An iteration produces all the prime numbers

Threaded Mode | Linear Mode

13th February, 2021, 20:05 **Post: #1**

EdS2
Senior Member

Posts: 525
Joined: Apr 2014

An iteration produces all the prime numbers

This very much appeals to me: initially it's rather a surprise.

Take the number 2.920050977316134712092562917112019... and perform the following iteration:
x := IP(x)*FP(x)+IP(x)

(Where IP is the integer part and FP the fractional part.)

Note the integer parts of the resulting sequence!

Ref: this paper or this video or this wikipedia section or this OEIS page.

EMAIL PM FIND QUOTE REPORT

13th February, 2021, 23:38 (This post was last modified: 13th February, 2021 23:39 by Valentin Albillo.) **Post: #2**

Valentin Albillo
Senior Member

Posts: 970
Joined: Feb 2015
Warning Level: 0%

RE: An iteration produces all the prime numbers

.
Hi, **EdS2**:

> **EdS2 Wrote:** (13th February, 2021 20:05)
>
> This very much appeals to me: initially it's rather a surprise.
>
> Take the number 2.920050977316134712092562917112019... and perform the following iteration:
> x := IP(x)*FP(x)+IP(x)
>
> (Where IP is the integer part and FP the fractional part.)

There are infinitely many such constants that produce primes (this one, Mills' constant, etc) but though they're somewhat interesting from a *'recreational maths'* point of view, they're useless as prime-producing constants because you need to know in advance the prime sequence to compute them, which makes the subject rather *circular*:

You can use the constant to produce primes but you need the primes to produce the constant.

It would be quite another matter if any such constant could be computed to arbitrary precision some other way *without* involving the primes, say as a limit, an infinite summatory, an integral ... Alas, that's *never* the case so far and it's highly dubious it will ever be.

Regards.
V.

PM WWW FIND EDIT QUOTE REPORT

14th February, 2021, 14:38 **Post: #3**

EdS2
Senior Member

Posts: 525
Joined: Apr 2014

RE: An iteration produces all the prime numbers

Indeed, so, all true. I still liked what I saw!

Thanks for Mills' constant (1.3063778838630806904686144926...) - that's perhaps a little more accessible, as we only need an expression, not an iteration. (Then again, it produces only primes, less impressive to me than producing all primes, although that is of course a matter of taste.)

EMAIL PM FIND QUOTE REPORT

14th February, 2021, 14:49 **Post: #4**

peacecalc
Member

Posts: 187
Joined: Dec 2013

RE: An iteration produces all the prime numbers

Hello all,

is there a relation between the precision of that constant and which greatest prime you can produce with it. That would be fancy: Let's say you know the constant with n decimal precision and the highest prime you can produce is say p(n). And with that prime you can improve this constant to a precision n+1 (or more) and with that you can produce the greatest prime let it call P(n+1) which is the successor of p(n), or the next two (or more) successors of p(n)...

Why I have to think about Münchhausen's lift oneself up by his own bootstraps?

EMAIL PM FIND QUOTE REPORT

14th February, 2021, 17:05 **Post: #5**

ttw
Member

Posts: 265
Joined: Jun 2014

RE: An iteration produces all the prime numbers

For fairly simple methods of quickly producing primes (faster than the sieve of Eratosthenes), check out "wheels." The idea is to start with 2 (or seed with 2,3,5,7...) and as one goes, one dynamically changes the sieving process to pickup more primes. First one gets 6*K+(1,5) then 30*K+(1,7,11,13,17,19,23,29) and so forth. I like the 30*k method as I can store 30 blocks of 30 numbers in a single byte. Later, one needs fancier methods. It becomes more efficient to store the distances between primes. Then even store that distance using a Universal Code.

https://en.wikipedia.org/wiki/Wheel_factorization

https://en.wikipedia.org/wiki/Universal_...mpression)

EMAIL PM FIND QUOTE REPORT

14th February, 2021, 17:38 **Post: #6**

Albert Chan
Senior Member

Posts: 2,148
Joined: Jul 2018

RE: An iteration produces all the prime numbers

> **ttw Wrote:** (14th February, 2021 17:05)
> First one gets 6*K+(1,5) then 30*K+(1,7,11,13,17,19,23,29) and so forth. I like the 30*k method as I can store 30 blocks of 30 numbers in a single byte. Later, one needs fancier methods. It becomes more efficient to store the distances between primes. Then even store that distance using a Universal Code.

Storing primes using wheels is nice, but data compression ratio is still a constant

wheel(2,3), compression ratio = 1 / (1 - 1/2 - 1/3 + 1/(2*3)) = 3
wheel(2,3,5), compression ratio = 1 / (1 - 1/2 - 1/3 - 1/5 + 1/(2*3) + 1/(2*5) + 1/(3*5) - 1/(2*3*5)) = 3.75
...

Assuming we do not need random access, but simply O(1) way to get next prime (like OP formula).
Is there a way to do better ?

Does storing prime gaps give better compression ratio ?

EMAIL PM FIND QUOTE REPORT

15th February, 2021, 03:27 (This post was last modified: 15th February, 2021 03:29 by ttw.) **Post: #7**

ttw
Member

Posts: 265
Joined: Jun 2014

RE: An iteration produces all the prime numbers

One doesn't (for really big stuff) use the same wheel. When possible, one switches to a bigger wheel. Most of those I've seen use 30030 to store 5760 for an efficiency of 192/1001. Of course, the last time I did this seriously, I had a 100 megaword Cray YMP for computation (I didn't use the SSD) for 100,000,000 words *64 bits or 6,400,000,000 bits. Using 8 bits for 30 numbers (easy coding) gave me N=24,000,000,000.

The density of primes is about ln(N)/N which means the spacing is ln(n) on the average.
This clearly increases as N gets large. For N=6,400,000,000 the average spacing is /Ln(6,400,000,000 or about 22. The 30-wheel has uses 30 numbers for 8 prime so the spacing is 15/4 or 3.75. A list of spacings is better.

To make things more complicated, Yitang Zhang proved that there are infinitely many prime pairs with gap less than 70,000,000, the first result related to prime pairs. Of course,70,000,000 is bigger than 2. (The latest number gap which occurs infinitely often is 246.)

https://online.kitp.ucsb.edu/online/coll...m_KITP.pdf

EMAIL PM FIND QUOTE REPORT

15th February, 2021, 03:29 (This post was last modified: 15th February, 2021 03:41 by Valentin Albillo.) **Post: #8**

Valentin Albillo
Senior Member

Posts: 970
Joined: Feb 2015
Warning Level: 0%

RE: An iteration produces all the prime numbers

> **EdS2 Wrote:** (14th February, 2021 14:38)
>
> Indeed, so, all true. I still liked what I saw!
>
> Thanks for Mills' constant (1.3063778838630806904686144926...) - that's perhaps a little more accessible, as we only need an expression, not an iteration. (Then again, it produces only primes, less impressive to me than producing all primes, although that is of course a matter of taste.)

Mills' constant *(1.306377883863...)* produces primes which grow extremely fast as the article you linked states *(2, 11, 1361, 2521008887, 1602236204009818131831320183, ~O(1e84))*, each term has about *3x* the digits of the previous term, so using it you can only produce and certify the primality of a very few before you are forced to check their primality using a probabilistic method for a little while and afterwards even that won't be feasible.

However, there's an infinity of prime-generating *Mills*-like procedures but with much reduced growing rates so they can be used to generate proven primes by the *hundreds*. On the other hand, there are some *iterative* procedures (many trivial, but not all) that do *not* depend on the accuracy of an irrational (transcendental ?) constant to generate an indefinite number of fully-certified primes.

Primes are always a source of awe. One of the many many things that awed me is that you can generate the sequence of primes using the sequence of zeros of the *Riemann's Zeta* function, and you can in turn generate the zeros of the function using the primes. Perfectly symmetrical, one sequence encodes the other.

By the way, the constant mentioned in your linked article can be generated to the full ***13*** *digits* given there by running this trivial 2-line *HP-71B* program, which produces the value in no time:

```
1   DESTROY ALL @ P=2 @ M=P @ S=-1
2   FOR I=1 TO 12 @ P=FPRIM(P+1) @ S=S+(P-1)/M @ M=M*P @ NEXT I @ DISP " 2";STR$(S)

>RUN
        2.920050977316
```

Regards.
V.

PM WWW FIND EDIT QUOTE REPORT

15th February, 2021, 19:02 **Post: #9**

KeithB
Senior Member

Posts: 382
Joined: Jan 2017

RE: An iteration produces all the prime numbers

What is FPRIM?

EMAIL PM FIND QUOTE REPORT

15th February, 2021, 19:34 (This post was last modified: 15th February, 2021 19:37 by Valentin Albillo.) **Post: #10**

Valentin Albillo
Senior Member

Posts: 970
Joined: Feb 2015
Warning Level: 0%

RE: An iteration produces all the prime numbers

> **KeithB Wrote:** (15th February, 2021 19:02)
> What is FPRIM?

FPRIM is a function which returns the next prime (either forward of backwards) from the given argument. For instance:

FPRIM(6)= 7, FPRIM(4,1)=3, FPRIM(5)=5

It can be found in the JPC ROM. As a matter of course, i always keep the HP-71B fitted with at least 150 Kb RAM, the MATH ROM, the JPC ROM, the HP-IL ROM and the STRINGLX LEX file. Anything less is a maimed HP-71B as far as i'm concerned.

V.

PM WWW FIND EDIT QUOTE REPORT

16th February, 2021, 17:34 **Post: #11**

EdS2
Senior Member

Posts: 525
Joined: Apr 2014

RE: An iteration produces all the prime numbers

13 digits - that's odd - I see the HP-71B described as having 12 digits. Can you explain please, Valentin?

(As a related question, I wonder if there's any extended precision library for the 71B or any other HP calculator, to allow us to work with more digits?)

EMAIL PM FIND QUOTE REPORT

17th February, 2021, 03:00 **Post: #12**

Valentin Albillo
Senior Member

Posts: 970
Joined: Feb 2015
Warning Level: 0%

RE: An iteration produces all the prime numbers

.
Hi, **EdS2**:

> **EdS2 Wrote:** (16th February, 2021 17:34)
> 13 digits - that's odd

Correct. 13 is indeed an odd integer.

> **Quote:**
> I see the HP-71B described as having 12 digits. **Can you explain** please, Valentin?

Yes. The value featured in your linked article is **2** followed by *12 decimals*, i.e., *13* digits in all.

To get the full 12 decimals in a 12-digit calc the integer part (**2**) gets in the way so my program simply gets rid of it by initializing the sum to *-1*, thus no *integer* part is ever involved and the 1*2-digit HP-71B* can merrily compute the 12-digit decimal part in its full glory (***.920050977316***), then it's a matter of just adding the **2** in front when printing and there you are, ***2.920050977316***, 13 digits.

> **Quote:**
> I wonder if there's any extended precision library for the 71B or any other HP calculator, to allow us to work with more digits?

Well, computing multiprecision results with the HP-71B is not that difficult at all once you've done it a few times, just for instance have a look at these PDf articles I wrote:

HP-71B Producing Digits of Pi one at a time
HP-71B Multiprecision E and its roots

which can compute thousands of digits of ***Pi*** and ***e*** (and its roots, square, etc), respectively. As for a multiprecision "library", have a look at the one I wrote for this challenge o'mine:

HP Challenge VA021 - Short Sweet Math Challenge 20 April 1st Spring Special

and in page **74** you'll find the following (abridged here):

Appendix A: The multiprecision library

*I'll briefly discuss here the small(just **32** lines of code) "Ad-hoc multiprecision library" I've implemented specifically for this particular subchallenge (the following listing is numbered as if to be included in the same file as the main program without line numbers conflicting, but could reside in a separate program file in RAM as well, if intended to be used by other programs).*

```
SUB MMUL(A(),B(),C())       { Multiprecision MULtiplication }
SUB MPOW(A(),N,C())       { Multiprecision POWer }
SUB MSUM(A(),B(),C())       { Multiprecision SUM }
SUB MSBI(A(),B())       { Multiprecision SuBtraction In-place }
SUB MMODK(A(),K,M)       { Multiprecision MODulus single-precision }
SUB MDIVK(A(),K)       { Multiprecision DIV (and modulus) single-precision }
SUB MNOR(A())       { Multiprecision NORmalization }
SUB MTOP(A(),U)       { Multiprecision TOP }
SUB M2N(A(),N)       { Multiprecision to real-precision Numeric variable }
SUB N2M(N,A())       { Numeric Real-precision variable to Multiprecision }
SUB MMOD(A(),B(),C())       { Multiprecision MODulus }
SUB MFAC(N,A())       { Multiprecision FACtorial }
```

Sample use:

69! MOD 13^{68} = 161707119156747337147933149666645422128978599226831537632782504385470771962

However, do not have any high hopes, it was written *ad-hoc* for the challenge so it's not *general purpose* by any means, though it might be useful to you as an example on how to proceed (read the *Caveats*).

Regards.
V.

PM WWW FIND EDIT QUOTE REPORT

17th February, 2021, 04:14 **Post: #13**

mfleming
Senior Member

Posts: 881
Joined: Jul 2015

RE: An iteration produces all the prime numbers

> **Valentin Albillo Wrote:** (15th February, 2021 19:34)
> <snip>
> As a matter of course, i always keep the HP-71B fitted with at least 150 Kb RAM, the MATH ROM, the JPC ROM, the HP-IL ROM and the STRINGLX LEX file. Anything less is a maimed HP-71B as far as i'm concerned. :)

The **HP-IL ROM** in your list, is that the ROM in the HP-IL peripheral, or some other (development?) ROM? I've seen the BIN files HPILROM-A.BIN, HPILROM-B.BIN, and HPILROM-H4.BIN with the first two being ROMs in the HP-IL peripheral. The -H4 bin file seems to be a version of the first two? I know STRINGLX functions can be found in J-F's ULIB52 ROM.

Just asking. I'm trying to put together a final list of available ROMs for the 71.

PM FIND QUOTE REPORT

17th February, 2021, 06:16 **Post: #14**

Valentin Albillo
Senior Member

Posts: 970
Joined: Feb 2015
Warning Level: 0%

RE: An iteration produces all the prime numbers

> **mfleming Wrote:** (17th February, 2021 04:14)
>
> The **HP-IL ROM** in your list, is that the ROM in the HP-IL peripheral, or some other (development?) ROM?

This one: **HP-IL adaptor 82401A,** i.e.: the plain-vanilla one.

> **Quote:**
>
> I know STRINGLX functions can be found in J-F's ULIB52 ROM.

Don't know, I'm using the ~838-byte individual *LEX* file, don't need anything else. You know, having many *LEX* files and *ROMs* with many keywords tends to slow down the *HP-71B*, so having the *MATH, JPC and HP-IL ROMs* is enough for me, I don't need more ROMs or LEX files.

> **Quote:**
>
> Just asking. I'm trying to put together a final list of available ROMs for the 71.

Good luck with that. I suppose you've consulted Sylvain's *HP-71B Compendium* and Joe Horn's and J-F's sites.

Regards.
V.

PM WWW FIND EDIT QUOTE REPORT

17th February, 2021, 09:10 **Post: #15**

mfleming
Senior Member

Posts: 881
Joined: Jul 2015

RE: An iteration produces all the prime numbers

> **Valentin Albillo Wrote:** (17th February, 2021 06:16)
>
> Good luck with that. I suppose you've consulted Sylvain's *HP-71B Compendium* and Joe Horn's and J-F's sites.

And MoHPC flash and HHC flash and PPC flash and ... anything else? :)

Thanks Valentin,
~Mark

PM FIND QUOTE REPORT

17th February, 2021, 10:09 **Post: #16**

J-F Garnier
Senior Member

Posts: 820
Joined: Dec 2013

RE: An iteration produces all the prime numbers

> **mfleming Wrote:** (17th February, 2021 09:10)
>
> > **Valentin Albillo Wrote:** (17th February, 2021 06:16)
> >
> > Good luck with that. I suppose you've consulted Sylvain's *HP-71B Compendium* and Joe Horn's and J-F's sites.
>
> And MoHPC flash and HHC flash and PPC flash and ... anything else? :)

and Matthias' module database.

J-F

EMAIL PM WWW FIND QUOTE REPORT

17th February, 2021, 12:00 **Post: #17**

EdS2
Senior Member

Posts: 525
Joined: Apr 2014

RE: An iteration produces all the prime numbers

.

> **Valentin Albillo Wrote:** (17th February, 2021 03:00)
>
> odd

One of my favourite jokes...

> **Quote:**
>
> > **Quote:**
> >
> > I see the HP-71B described as having 12 digits. **Can you explain** please, Valentin?
>
> **Yes**. The value featured in your linked article is **2** followed by *12 decimals*, i.e., *13* digits in all.
>
> To get the full 12 decimals in a 12-digit calc the integer part (**2**) gets in the way so my program simply gets rid of it by initializing the sum to *-1*, thus no *integer* part is ever involved and the 1*2-digit HP-71B* can merrily compute the 12-digit decimal part in its full glory...

Thanks for explaining!

> **Quote:**
>
> > **Quote:**
> >
> > I wonder if there's any extended precision library for the 71B or any other HP calculator, to allow us to work with more digits?
>
> Well, computing multiprecision results with the HP-71B is not that difficult at all once you've done it a few times, just for instance have a look at ...

Excellent - thanks again.

EMAIL PM FIND QUOTE REPORT

17th February, 2021, 23:57 **Post: #18**

Valentin Albillo
Senior Member

Posts: 970
Joined: Feb 2015
Warning Level: 0%

RE: An iteration produces all the prime numbers

Hi, **EdS2**:

> **EdS2 Wrote:** (17th February, 2021 12:00)
>
> Thanks for explaining!

You're welcome.

> **Quote:**
>
> Excellent - thanks again.

You're welcome again.

Re the **HP-71B** 2-liner I posted above, which instantly computes the constant as given in your linked article (13 decimal places, ***2.920050977316***), it's quite possible to convert it to a multiprecision program which would compute the constant to the digits shown in this *Wikipedia* article:

Formula for primes

where the constant is given as ***2.920050977316134712092562917112019***. However, for variety's sake let's use instead *SwissMicros* awesome **DM42** calc, which can run *HP-42S RPN* programs and supports *34-digit* precision instead of just 12-digit as both the *HP-42S* and the *HP-71B* do.

Thus, I wrote the following *50-step, 86-byte RPN* program for the **DM42** to perform the feat:

```
LBL "CCN"    1          5            X>Y?
SIZE 30      -          STO 00       GTO 01
 -1          RCL/ 02    LBL 01       RCL 04
STO 03       RCL+ 03    RCL IND 00   STO IND 01
```

```
  6          ENTER       RCL 04       ISG 01
STO 01       X<> 03      RCL/ ST Y    LBL 03
  2          X=Y?        FP           GTO 02
STO 02       GTO 04      X=0?         LBL 04
  3          RCL ST Z    GTO 00         2
STO 04       STOx 02     Rv             +
STO 05       LBL 00      LASTX        END
LBL 02         2         X>Y?
RCL ST X     STO+ 04     ISG 00
```

Let's execute it:

```
XEQ "CCN" -> 2.92005097732
   [SHOW] -> 2.920050977316134712092562917112019
```

which is computed instantly and exactly matches the *Wikipedia*'s value.

As the **DM42** doesn't have any *number-theoretic* functions in its instruction set (like `PRIM`, `FPRIM`, `GCD`, etc.), my program above generates and uses the prime numbers on the fly. *101* is the last prime needed.

BTW, the program is a quick job, also written on the fly, so it's not optimized to any extent as it already runs instantly and takes little program memory. It can be optimized by improving stack use but I see little need, be my guest if you want to try.

Regards.
V.

PM WWW FIND EDIT QUOTE REPORT

18th February, 2021, 10:42 **Post: #19**

EdS2
Senior Member

Posts: 525
Joined: Apr 2014

RE: An iteration produces all the prime numbers

very nice! I don't have a DM42, but I do have Free42 on my phone...

EMAIL PM FIND QUOTE REPORT