HP Forum Archive 20

Pi Day's ramblings

*Message #1 Posted by **Valentin Albillo** on 14 Mar 2011, 1:02 p.m.*

Hi all, long time no see:

Today's Pi Day, so I feel like indulging in some quick Pi antics to commemorate the event, namely showing off two little-known, rarely seen, extremely short ways to compute this most ubiquitous constant.

I'll use the HP-71B (+ MathROM) to enter and run the algorithms described. If you'd like to key them in as shown and don't own an HP-71B, get the excellent freeware emulator Emu71 by J-F Garnier (Google for it) or any other capable emulator of your choice (Math ROM emulation required) or that failing try and adapt the code to your favorite HP calculator.

Let's see:

1. **An asymptotically exact, probabilistic way:**

   This 50-byte one-liner will compute Pi using random numbers, you'll have to input how many tries (say 100,000) and it'll display the resulting Pi approximation at the end. The more tries the more correct Pi digits you'll get:

   ```
   10 INPUT K @ N=0 @ FOR I=1 TO K @ N=N-MOD(IROUND(RND/RND),2) @ NEXT I @ DISP 1-4*N/K
   ```

   To run it, key it in and then key in the following at the ">" prompt:

   ```
   >DESTROYALL @ RANDOMIZE 260 @ FIX 4    [Enter]
   >RUN   [Enter]


      ? 100000  [Enter]


         3.1416
   ```

As I said, the more tries the more correct digits you'll get. See if you can figure how and why it works, and if so check if you can prove that it actually produces the exact value of Pi in the limit.

2. **A faster, very precise, approximate way:**

Do the following:

a) Compute the real root of the cubic equation $\mathbf{x^3 - 6*x^2 + 4*x - 2}$**:**

```
>DESTROYALL @ STD
>FNROOT(-10,10,FVAR^3-6*FVAR^2+4*FVAR-2)


    5.31862821775
```

b) raise it to the 24th power:

```
>RES^24


    2.62537412641E17
```

c) compute the natural logarithm of the result:

```
>LN(RES)


    40.1091699911
```

d) Divide the result by the square root of 163:

```
>RES/SQR(163)


    3.14159265359
```

You'll get Pi correct to all displayed digits and then some. Again, see if you can figure how and why this works and if so, check if you can prove that it won't produce an exact value of Pi but just a fairly good approximation.

That's all. Best regards from V.

Re: Pi Day's ramblings

Message #2 Posted by ***gene wright*** *on 14 Mar 2011, 2:59 p.m.,*
in response to message #1 by Valentin Albillo

Hi Valentin. Good write-up. Give me a shout!

Re: Pi Day's ramblings

Message #3 Posted by ***Valentin Albillo*** *on 15 Mar 2011, 6:46 a.m.,*
in response to message #2 by gene wright

> Quote:
>
> Hi Valentin. Good write-up. Give me a shout!

Thanks, Gene, I'll give you "a shout" this next weekend for sure, sooner if I can manage.

Best regards from V.

Re: Pi Day's ramblings

Message #4 Posted by ***Fernando del Rey*** *on 14 Mar 2011, 3:43 p.m.,*
in response to message #1 by Valentin Albillo

Hi Valentín, welcome back!!!

I will try the two programs in my 71+MathROM, which I haven't used in ages. It should be fun.

Regards

Re: Pi Day's ramblings

Message #5 Posted by ***Valentin Albillo*** *on 15 Mar 2011, 6:56 a.m.,*
in response to message #4 by Fernando del Rey

Hi, Fernando !! :D

> Quote:

> Hi Valentín, welcome back!!!
>
> I will try the two programs in my 71+MathROM, which I haven't used in ages. It should be fun.

Yes, it should.

You might also want to try this Java version for the first one as well:

```
	private static void pi() {
		double pi = 0.0;
		for (long i = 0; i < Long.MAX_VALUE; i++) {
			pi = pi - Math.round((Math.random()/Math.random()))%2;
			if ((i&0x80000)==0x80000) {
				System.out.printf("%d: %s%n", i, 1-4*pi/i);
			}
		}
	}
```

I left it running yesterday on one of my servers, all night long (16 hours), and it produced the following:

```
...
487,446,282,239 tries  -> 3.141592254139215
487,447,330,815 tries  -> 3.141592246519439
487,448,379,391 tries  -> 3.141592254556738
...
```

which is as expected, 7 correct digits for some 500 (US) billion tries. Only one core was used and it managed *22,000 tries per millisecond*. Not bad.

The result also shows that Java's random number generator is working Ok, else there would be some bias which would be noticeable in the final result's accuracy.

Best regards from V.

Re: Pi Day's ramblings

Message #6 Posted by ***Dieter*** *on 14 Mar 2011, 4:59 p.m.,*
in response to message #1 by Valentin Albillo

Regarding the second challenge: that's not too hard to solve once you got access to some symbolic math software - Wolfram Alpha will do. ;-)

Determine the real solution of the polynomial, raise the result to the 24th power, take the natural log and divide by sqrt(163), and you'll get a result, whose continued fraction representation looks quite familiar:

```
[3; 7, 1, 15, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, 1, 1, ...]
```

Right, these are the first sixteen terms of the continued fraction representation of Pi. Its first seventeen significant digits agree with the true value.

I'm sure someone else will provide a decutive proof. ;-)

Dieter

Re: Pi Day's ramblings

Message #7 Posted by ***gene wright*** *on 14 Mar 2011, 5:07 p.m.,*
in response to message #1 by Valentin Albillo

Valentin wrote: "that it won't produce an exact value of Pi"

What is this exact value anyway? :-)

Re: Pi Day's ramblings

Message #8 Posted by ***Dieter*** *on 14 Mar 2011, 5:12 p.m.,*
in response to message #7 by gene wright

Simple: the exact value of Pi is... Pi. :-)

Dieter

Re: Pi Day's ramblings

Message #9 Posted by ***Gerson W. Barbosa*** *on 14 Mar 2011, 5:43 p.m.,*
in response to message #1 by Valentin Albillo

Hello Valentin,

Welcome back!

I've used WolframAlpha to solve the equation (solve x^3 - 6*x^2 + 4*x - 2 = 0 for x) and asked for more digits and repeated the steps using the windows calculator:

```
5.31862821775018565910968015331802247^24


262537412640768767.99999999999922


40.10916999113251984676562611108


3.1415926535897932456228612777983
```

When I compared the result to pi

```
3.1415926535897932384626433832795
```

I found the first 17 digits to match.

However, if 24 is subtracted from the result between steps b and c, the approximation is even more impressive:

```
ln(5.31862821775018565910968015331802246772191980883690026023^24-24)/sqrt(163) =


3.14159265358979323846264338327950319783820424737150850414


3.141592653589793238462643383279502884197169399375105820974945 (pi)
```

I have no idea why this works though, even after checking this MathWorld reference:

http://mathworld.wolfram.com/RamanujanConstant.html

Best regards,

Gerson.

Re: Pi Day's ramblings

Message #10 Posted by ***Valentin Albillo*** *on 15 Mar 2011, 7:07 a.m.,*
in response to message #9 by Gerson W. Barbosa

.
Hi, Gerson:

> Quote:
>
> Hello Valentin,
>
> Welcome back!

Thanks.

> Quote:
>
> I've used Wolfram Alpha to solve the equation (solve x^3 - 6*x^2 + 4*x - 2 = 0 for x) and asked for more digits and repeated the steps using the windows calculator:
>
> However, if 24 is subtracted from the result between steps b and c, the approximation is even more impressive:

Yes, I knew. However, I chose to omit the subtraction of 24 as the final result was close enough to Pi that it would nevertheless be indistinguishable in 10-digit and 12-digit calculators, and you wouldn't be able to notice it unless you resorted to multiprecision computations which would deny the simplicity, so I opted for one step less.

Best regards from V.
.

Re: Pi Day's ramblings

Message #11 Posted by ***Crawl*** *on 14 Mar 2011, 9:06 p.m.,*
in response to message #1 by Valentin Albillo

The first one implies that the probability of one (evenly distributed between 0 and 1) random variable divided by another having a result between (.5 to 1.5) or (2.5 to 3.5) or (4.5 to 5.5) or (6.5 to 7.5) or ... is

(pi - 1) / 4.

Wikipedia talks about ratio distributions, and in fact that one in particular here.

Using the result there, you then have to integrate between the limits in the first paragraph, giving

1/4 + 1/2*((1 - 2/3) + (2/5 - 2/7) + (2/9 - 2/11) + (2/13 - 2/15) +...

(the first term is the hardest one: You have to break up the first integral into one from 0.5 to 1, of 1/2; and one from 1 to 1.5, of 1/(2x^2). The others are all of 1/(2x^2), for the whole range)

or -1/4 + (1-1/3+1/5-1/7+...)

That infinite series is the Leibniz formula for pi, so the probability is

-1/4 + pi / 4.

which is what was to be shown.

Re: Pi Day's ramblings

Message #12 Posted by ***Crawl*** *on 14 Mar 2011, 9:15 p.m.,*
in response to message #11 by Crawl

By the way, I do think that is a more interesting example than the well-known Buffon needle problem. That is supposedly an example of pi coming up in a non-obvious situation, but I don't agree. It's obvious that when you toss a needle, the possible angles the needle could trace out when it lands will make a circle. And in fact you use circles and geometry to derive the probability.

In this example, however, it's really not obvious how pi is involved. And geometry is not used in the derivation.

Re: Pi Day's ramblings

Message #13 Posted by ***Valentin Albillo*** *on 15 Mar 2011, 7:14 a.m.,*
in response to message #12 by Crawl

> Quote:
>
> By the way, I do think that is a more interesting example than the well-known Buffon needle problem.

Agreed. As far as obtaining Pi via Monte-Carlo methods (algorithms using random numbers) I think that this is by far the simplest and most elegant, apart from being utterly non-obvious.

Best regards from V.

.

Re: Pi Day's ramblings

*Message #14 Posted by **Valentin Albillo** on 15 Mar 2011, 7:11 a.m.,*
in response to message #11 by Crawl

> Quote:
>
> ---
>
> The first one implies that the probability of one (evenly distributed between 0 and 1) random variable divided by another having a result between (.5 to 1.5) or (2.5 to 3.5) or (4.5 to 5.5) or (6.5 to 7.5) or ... is
>
> (pi - 1) / 4.
>
> ---

Correct.

Best regards from V.

Re: Pi Day's ramblings

*Message #15 Posted by **Paul Dale** on 15 Mar 2011, 7:17 a.m.,*
in response to message #1 by Valentin Albillo

How about using the following loop to get an estimate:

```
    RAN#
    RAN#
    R->P
    IP
    SIGMA+
```

Any takers to finish this?

Had an open day when I was studying mathematics at university. We did this dartboard PI estimate and the value was running a bit low. In stepped one of the older tutors who proceeded to correct the estimate a bit by putting a pile of darts on the same spot :-)

- Pauli

Re: Pi Day's ramblings

Message #16 Posted by ***Gerson W. Barbosa*** *on 16 Mar 2011, 8:49 a.m.,*
in response to message #15 by Paul Dale

HP-33s program

```
P0001 LBL P
P0002 STO N
P0003 CLSIGMA
R0001 LBL R
R0002 RANDOM
R0002 RANDOM
R0004 y,x->theta,r
R0005 IP
R0006 SIGMA+
R0007 RCL N
R0008 n
R0009 x<y?
R0010 GTO R
R0011 1/x
R0012 x<>y
R0013 SIGMAx
R0014 -
R0015 *
R0016 4
R0017 *
R0018 RTN


P: LN= 9 CK=B364
R: LN=66 CK=E2BB
```

Regards,

Gerson.

Re: Pi Day's ramblings

Message #17 Posted by ***Gerson W. Barbosa*** *on 16 Mar 2011, 10:31 a.m.,*
in response to message #16 by Gerson W. Barbosa

Average of 14 runs of 7 XEQ P:

```
44/14 or 22/7 (YMMV)
```

Free42 might be an option for much faster results:

```
00 { 31-Byte Prgm }
01 LBL "PI"
02 STO 00
03 CLSIGMA
04 LBL 00
05 RAN
06 RAN
07 ->POL
08 IP
09 SIGMA+
10 RCL 00
11 RCL 16
12 x<y?
13 GTO 00
14 1/X
15 X<>Y
16 RCL 11
17 -
18 *
19 4
20 *
21 RTN
22 .END.
```

Re: Pi Day's ramblings

Message #18 Posted by ***Dieter*** *on 16 Mar 2011, 2:55 p.m.,*
in response to message #17 by Gerson W. Barbosa

May I humbly suggest some improvements? :-)

Since the number of loops is known I think using DSE is the obvious way to go here. When the program finishes the values for n and N are the same, so there is no need to distinguish between both. And since only the sum of all x-values is used, the relatively slow Sigma+ is not required either.

Here's a version for the HP-35s which also shows how to get along without the infamous R->P command. ;-)

```
P001 LBL P
P002 INPUT N   ; get and save n
P003 STO K     ; initialize loop counter
P004 CLX
```

```
P005 STO S     ; initialize sum
P006 RANDOM
P007 RANDOM
P008 i
P009 x
P010 +
P011 ABS       ; = sqrt(random1^2 + random2^2)
P012 IP
P013 STO+ S    ; add 0 or 1
P014 DSE K     ; decrement and check loop counter
P015 GTO P006
P016 RCL N
P017 RCL- S
P018 RCL/ N
P019 4
P020 x         ; result = 4*(n-sum)/n
P021 RTN
```

My 35s runs 1000 loops in 2:45 minutes.

Edit: here's another version that requires just one memory register.

```
P001 LBL P
P002 INPUT N   ; get and save n
P003 CLSTK     ; clear sum
P004 RCL N
P005 R^        ; save n in stack
P006 RANDOM
P007 i
P008 x
P009 RANDOM
P010 +
P011 ABS       ; = sqrt(random1^2 + random2^2)
P012 IP
P013 +         ; add 0 or 1
P014 DSE N     ; decrement and check loop counter
P015 GTO P006
P016 -
P017 X<>Y
P018 /
P019 4
P020 x         ; result = 4*(n-sum)/n
P021 RTN
```

Dieter

Edited: 16 Mar 2011, 3:15 p.m.

Re: Pi Day's ramblings

Message #19 Posted by ***Gerson W. Barbosa*** *on 16 Mar 2011, 6:21 p.m.,*
in response to message #18 by Dieter

> Quote:
>
> May I humbly suggest some improvements? :-)

Of course, and don't be humble: your improvements are perfect!

I imagined Sigma+ was slow but I wanted to use all the instructionS Paul had provided. Prior to using Sigma+, I tested the idea in the third program below, which runs 1000 loops in 1 minute and 46 seconds (at first, I had used an HP-15C). As a comparison, the HP-33s program above runs in 2 minutes and 1 second, using Sigma+; I was expecting it to take more time.

The first program below is based in another (not so good) idea I had tried on the 15C. The indirect addressing slows it a bit. The second one is based on your first version, except the R->P instruction is still being used. It appears the newer machines don't benefit much from the faster DSE and ISG loop control instructions.

Regards,

Gerson.

```
T0001 LBL T
T0002 STO N
T0003 CLx
T0004 STO A
T0005 STO B
V0001 LBL V
V0002 RANDOM
V0003 RANDOM
V0004 y,x->theta,r
V0005 IP
V0006 1
V0007 +
V0008 STO i
V0009 STO+(i)
V0010 DSE N
V0011 GTO V
```

```
V0012 2
V0013 RCL* A
V0014 RCL+ B
V0015 RCL/ A
V0016 8
V0017 /
V0018 1/x
V0019 RTN
```

2' 13"

```
T0001 LBL T
T0002 STO N
T0003 STO A
T0004 CLx
T0005 STO S
V0001 LBL V
V0002 RANDOM
V0003 RANDOM
V0004 y,x->theta,r
V0005 IP
V0006 STO+ S
V0007 DSE A
V0008 GTO V
V0009 RCL N
V0010 RCL- S
V0011 RCL/ N
V0012 4
V0013 *
V0014 RTN
```

2'03"

```
T0001 LBL T
TO002 STO N
T0003 CLx
T0004 STO A
T0005 STO B
V0001 LBL V
```

```
V0002 RANDOM
V0003 RANDOM
V0004 y,x->theta,r
V0005 1
V0006 x>y?
V0007 STO+ B
V0008 STO+ A
V0009 RCL N
V0010 RCL A
V0011 x<y?
V0012 GTO V
V0013 1/x
V0014 RCL* B
V0015 4
V0016 *
V0017 RTN


1' 46"
```

Re: Pi Day's ramblings

Message #20 Posted by ***Dieter*** *on 16 Mar 2011, 7:05 p.m.,*
in response to message #19 by Gerson W. Barbosa

Hi Gerson,

> Quote:
>
> The first program below is based in another (not so good) idea I had tried on the 15C. The indirect addressing slows it a bit. The second one is based on your first version, except the R->P instruction is still being used. It appears the newer machines don't benefit much from the faster DSE and ISG loop control instructions.

This is an interesting point. Obviously every calculator has its specific "slow" and "fast" instructions. Regarding the 35s I can say that especially numeric constants are processed *very* slowly. So I often use workarounds like these:

```
   constant    workaround
    (slow)      (faster)
     1          Clx e^x  (or simply SGN if x>0)
     2          e   IP
     3          Pi  IP
    ...         ...
```

There also is another advantage: on the 35s the "fast" versions even use less memory.

Back to the various Pi-programs. Looking at the second 35s-version I wondered how fast or slow the 35s might process complex numbers, so I replaced lines P006 to P011 by a straightforward...

```
P006 RANDOM
P007 x²
P008 RANDOM
P009 x²
P010 +
P011 SQRT
```

...and guess what: it's much faster! 1000 loops are now done in 2:09.

Dieter

Re: Pi Day's ramblings

Message #21 Posted by ***Gerson W. Barbosa*** *on 16 Mar 2011, 7:32 p.m.,*
in response to message #20 by Dieter

> Quote:
>
> Looking at the second 35s-version I wondered how fast or slow the 35s might process complex numbers, so I replaced lines P006 to P011 by a straightforward...
>
> ```
> P006 RANDOM
> P007 x²
> P008 RANDOM
> P009 x²
> P010 +
> P011 SQRT
> ```
>
> ...and guess what: it's much faster! 1000 loops are now done in 2:09.

Likewise when these instructions replace lines V0002 through V0004 in the third HP-33s program above, the running time drops from 1min 46sec to 1min 04sec. However, I think Paul Dale was concerned about size rather than speed.

Actually SQRT is not necessary. If both terms of SQRT(random$\#_1^2$ + random$\#_2^2$) > 1 are squared, the inequality still holds. Thus, when SQRT is removed the running time for 1000 loops drops down to 53 seconds for the HP-33s program and to about 2 minutes for your 35s program.

Edited: 16 Mar 2011, 9:54 p.m.

Re: Pi Day's ramblings

Message #22 Posted by ***Paul Dale*** *on 17 Mar 2011, 1:51 a.m.,*
in response to message #21 by Gerson W. Barbosa

> Quote:
>
> However, I think Paul Dale was concerned about size rather than speed.

I wasn't. I just wanted to see what the folks here could do to the dart board approach to finding Pi :-)

I haven't been disappointed by the interesting programs.

- Pauli

Re: Pi Day's ramblings

Message #23 Posted by ***Dieter*** *on 17 Mar 2011, 8:47 a.m.,*
in response to message #21 by Gerson W. Barbosa

Right, the square root is obsolete - that's what I also realized after my last message yesterday.
The modified program now takes 1:55 minutes. So it's shorter *and* faster. ;-)

Dieter

Re: Pi Day's ramblings

Message #24 Posted by ***Gerson W. Barbosa*** *on 17 Mar 2011, 9:25 a.m.,*
in response to message #23 by Dieter

Hello Dieter,

DSE is really slower on the HP-33s, as we can see from the timings belows. Is it also slower on the HP-35s? Thanks!

```
HP-33s & HP-32SII


T0001 LBL T                     T0001 LBL T
TO002 STO N                     TO002 STO N
T0003 CLx                       T0003 CLx
T0004 STO A                     T0004 STO A
T0005 STO B                     T0005 STO B
V0001 LBL V                     V0001 LBL V
V0002 RANDOM                    V0002 RANDOM
V0003 x^2                       V0003 x^2
V0004 RANDOM                    V0004 RANDOM
V0005 x^2                       V0005 x^2
V0006 +                         V0006 +
V0007 1                         V0007 1
V0008 x>y?                      V0008 x>y?
V0009 STO+ B                    V0009 STO+ B
V0010 STO+ A                    V0010 STO+ A
V0011 RCL N                     V0011 DSE N
V0012 RCL A                     V0012 GTO V
V0013 x<y?                      V0013 4
V0014 GTO V                     V0014 RCL* B
V0015 1/x                       V0015 RCL/ A
V0016 RCL* B                    V0016 RTN
V0017 4
V0018 *
V0019 RTN


1,000 loops timing


0' 53" (HP-33s)                 1' 14" (HP-33s)
1' 28" (HP-32SII)               1' 26" (HP-32SII)



HP-42S


00 { 38-Byete Prgm }
01>LBL "PI"
02 STO 00
03 CLX
04 STO 01
```

```
05 STO 02
06>LBL 00
07 RAN
08 x^2
09 RAN
10 x^2
11 +
12 1
13 X>Y?
14 STO+ 02
15 STO+ 01
16 DSE 00
17 GTO 00
18 4
19 RCL* 02
20 RCL/ 01
21 RTN
22 .END.


30,000,000 loops timing


1' 10" ( Free42 Binary 1.4.67 @ 1.86 GHz )
```

Re: Pi Day's ramblings

Message #25 Posted by ***Dieter*** *on 17 Mar 2011, 10:27 a.m.,*
in response to message #24 by Gerson W. Barbosa

On the 35s I excpected a version without DSE to run even slower, as it requires a numeric constant ("1") within the loop. And so it was:

```
P001 LBL P
P002 INPUT N
P003 CLSTK
P004 RCL N
P005 R^
P006 ENTER
P007 Clx
P008 RANDOM
P009 x^2
P010 RANDOM
P011 x^2
P012 +
P013 IP
```

```
P014 +
P015 1
P016 STO- N
P017 Clx
P018 RCL N
P019 X>0?
P020 GTO P007
P021 RDN
P022 -
P023 X<>Y
P024 /
P025 4
P026 x
P027 RTN


Running time: 2:16
```

As mentioned before there are several ways to avoid numeric constants:

```
P015 RCL N                    P015 RCL N
P016 SGN                      P016 ENTER
P017 STO- N                   P017 SGN
P018 Clx                      P018 -
P019 RCL N                    P019 STO N
P020 X>0?                     P020 X>0?
P021 GTO P007                 P021 GTO P007
...                           ...


Running time: 1:55            Running time: 1:59
```

So the two last solutions take about the same time as the original version with DSE.

Dieter

Re: Pi Day's ramblings

Message #26 Posted by ***Crawl*** *on 17 Mar 2011, 9:39 a.m.,*
in response to message #17 by Gerson W. Barbosa

Quote:

Average of 14 runs of 7 XEQ P:

```
44/14 or 22/7 (YMMV)
```

There's a good point in there. pi is known to be approximately 22/7, or, even better, approximately 355/113.

If you used a multiple of 113 as the number of trials, you might be able to get very close this way.

The same idea is suggested in the article on Buffon's needle I linked above.

Re: Pi Day's ramblings

Message #27 Posted by ***Dieter*** *on 17 Mar 2011, 10:47 a.m.,*
in response to message #26 by Crawl

Crawl wrote:

> Quote:
>
> If you used a multiple of 113 as the number of trials, you might be able to get very close this way.

Since the experiment approximates Pi/4 I would suggest an integer multiple of 4*113 = 452.

This way n*452 trials will result in Pi ~= 355/113 if the number of hits is n*355.

Dieter

Re: Pi Day's ramblings

Message #28 Posted by ***Gerson W. Barbosa*** *on 17 Mar 2011, 2:20 p.m.,*
in response to message #26 by Crawl

> Quote:
>
> If you used a multiple of 113 as the number of trials, you might be able to get very close this way.

I had tried that earlier, but to no avail. I was trying single runs of 113 trials. I've finally made it with a single run of 452, as per Dieter's suggestion, the first time I tried. I little cheating though in order to ensure immediate success ;-)

On the real HP-42 and the 38-byte program above:

```
     94 SEED 452 XEQ PI  =>  3.14159292035  (after 1 min 45 sec)
```

On Free42 Binary 1.4.67:

```
66 SQRT SEED 452 XEQ PI  =>  3.14159292035  (instantaneously)
```

```
00 { 34-Byte Prgm }
01>LBL "X"
02 STO 04
03>LBL 01
04 RCL 04
05 SQRT
06 SEED
07 452
08 XEQ "PI"
09 PI
10 -
11 ABS
12 1E-4
13 X>Y?
14 STOP
15 DSE 04
16 GTO 01
17 RTN
18 .END.
```

On the real 42S, line 05 is not necessary to find a "right" seed for the random numbers generator. That's because Free42 will always generate the same random number for seeds with same fractional parts, except when the seed is set to 0, when SEED behaves as it does on the real 42S, that is, always generates internally a different random number:

```
                          Free42             HP-42S
1   SEED   RAN  =>  5.23054829358E-1    7.31362440213E-1
2   SEED   RAN  =>  5.23054829358E-1    4.31362440213E-1
3   SEED   RAN  =>  5.23054829358E-1    1.31362440213E-1
3.1 SEED   RAN  =>  1.20349678274e-1    8.01362440213E-1
4.1 SEED   RAN  =>  1.20349678274e-1    5.01362440213E-1
5.1 SEED   RAN  =>  1.20349678274e-1    2.01362440213E-1
0   SEED   RAN  =>   undetermined         undetermined
0   SEED   RAN  =>   undetermined         undetermined
```

On Free42 Binary 1.4.67 @ 1.86 GHz:

```
9236 SQRT SEED 132408 XEQ PI  =>  3.14159265301  (about half a second)
```

Edited: 18 Mar 2011, 10:48 p.m.

Those days...

Message #29 Posted by ***Frank Boehm (Germany)*** *on 15 Mar 2011, 8:28 a.m.,*
in response to message #1 by Valentin Albillo

I remember a "Spektrum der Wissenschaft" (the german release of "Scientific American") essay about "shooting at pi", using the random generator. If I remember correctly, I used my newly purchased Amiga to do the calculations and graphical representation. Lots of fun. While googling for the essay, I stumbled across this:
Pi
Looks like someone invested quite a bit of time for pi day ;)

Re: Those days...

Message #30 Posted by ***Palmer O. Hanson, Jr.*** *on 15 Mar 2011, 11:41 p.m.,*
in response to message #29 by Frank Boehm (Germany)

A. K. Dewdney's column "Computer Recreations" in the April 1985 issue of *Scientific American* discussed this methodology. He used the analogy of a cannon firing into a square field which included an inscribed circular pond. Readers were asked to send the results for 1000 shots to him.

Re: Pi Day's ramblings

Message #31 Posted by ***Dieter*** *on 15 Mar 2011, 12:58 p.m.,*
in response to message #1 by Valentin Albillo

Here's my approach for the second challenge.

The key is a constant found by Ramanujan who showed that

$$R = e^{(Pi * sqrt(163))}$$

is *almost* an integer. In fact, the exact value for R is

```
R = 262 537 412 640 768 743,999 999 999 999 250...
```

and so

```
R = ~= 262 537 412 640 768 744
```

In other words:

```
Pi ~= sqrt(163) * ln R
```

The value of this expression agrees with Pi in its first 30 decimals.

Now, R also is the intermediate value 2,62537412641 E+17 mentioned in the challenge. At least is looks quite close - the given 12 digits are the same as in R.

So the remaining question is: Why is the real solution of the given cubic equation, raised to the 24th power, so close to R? Or, more precisely, indistinguishable in its first 12 digits.

The analytic solution to the given cubic equation is

$$x = 2 + (5 + \mathrm{sqrt}(489)/9)^{1/3} + (5 - \mathrm{sqrt}(489)/9)^{1/3}$$
$$= 5{,}318\ 628\ 217\ 750\ 185\ 659\ 109\ 680\ \ldots$$

Now, the 24th power of x is

$$x^{24} = 262\ 537\ 412\ 640\ 768\ 767{,}999\ 999\ 999\ 999\ 251\ \ldots$$

which is quite exactly R + 24. Which in turn is close enough to R to that also

$$R \sim= x^{24} \sim= e^{(Pi * \mathrm{sqrt}(163))}$$

or finally

$$Pi \sim= \ln(x^{24}) * \mathrm{sqrt}(163)$$

which is the value that was calculated in the challenge.

This also explains why the Pi-approximation gets much better if the intermediate is corrected by 24, as Gerson mentioned.

However, there still is one thing left for the perfect solution:

Assume R resp. $R^{1/24}$ is given. Now find a cubic equation with integer coefficents where its real solution is as close as possible to this value.

Dieter