________________________________
N terms   :        200,000              -
PN(N)     :  3.14160050756              -
Corrected :  3.14159265359              -
Error     : -2.58729643320E-23          -
Digits    :             ~ 24            -
                ________________________________________________________________
N terms   :      1,000,000              -
PN(N)     :  3.14159422439              -
Corrected :  3.14159265359              -
Error     : -9.89287385519E-20          -
Digits    :             ~ 20            -
                ________________________________________________________________
```

where we see the improvement afforded by using ***3*** and ***4*** correction factors, and we also see that the limits of the 34-digit accuracy provided by ***Free42*** *Decimal* begin to take its toll. For instance, with ***N*** = *1,000,000*, which gave us ***20*** decimal digits using just ***2*** factors, we would expect here to get improved accuracy when using all ***4*** c.f., yet we still get only *20* decimal digits and matter of fact the result obtained for ***N***=*200,000*, i.e. five times *less* terms, is *much* better, obtaining ~ *24 digits* instead of *20*.

This is due to several limitations: ***(a)*** when using *1,000,000* terms in the product, we might lose some *6 or 7* digits just to rounding or truncation, so we aren't getting *34* correct digits for the product, more like *27 or 28* at best. ***(2)*** using *1,000,000* terms means that we're computing expressions like $(1-10^{-12})$^(10^{12}), $(1-10^{-18})$^(10^{18}) and $(1-10^{-24})$^(10^{24}), for *2*, *3* and *4* factors, respectively, and those are likely to exceed the accuracy achievable by using only 34 digits in their computation.

Note also the improvement afforded by using ***4*** c.f. instead of ***3***: the result using ***N*** = *20,000* terms with ***4*** c.f. has about the same precision *(~ **24** correct digits)* as using ***N*** = *200,000* terms with ***3*** c.f., a *10x* speed improvement.

In short, **Ángel**, try to use these two additional correction factors but, if you don't get the desired expected, significant improvements, it might be the case that you're running against the limits of the ***HP-41*** accuracy, as happened with ***Free42*** *Decimal* above, and then it's just a case of finding the *sweet spot* and see if the additional terms do any good to achieve the sweetest one possible.

Fernando del Rey Wrote:

> Then I decided to try to run all the code in a physical HP-42S. What I found is that you can run PX and PN up to N=100 in relatively short times (seconds, not minutes). Even when using the solver with the wrapper program PXEQ, you can start with a small N value (say, N=10) to get a first estimate of X, and then gradually progress to higher N values (N=20, 50, 100), ***letting the resulting X value from the previous iteration be the initial guess of the next iteration. In that way, you get a relatively fast (in time) convergence, even for N=100***.

Thanks for your interest, ***Fernando***, and I like *a lot* that, apart from trying my code on ***Free42***, you also ran it on a *physical* ***HP-42S***, despite its precision and speed limitations. Your idea of using the result of the previous iteration as the initial guess of the next is rather clever, even if doing it in ***Free42***, because the ***Solver*** gets rather slow for high ***N***, as it's an iterative process which evaluates the product a sizable number of times. *Well done !* :)

> **Fernando del Rey Wrote:**
>
> Now, I wonder if you would have been able to derive this function and the corrections terms, and to write a similar article back in 1988, using no computer and just a physical HP-42. [...] My guess is that you would have managed to make the same discovery in 1988 with a physical HP-42S, intuition, and a lot of patience. What do you think?

I (unmodestly) think that I would have managed back then. I wrote a number of multiprecision programs for both my ***HP-67*** first and ***HP-41C*** afterwards, and it's pretty likely that I would have tried to compute the product and the first two correction terms when using ***N*** *= 1,000* or more on the new, faster, much more capable (matrix operations, much larger *RAM*) ***HP-42S***, it's just a matter of leaving the program running for as long as the batteries last. So, yes, I think it was doable in 1988.

> **Albert Chan Wrote:**
>
> Below confirmed expression numerically, **by turning sum to integral**.

Very clever, to think of that, **Albert Chan** !

And without using ***XCAS*** no less, as you know that I don't like people using such tools in my challenges or articles because I want people to use their *vintage* ***HP*** *calculators*, as this is the ***Museum of HP calculators***, not ***MathOverflow*** or ***Stack Overflow***, so thanks for respecting my wishes, though I'm sure you were *itching* to use ***XCAS*** or *Wolfram Alpha* or something like that. *Congratulations !*

> **Jean-François Garnier Wrote:**
>
> Thanks Valentin for this interesting reading. **Relations between pi and e always intrigued me**. [...] Maybe it's better to transform Valentin's expression **with log** and then compute the exponential at the end.

You're welcome and yes, I know that you're fond of ***Pi-e*** relationships, me too ! And your idea of taking *logs* and then taking advantage of the built-in `LN1+X` function in order to enhance accuracy is *brilliant*, congratulations ! ... Perhaps Ángel might use it to achieve better results with its ***HP-41C*** *MCODE* version.

V.

All My Articles & other Materials here: **Valentin Albillo's HP Collection**

PM WWW FIND EDIT QUOTE REPORT

03-18-2022, 09:48 AM (This post was last modified: 03-19-2022 09:31 AM by Ángel Martin.) **Post: #14**

Ángel Martin
Senior Member

Posts: 1,276
Joined: Dec 2013

RE: [VA] SRC #010 - Pi Day 2022 Special

> **Valentin Albillo Wrote:** (03-18-2022 01:25 AM)
>
> In short, **Ángel**, try to use these two additional correction factors but, if you don't get the desired expected, significant improvements, it might be the case that you're running against the limits of the ***HP-41*** accuracy, as happened with ***Free42*** *Decimal* above, and then it's just a case of finding the *sweet spot* and see if the additional terms do any good to achieve the sweetest one possible.

Indeed the two additional correction factors make a huge difference: once added to the code the sweet spot for 10-digit pi now occurs with N=35 terms, giving the "exact" same value returned by the native "PI" function, i.e. 3.141592654.

Here's the new table for your reference: (also includes the execution time using a *default settings* on V41, definitely not TURBO mode)

Code:

```
n       result          |Delta%|        Time
5       3.141630979    1.21992E-05     0.000174
10      3.141593984    4.23352E-07     0.000297
15      3.141592834    5.72958E-08     0.000438
20      3.141592696    1.3369E-08     0.000568
25      3.141592666    3.81972E-09     0.000698
30      3.141592658    1.27324E-09     0.000829
35      3.141592654    0                0.00096
40      3.141592652    6.3662E-10       0.001088
45      3.141592651    9.5493E-10       0.001219
50      3.14159265      1.27324E-09      0.001337
```

What a difference!

BTW the previous version had a glitch that shifted the number of terms by one, now duly corrected. This is now old history but it skewed the results in about 50-60 terms due to cumulative errors, but that's immaterial now with the new version posted here.

> **Valentin Albillo Wrote:** (03-18-2022 01:25 AM)
>
> ... And J-F Garnier's idea of taking logs and then taking advantage of the built-in LN1+X function in order to enhance accuracy is brilliant, congratulations ! ... Perhaps Ángel might use it to achieve better results with its HP-41C MCODE version.

Yes, I've changed the approach to using a summation instead of a product - even if in the MCODE realm there's not much of a difference at the end of the day: you may gain some accuracy in the sums (instead of multiplications) but you lose some in the final Log/Exp conversions. BTW, on the LN1+X, well such function exists for 10-digit accuracy but does not have a 13-digit counterpart - simply because it's not needed in MCODE, where we have access to the "real" things with calls to [LN13] and [ADDONE] of course.

Lovely end-game even on the 41, thanks again!

ÁM

"It's not the size of the wand but the skill of the wizard what counts"

Attached File(s)

PPIE MCODE.pdf (Size: 872.04 KB / Downloads: 8)

PM FIND QUOTE REPORT

03-18-2022, 05:22 PM **Post: #15**

Albert Chan
Senior Member

Posts: 1,696
Joined: Jul 2018

RE: [VA] SRC #010 - Pi Day 2022 Special

> **Ángel Martin Wrote:** (03-18-2022 09:48 AM)
>
> Yes, I've changed the approach to using a summation instead of a product - even if in the MCODE realm there's not much of a difference at the end of the day: you may gain some accuracy in the sums (instead of multiplications) but you lose some in the final Log/Exp conversions.

We don't notice the difference because correction is not strong enough.
Summing smallest term first, we would keep almost all good digits.

> **J-F Garnier Wrote:** (03-16-2022 12:49 PM)
>
> N=1E5, w/o correction:
> VA : 3.14160 83615 13791 56287 28512 11516 805
> JFG: 3.14160 83615 13791 56287 28668 95754 789

Lets recover true PN, and compare errors of products vs exp(sum of logs)

PN = C*PI = exp(ln(C))*PI = expm1(ln(C))*PI + PI

Albert Chan Wrote: (03-17-2022 07:22 PM)

Or, based from continued fraction approximation of little c: (again, N = 2n+1)

$$\ln(C) = \cfrac{1}{N - \cfrac{1}{\frac{9}{5}N + \cfrac{1}{\frac{125}{8}N + \ldots}}}$$

Note that ln(C) is odd function. Rewrite ln(C) as polynomial of 1/N, we have:

$$\ln(C) = \frac{1}{N} + \frac{5/9}{N^3} + \frac{13/45}{N^5} + \frac{127/315}{N^7} - \frac{89/135}{N^9} + \ldots$$

n = 1E5 // note: my n is JFG N
N = 2n+1 ⇒ N^7 ≈ 128E35 > 1E37

Free42: ln(C), summing to N^5 only (slight errors doesn't matter)

ln(C) → 4.9999750001944427777922220972229e-6
E↑X-1 → 4.9999875000902771093797482312670083e-6
PI * PI + → 3.141608361513791562872866895754895 // true PN

VA (products for PN) errors = 15,684,238,090 ULP // O(n^2) error ?
JFG (log sum for PN) errors = 106 ULP

EMAIL PM FIND QUOTE REPORT

Today, 01:14 AM **Post: #16**

Valentin Albillo
Senior Member

Posts: 792
Joined: Feb 2015
Warning Level: 0%

RE: [VA] SRC #010 - Pi Day 2022 Special

.
Hi, all,

Albert Chan Wrote: (03-18-2022 05:22 PM)

Lets recover true PN, and compare errors of products vs exp(sum of logs) [...] Note that ln(C) is odd function. Rewrite ln(C) as polynomial of 1/N, we have:

$$\ln(C) = \frac{1}{N} + \frac{5/9}{N^3} + \frac{13/45}{N^5} + \frac{127/315}{N^7} - \frac{89/135}{N^9} + \ldots [\ldots]$$

I must point out that this formal series of correction factors is *asymptotic* and *divergent*, i.e., its coefficients might be small and even decreasing for a while but eventually they grow bigger and bigger, both numerators and denominators, and thus can't be used to obtain arbitrary precision, as I explained in another case in post #27 of my ***Short & Sweet Math Challenge #24***. Quoting myself from that post:

Quote:

The coefficients of the formal series for *cin(x)* and *tin(x)* can be obtained in a number of ways [...] but it's important to be aware that both formal series *do not converge*. In fact, their *radius of convergence* is ***0*** and thus they behave like asymptotic series, so you can't get arbitrarily accurate results by taking more and more terms, you must instead truncate the series after a certain number of terms to get the most accurate results. Using further terms only *worsens* the accuracy.

Although at first sight the coefficients of the formal series for *cin(x)* and *tin(x)* seem to (slowly) get *smaller* and smaller, matter of fact they tend to grow ever *bigger* after a while, tending to *infinity*. For instance, for *tin(x)* we find that the smallest coefficient in absolute value is:

Coeff_{37} = -0.000000000594338574503

but afterwards we have, e.g.:

Coeff_{101} = 0.0833756228055
Coeff_{151} = 388536047335.239
Coeff_{201} = 6555423874651256623811186991.51
Coeff_{251} = -35365220492708296140377087748804440170254492009.57

The same happens in the present case: you can use a certain number of coefficients to improve accuracy up to the *"sweet point"* of maximum accuracy, but after that the accuracy quickly degrades and thus using more coefficients is useless and to be avoided.

> **Quote:**
>
> PI * PI + → **3.141608361513791562872866895754895**   **// true PN**
>
> **VA** (products for PN) **errors = 15,684,238,090** ULP   // O(n^2) error ?
> **JFG** (log sum for PN) errors = **106** ULP

Regrettably, presently I have no software available to compute the product for *n = 2* to *n = 100,000* with high accuracy (say, to 100 digits) so I can't check for sure, but I find it somewhat *hard to believe* that my computation using the ***34*** digits afforded by ***Free42*** *Decimal* would *lose **11** digits* in the process, I'd rather expect *6-7* digits lost at most.

Likewise, *Jean-François Garnier* computation of said product using logarithms performs about *100,000* multiplications, divisions (**1/x**) and logarithms (**LN1+X**) but only *loses **3** digits* ? *Really* ?

To settle down the question, if someone with access to *Mathematica* or some other arbitrary-precision software can compute the product for ***N****=100,000* using *100* digits, say, or as many as necessary to ensure full *34* correct digits or more, and post here the resulting value I'd appreciate it. Thanks in advance.

V.

All My Articles & other Materials here: **Valentin Albillo's HP Collection**

PM WWW FIND EDIT QUOTE REPORT

Today, 02:01 AM **Post: #17**

Albert Chan
Senior Member

Posts: 1,696
Joined: Jul 2018

RE: [VA] SRC #010 - Pi Day 2022 Special

> **Valentin Albillo Wrote:** (Today 01:14 AM)
>
> > **Albert Chan Wrote:** (03-18-2022 05:22 PM)
> >
> > Lets recover true PN, and compare errors of products vs exp(sum of logs) [...] Note that ln(C) is odd function. Rewrite ln(C) as polynomial of 1/N, we have:
> >
> > $$\ln(C) = \frac{1}{N} + \frac{5/9}{N^3} + \frac{13/45}{N^5} + \frac{127/315}{N^7} - \frac{89/135}{N^9} + \ldots [...]$$
>
> I must point out that this formal series of correction factors is *asymptotic* and *divergent*, i.e., its coefficients might be small and even decreasing for a while but eventually they grow bigger and bigger, both numerators and denominators, and thus can't be used to obtain arbitrary precision, as I explained in another case in post #27 of my ***Short & Sweet Math Challenge #24***.
> ...
>
> To settle down the question, if someone with access to *Mathematica* or some other arbitrary-precision software can compute the product for ***N****=100,000* using *100* digits, say, or as many as necessary to ensure full *34* correct digits or more, and post here the resulting value I'd appreciate it. Thanks in advance.

```
>>> from mpmath import *
>>> mp.dps = 100
>>> pn = lambda n: exp(nsum(lambda x: 1+log1p(-1/(x*x))*x*x,[2,n]) + 1.5)
>>> n = mpf(100000)
>>> N = 2*n+1
>>> x = pn(n)
>>> print x
3.141608361513791562872866895754895278060325823725833279147116393910631517290786764227775828378244404
```

It does matched my 34-digits "true" PN.

ln(C) correction (terms upto 1/N^9) seems safe to use.

```
>>> err = lambda c: float(pi - x * exp(-c))
>>> err(13/(45*N**5) + 5/(9*N**3) + 1/N)
```

-9.8950471946808673e-38
>>> err(127/(315*N**7) + 13/(45*N**5) + 5/(9*N**3) + 1/N)
4.0449821226917704e-48
>>> err(-89/(135*N**9) + 127/(315*N**7) + 13/(45*N**5) + 5/(9*N**3) + 1/N)
-1.229817502771026e-57

EMAIL PM FIND QUOTE REPORT

Today, 11:47 AM **Post: #18**

J-F Garnier
Senior Member

Posts: 588
Joined: Dec 2013

RE: [VA] SRC #010 - Pi Day 2022 Special

> **Valentin Albillo Wrote:** (Today 01:14 AM)
>
> > **Quote:**
> > PI * PI + → **3.14160836151379156287286689575489 5** // **true PN**
> >
> > **VA** (products for PN) **errors = 15,684,238,090** ULP // O(n^2) error ?
> > **JFG** (log sum for PN) errors = **106** ULP
>
> Regrettably, presently I have no software available to compute the product for *n = 2* to *n = 100,000* with high accuracy (say, to 100 digits) so I can't check for sure, but I find it somewhat *hard to believe* that my computation using the ***34*** digits afforded by ***Free42*** *Decimal* would *lose* ***11*** *digits* in the process, I'd rather expect *6-7* digits lost at most.
>
> Likewise, *Jean-François Garnier* computation of said product using logarithms performs about *100,000* multiplications, divisions (**1/x**) and logarithms (**LN1+X**) but only *loses* ***3*** *digits* ? *Really* ?

Honestly, I'm surprised by this result too, confirmed then by Albert.
I looked further and it *may* be an accuracy flaw in Free42. I will open an other thread to discuss it.

J-F

EMAIL PM WWW FIND QUOTE REPORT

Today, 01:29 PM **Post: #19**

Albert Chan
Senior Member

Posts: 1,696
Joined: Jul 2018

RE: [VA] SRC #010 - Pi Day 2022 Special

It is not hard to see why for log sums, we have error O(√(n)

1 + log1p(-1/k^2) * k^2
= 1 - (k^-2 + k^-4/3 + k^-6/4 + ...) * k^2
= 1 - (1 + k^-2/3 + k^-4/4 + ...)
= -(k^-2/3 + k^-4/4 + ...)

Because of 1 in front, term errors are in orders of machine epsilon.
Worst case, we have errors of O(n).

But, because errors spread-out somewhat randomly, we have O(√n)

You might try sum terms from index of 2 to n, instead of in reverse.
I would guess you would produce similar sized error for PN

--

For products of factors, **(1-1/k^2)^(k^2)**:

We expected base have errors, also in order of machine epsilon.
However, errors are not random, but clustered when k is huge.
Example, for 10-digits calculator, this is the rounded base.

b(k) = 1-1/k^2

b(99999) = 0.99999 99998 99998, rounded up
b(99998) = 0.99999 99998 99996, rounded up
...
b(82000) = 0.99999 99998 51279, *still* rounded up

(1+ε)^(n^2) = 1 + n^2 ε

Product of n-1 terms, we expected worst case errors of O(n^3)
Of course, errors are not totally skewed, we expected O(n^2+)

From previous post:
PN(n=1e5) errors = 15,684,238,090 ULP ≈ 1e5 ^ 2.04

> **Valentin Albillo Wrote:** (03-14-2022 09:03 PM)
>
> ***PN(N)*** = 3.14159 42243 85727 33446 22511 05879 403 *(**7** correct digits save 2 ulp)*

Using ln(C) correction, "true" PN = 3.14159 42243 85727 33456 11796 83910 689

PN(n=1e6) errors = 98,928,578,031,286 ULP ≈ 1e6 ^ 2.33

EMAIL PM FIND QUOTE REPORT

Today, 07:59 PM (This post was last modified: Today 08:20 PM by Valentin Albillo.) **Post: #20**

Valentin Albillo
Senior Member

Posts: 792
Joined: Feb 2015
Warning Level: 0%

RE: [VA] SRC #010 - Pi Day 2022 Special

> **Albert Chan Wrote:** (Today 01:29 PM)
>
> It is not hard to see why for log sums, we have error O(√(n) [...] You might try sum terms from index of 2 to n, instead of in reverse. I would guess you would produce similar sized error for PN
> [...]
> Using ln(C) correction, "true" PN = 3.14159 42243 85727 33456 11796 83910 689
>
> PN(n=1e6) errors = 98,928,578,031,286 ULP ≈ 1e6 ^ 2.33

Thanks, **Albert Chan**, that explains a lot, and it also explains why I found it difficult to believe such big errors were possible while doing pretty basic arithmetic with 34-digit precision.

I reckoned that I would lose about *6-7 digits* due to rounding/truncation but in the end, I was losing as much as ***11*** digits for ***N****=100,000* (let alone for *N=1,000,000*) because the internal code used in ***Free42*** for large integer exponents is seriously *flawed*. Serves me right for blindly trusting it ! :)

And if it were only that ... there are other *incredibly newbie-style*, *face palm* errors in some ***Free42*** math operations but I'll leave that for ***J-F Garnier***'s thread.

Regards.
V.

Edited to include a link to J-F's thread.

All My Articles & other Materials here: **Valentin Albillo's HP Collection**

PM WWW FIND EDIT QUOTE REPORT

« Next Oldest | Next Newest »

Enter Keywords Search Thread

NEW REPLY

View a Printable Version
Send this Thread to a Friend
Subscribe to this thread

User(s) browsing this thread: Valentin Albillo*, 2 Guest(s)