The Pell Prime Conjectures

Al Kelley
Department of Mathematics
University of California, Santa Cruz
Santa Cruz, CA 95064, USA
---
email: blufox@ucsc.edu

Abstract

For each positive integer $d \geq 2$ and d not square the Pell equation $x^2 - d\,y^2 = 1$ has an infinite number of positive integer solutions x, y. For a given d there is a smallest solution x, y with both x and y positive called the fundamental solution. These are the only solutions that we will consider. Using the x increasing algorithm to select solutions d of the Pell equation, we generate an infinite sequence of integers that are conjectured to be primes. The sequence is closely related to A033316 at oeis.org:

53, 61, 109, 181, 277, 397, 409, 421, 541, 661, 1021, 1069, 1381, 1549, …

Algorithm

Start with a non-square positive integer $d_0 \geq 2$ and let $(d_{0,} x_{0,} y_0)$ be the associated fundamental solution to the Pell equation. Choose the least $d_1 > d_0$ such that $x_1 > x_0$ where $(d_{1,} x_{1,} y_1)$ is the solution to the Pell equation associated with d_1. Similarly, choose the least $d_2 > d_1$ such that $x_2 > x_1$, where x_2 comes from the solution to the Pell equation associated with d_2. Continuing in this manner we get an increasing sequence of positive integers:

$d_0 < d_1 < d_2 < d_3$ ...

Getting started

Here is a list of fundamental solutions to the Pell equation starting with d = 2:

d	x	y
2	3	2
3	2	1
5	9	4
6	5	2
7	8	3
8	3	1
10	19	6
11	10	3
12	7	2
13	649	180
14	15	4
15	4	1
17	33	8
18	17	4
19	170	39
20	9	2
21	55	12
22	197	42
23	24	5

d	x	y
24	5	1
26	51	10
27	26	5
28	127	24
29	9801	1820
30	11	2
31	1520	273
32	17	3
33	23	4
34	35	6
35	6	1
37	73	12
38	37	6
39	25	4
40	19	3
41	2049	320
42	13	2
43	3482	531
44	199	30

d	x	y
45	161	24
46	24335	3588
47	48	7
48	7	1
50	99	14
51	50	7
52	649	90
53	66249	9100
54	485	66
55	89	12
56	15	2
57	151	20
58	19603	2574
59	530	69
60	31	4
61	1766319049	226153980
62	63	8
63	8	1
65	129	16

For d in the range $2 \le d \le 108$ it is well known that d = 61 produces the largest x, y values. We want to use the above list of solutions to the Pell equation to start building our sequence of integers. From the first line in the list we see that d = 2, x = 3, y = 2 is a fundamental solution to the Pell equation. Starting with the x value of 3 in the second column and writing down only the increasing values of x as we go down the column, we get

3, 9, 19, 649, 9801, 24335, 66249, 1766319049

The corresponding d values are

2, 5, 10, 13, 29, 46, 53, 61

We use our C++ program called *pell* [it will be described below] to get more d values that correspond to increasing x values. Here is some output from the pell program:

```
   2                                          3                                          2
   5                                          9                                          4
  10                                         19                                          6
  13                                        649                                        180
  29                                       9801                                       1820
  46                                      24335                                       3588
  53                                      66249                                       9100
  61                                 1766319049                                  226153980
 109                             158070671986249                              15140424455100
 181                         2469645423824185801                        183567298683461940
 277                        159150073798980475849                       9562401173878027020
 397                        838721786045180184649                      42094239791738433660
 409                       25052977273092427986049                     1238789998647218582160
 421             3879474045914926879468217167061449               189073995951839020880499780706260
 541           3707453360023867028800645599667005001           159395869721270110077187138775196900
 661          16421658242965910275055840472270471049          638728478116949861246791167518480580
1021 198723867690977573219668252231077415636351801801  6219237759214762827187409503019432615976684540
1069 742925865816843150858935268959512942700219559049 22722526912283010072320240710785462723519145740
1381                                      #65:9                                      #64:0
1549                                      #71:1                                      #70:0
```

Observe that in the last two rows the values of x and y are not printed, because the values are too large to fit on one line. In the last row with d = 1549 the program printed instead #71:1 and #70:0, which means that x has 71 digits and ends in 1 and that y has 70 digits and ends in 0. The pell program writes the actual values of x and y corresponding to each d into a file. Thus the values are available if for some reason we want to see them. Here, for example, are the values of x and y corresponding to d = 1549:

```
x = 48106848972197087743588687481413975084698632248110750633952591202305801
y = 1222309542826747495934242683346380508818076263178681966098672827963220
```

Conjecture

Starting at d = 53 and using the increasing x algorithm described above, we generate an infinite increasing sequence of integers that we conjecture consists of primes, the ***Pell primes***:

53, 61, 109, 181, 277, 397, 409, 421, 541, 661, 1021, 1069, 1381, 1549, 1621, 2389, 3061, 3469, 4621, ...

The gmod and the signed count

Our pell program can be used to produce further information that we now want to illustrate. Here is some program output that again starts at d = 2 and continues with successive values of d:

```
  2  p    2:2:2:2:2:2:2   0     3    2
  3  p    0:3:3:3:3:3:3  -1*    2    1
  5  p    2:1:0:5:5:5:5  +1*    9    4
  6       0:2:1:0:6:6:6  -1*    5    2
  7  p    1:3:2:1:0:7:7  +1*    8    3
  8       2:0:3:2:1:0:8  -1*    3    1
 10       1:2:0:4:3:2:1  +1*   19    6
 11  p    2:3:1:5:4:3:2  -1    10    3
 12       0:0:2:0:5:4:3  -2*    7    2
 13  p n  1:1:3:1:6:5:4  +1*  649  180
 14       2:2:4:2:0:6:5  -1    15    4
 15       0:3:0:3:1:7:6  -2*    4    1
 17  p    2:1:2:5:3:1:8  +1*   33    8
 18       0:2:3:0:4:2:0  -1*   17    4
 19  p    1:3:4:1:5:3:1  +1*  170   39
 20       2:0:0:2:6:4:2  -1*    9    2
 21       0:1:1:3:0:5:3  +1    55   12
 22       1:2:2:4:1:6:4  +2*  197   42
 23  p    2:3:3:5:2:7:5  -1    24    5
```

Note that d is on the left and x and y are on the right. The letter p following a d value is used to indicate that the value of d is prime. Later, we will use q to indicate that d is probabilistically prime. In the line corresponding to $d = 13$ there is 13 followed by p which in turn is followed by n. We use n to indicate that the so-called *negative Pell equation* $x^2 - d y^2 = -1$ has a solution for the associated d, in this case $d = 13$. For a given non-square d the Pell equation always has solutions, but for the negative Pell equation, this need not be the case. In the above list only $d = 13$ has a solution to the negative Pell equation.

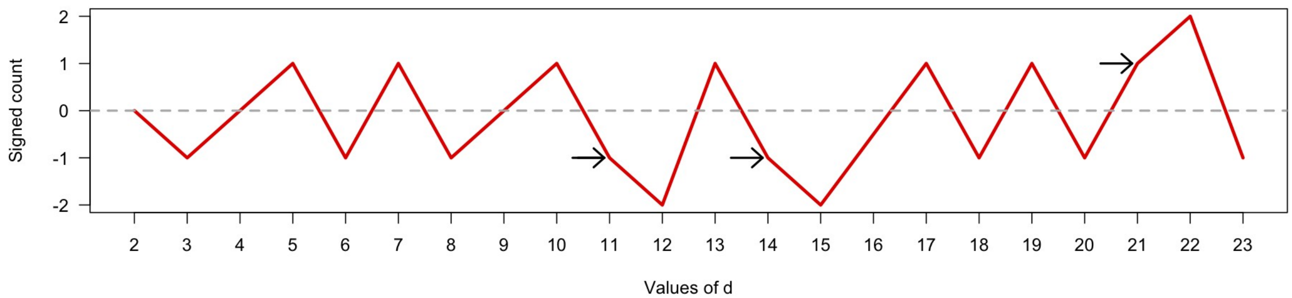
Although our pell program does not exhibit a solution to the negative Pell equation on the screen, the solution does get written to a file so that we can look at it if we want to. For example, from one of the program output files, we find that $d = 13$, $x = 18$, $y = 5$ is a solution to the negative Pell equation, which is easily checked.

In each line in the program output listed above there are seven integers separated by colons. The numbers with colons between them are the values of

d mod 3, 4, 5, 6, 7, 8, 9

For convenience we will call this the *gmod* [*g* for *general*] and we will see below that it provides useful information. In the above list a *signed count* follows each gmod. We want to explain how the signed count is computed. In the first line in the above list $x = 3$ and the signed count is zero, because this is the starting point. When we go from the first line to the second line, the value of x goes from 3 down to 2, and because the value went down the signed count is -1. [We will discuss the star following the -1 below]. What we are counting is the signed direction of the value of x from one line to the next, or more precisely from one d to the next d. When we go from the second line to the third line, the value of x goes from 2 up to 9. We have changed direction, so the signed count is +1. When we go from the third line to the fourth line, the value of x goes from 9 down to 5. We have changed direction again, so the signed count goes from +1 to -1. Note that when x goes from 19 to 10 to 7, decreasing in value twice, we get a signed count of -2.

We can think of the signed count as being defined for each d because the d's determine the x's and the direction of the value of the x's determines the signed count. Here is a graph of the signed count given in the above list:



The graph is inaccurate because d = 4, 9, 16 do not exist. Nonetheless the graph conveys the right ideas. Except for the three points on the graph marked with arrows, the signed count values occur at a local extremum. In the program output listed above, a star after a signed count means that the signed count is at a local extremum. We do not need the graph to see this. If we see, for example, +1* next to an x value, it means that values of x just above and below have less value. In the above list consider d = 12, x = 7, y = 2. The signed count is -2 and the star following indicates that the value of the x's just above and below 7 have higher value, namely 10 and 649.

The ups and downs of the x values are quite unpredictable. Although the direction of the x values changes often, there can be repeated change in the same direction, at least for a while. Here is some program output that shows the value of x going down 5 times in a row:

```
539       2:3:4:5:0:3:8  -1*             3970            171
540       0:0:0:0:1:4:0  +1            119071           5124
541  p n  1:1:1:1:2:5:1  +2*            #37:1          #36:0
542       2:2:2:2:3:6:2  -1           4293183         184408
543       0:3:3:3:4:7:3  -2            669337          28724
544       1:0:4:4:5:0:4  -3              2449            105
545       2:1:0:5:6:1:5  -4              1961             84
546       0:2:1:0:0:2:6  -5*              701             30
547  p    1:3:2:1:1:3:7  +1*  160177601264642  6848699678673
```

Looking at the first line, we cannot tell that the signed count value -1 is correct without looking at previous x values, but we will assume that the pell program gets it right. Looking at the x values in the next to the last column, we see that the direction of the value of x, either up or down, is captured by the corresponding signed count value. Note that there is a +2 because the value of x went up from 3970 to 119071 to #37:1. Although we do not know what this last value is, since it has 37 digits it is certainly larger than the value of x just above it.

Let us use our pell program to generate output where the d's consist of consecutive primes:

```
3701  p n  2:1:1:5:5:5:2  +1*                            #44:9                          #43:0
3709  p n  1:1:4:1:6:5:1  +3*  498938622490100272191791386249  8192542575130583730660585900
3719  p    2:3:4:5:2:7:2  -1                              3720                             61
3727  p    1:3:2:1:3:7:1  +4*                            #38:4                          #36:5
3733  p n  1:1:3:1:2:5:7  +1*                            #78:9                          #76:0
3739  p    1:3:4:1:1:3:4  +2*               209743543470762890               3430132277703579
3761  p n  2:1:1:5:2:1:8  +1*                            #57:9                          #55:0
```

We see that many primes have solutions to the negative Pell equation, but not all primes do. Also, we see that many primes have positive signed counts, but again not all do. Moreover, the signed count for a prime is often a local extremum, but not always. Note that what looks like the largest x in the in the second row in the list is actually one of the smaller x values. The largest x is #78:9. Finally, observe that many x's end in 9 and many y's end in 0.

Now let us use our pell program with the -i option, which implements the increasing x algorithm described above. Here is some output that gets written to a file:

```
        2  p    2:2:2:2:2:2:2   0           3          2
        5  p    2:1:0:5:5:5:5  +1*          9          4
       10       1:2:0:4:3:2:1  +1*         19          6
       13  p n  1:1:3:1:6:5:4  +1*        649        180
       29  p n  2:1:4:5:1:5:2  +2*       9801       1820
       46       1:2:1:4:4:6:1  +1*      24335       3588
       53  p n  2:1:3:5:4:5:8  +2*      66249       9100
       61  p n  1:1:1:1:5:5:7  +1*  1766319049  226153980
      109  p n  1:1:4:1:4:5:1  +2*      #15:9      #14:0
      181  p n  1:1:1:1:6:5:1  +1*      #19:1      #18:0
      277  p n  1:1:2:1:4:5:7  +2*      #21:9      #19:0
      397  p n  1:1:2:1:5:5:1  +2*      #21:9      #20:0
      409  p n  1:1:4:1:3:1:4  +1*      #23:9      #22:0
      421  p n  1:1:1:1:1:5:7  +1*      #34:9      #33:0
      541  p n  1:1:1:1:2:5:1  +2*      #37:1      #36:0
      661  p n  1:1:1:1:3:5:4  +1*      #38:9      #36:0
     1021  p n  1:1:1:1:6:5:4  +1*      #48:1      #46:0
     1069  p n  1:1:4:1:5:5:7  +2*      #48:9      #47:0
      . . . . . .
     8941  p n  1:1:1:1:2:5:4  +1*     #202:1     #200:0
     9949  p n  1:1:4:1:2:5:4  +1*     #212:9     #210:0
    12541  p n  1:1:1:1:4:5:4  +1*     #236:1     #234:0
    13381  p n  1:1:1:1:4:5:7  +2*     #256:9     #254:0
    16069  p n  1:1:4:1:4:5:4  +1*     #261:9     #259:0
     . . . . . .
    82021  p n  1:1:1:1:2:5:4  +1*     #682:9     #680:0
    92821  p n  1:1:1:1:1:5:4  +1*     #724:9     #722:0
   107101  p n  1:1:1:1:1:5:1  +1*     #726:1     #723:0
   115021  p n  1:1:1:1:4:5:1  +1*     #771:1     #768:0
   125101  p n  1:1:1:1:4:5:1  +2*     #818:1     #815:0
    . . . . . .
   799621  p n  1:1:1:1:4:5:7  +1*    #2398:1    #2395:0
   952429  p n  1:1:4:1:2:5:4  +2*    #2475:9    #2472:0
  1026061  q n  1:1:1:1:1:5:7  +2*    #2555:9    #2552:0
  1027261  q n  1:1:1:1:4:5:1  +1*    #2638:9    #2635:0
  1047589  q n  1:1:4:1:4:5:7  +1*    #2651:1    #2648:0
   . . . . . .
  9670621  q n  1:1:1:1:2:5:4  +1*    #8682:1    #8678:0
  9747061  q n  1:1:1:1:2:5:7  +1*    #8982:9    #8979:0
 10675261  q n  1:1:1:1:2:5:1  +1*    #9135:9    #9132:0
 10774261  q n  1:1:1:1:1:5:1  +1*    #9344:1    #9341:0
 11486029  q n  1:1:4:1:2:5:4  +2*    #9689:9    #9685:0
  . . . . . .
 92919061  q n  1:1:1:1:4:5:1  +1*   #29564:9   #29560:0
 99890389  q n  1:1:4:1:4:5:1  +1*   #29739:9   #29735:0
100460221  q n  1:1:1:1:1:5:7  +1*   #29983:1   #29979:0
100685341  q n  1:1:1:1:1:5:1  +1*   #30410:9   #30406:0
101247589  q n  1:1:4:1:2:5:1  +2*   #30446:1   #30442:0
. . . . . .
235311301  q n  1:1:1:1:1:5:1  +1*   #47427:9   #47423:0
236690749  q n  1:1:4:1:1:5:1  +1*   #48517:9   #48513:0
251342701  q n  1:1:1:1:1:5:7  +1*   #48972:1   #48968:0
262831501  q n  1:1:1:1:2:5:1  +2*   #49317:1   #49313:0
266208541  q n  1:1:1:1:4:5:7  +1*   #49648:9   #49644:0
. . . . . .
346477069  q n  1:1:4:1:1:5:1  +1*   #58388:1   #58384:0
362047981  q n  1:1:1:1:1:5:4  +1*   #59136:9   #59132:0
380960869  q n  1:1:4:1:2:5:4  +3*   #59957:1   #59953:0
382399021  q n  1:1:1:1:4:5:1  +1*   #60540:1   #60536:0
395479309  q n  1:1:4:1:1:5:4  +1*   #62416:9   #62412:0
```

The Pell prime conjectures

Based on the computational evidence presented above, we make the following conjectures:

- Starting at d = 53 and using the increasing x algorithm described above, we generate an infinite increasing sequence of integers, the d's, that we conjecture are primes, the ***Pell primes***:

  53, 61, 109, 181, 277, 397, 409, 421, 541, 661, 1021, 1069, 1381, 1549, 1621, 2389, 3061, 3469, ...

- In addition to being prime, each d has a solution to the negative Pell equation, and the signed count is positive and occurs at a local maximum.
- For each d the corresponding solution x, y to the Pell equation has the property that x ends in 1 or 9 and y ends in 0.

Moreover, each d satisfies

- d = 1 mod 3 except for d = 53
- d = 1 mod 4
- d = 1 or 4 mod 5 except for d = 53, 277, 397
- d = 1 mod 6 except for d = 53
- d = 1, 2, or 4 mod 7 except for d = 61, 181, 397, 409, 661, 1021, 1069
- d = 5 mod 8 except for d = 409 and 24049
- d = 1, 4, or 7 mod 9 except for d = 53
- d = 1 or 9 mod 10 except for d = 53, 277, 397
- d = 10 mod 11 only for d = 109 [not shown in the above output]

The C++ program

The C++ program, pell, is being developed on a Dell Dimension XPS workstation that is now more than eight years old. The CPU is a P4 Intel chip running at 3.6 ghz, and there is 2 gigabytes of main memory. The linux operating system on the computer is Fedora 18, which is the most recent distribution of Fedora.

The program consists of some 6,000 lines of code. It is compiled with the -std=c++11 option using version 4.7.2 of the GNU g++ compiler. The program uses GMP, the GNU multiprecision arithmetic library, along with the associated header file gmpxx.h that provides overloaded arithmetic operators. The algorithm that is used to solve the Pell equation follows precisely what is given in [1], and each solution is checked to see that it does, in fact, solve the Pell equation. The GMP package provides a means to check whether an integer is composite or prime or probabilistically prime. An integer reported as probabilistically prime has less than 1 chance 2^{50} of being composite.

As of this writing, an older version of the program has been running for approximately 44 days on a Toshiba Qosmio X505 laptop computer that is three years old. The chip in the laptop is an Intel Core i7 running at 2.93 ghz, and the laptop has 6 gigabytes of main memory. The linux operating system is also Fedora 18.

The program that has been running for 44 days is using the -i increasing option to find solutions to the Pell equation with increasing x values as described above. So far, 304 solutions have been found. The most recent d value is 395,479,309 and the corresponding x value has 62,416 digits. In the last 6 days only 3 new solutions have been found.

The computational evidence for our conjectures looks good, but the evidence is very skimpy. In general, solving the Pell equation is much harder than factoring, which itself is known to be very difficult for large integers. Nonetheless, factoring a 12-digit integer takes only a few milliseconds. By comparison, solving the Pell equation when d has 12 digits can be much more difficult.

To show this, we construct via another program the 12-digit prime 753,352,993,489 and use the pell program to write the following on the screen:

```
753352993487       2:3:2:5:3:7:2  -1*     #7775:1     #7769:0
753352993488       0:0:3:0:4:0:3  +1     #14237:7    #14231:9
753352993489  q n  1:1:4:1:5:1:4  +2*  #1199818:9  #1199812:0
753352993490       2:2:0:2:6:2:5  -1*    #26592:9    #26586:0
753352993491       0:3:1:3:0:3:6  +1     #50076:1    #50070:0
```

Although it is the middle d that is of interest, we computed in addition two d's above and two d's below to see the middle d in context. We chose the d in the middle to be a prime for this exercise, because in general, but not always, primes are more difficult to solve than an adjacent composite d. In any case, to get the above output required more than 29 minutes of computation time. Note that the x and y corresponding to the d of interest have almost 1.2 million digits. The adjacent d's take very little time to solve in comparison to the d of interest. For example, solving the Pell equation with the first d in the list takes just over a minute.

Generating similar output when d is prime and has 13 digits can take more than 4.6 hours, but that is not always the case. For some 13-digit primes the computation time is much less. For example, when d has the value 8,735,653,484,527 the computation time is approximately 15 minutes.

The Pell primes have an interesting kind of fractal quality that we will report on later.

References

1. http://mathworld.wolfram.com/PellEquation.html
2. http://en.wikipedia.org/wiki/Pell's_equation