Sieving for pseudosquares and pseudocubes in parallel using doubly-focused enumeration and wheel datastructures*  

Jonathan P. Sorenson  
Butler University, Indianapolis IN 46208, USA,  
sorenson@butler.edu,  
http://www.butler.edu/~sorenson  

Abstract. We extend the known tables of pseudosquares and pseudocubes, discuss the implications of these new data on the conjectured distribution of pseudosquares and pseudocubes, and present the details of the algorithm used to do this work. Our algorithm is based on the space-saving wheel data structure combined with doubly-focused enumeration, run in parallel on a cluster supercomputer.

1 Introduction

It is well-known that testing for primality can be done in polynomial time [13]. However, the fastest known deterministic algorithms are conjectured to be the pseudosquares prime test of Lukes, Patterson, and Williams [6], and its generalization, the pseudocube prime test of Berrizbeitia, Müller, and Willimas [4], both of which run in roughly cubic time, if a sufficiently large pseudosquare or pseudocube is available. In particular, the pseudosquares prime test is very useful in the context of finding all primes in an interval [9], where sieving can be used in place of trial division. This, then, motivates the search for larger and larger pseudosquares and pseudocubes, and attempts to predict their distribution. See, for example, Wooding and Williams [12] and also [7,11,8,2,10].

In this paper, we present extensions to the known tables of pseudosquares and pseudocubes in §2. We discuss the implications of this new data on the conjectured distribution of pseudosquares and pseudocubes in §3 and give a minor refinement of the current conjectures. Then we describe our parallel algorithm, based on Bernstein’s doubly-focused enumeration [2], which is used in a way similar, but not identical to the work of Wooding and Williams [12], combined with the space-saving wheel data structure presented in [9, §4.1]. We then suggest ideas for future work in §5.

* Supported by a grant from the Holcomb Awards Committee, and computing resources provided by the Frank Levinson Supercomputing Center at Butler University.
2 Computational Results

Let \((x/y)\) denote the Legendre symbol \([5]\). For an odd prime \(p\), let \(L_{p,2}\), the *pseudosquare* for \(p\), be the smallest positive integer such that

1. \(L_{p,2} \equiv 1 \pmod{8}\),
2. \((L_{p,2}/q) = 1\) for every odd prime \(q \leq p\), and
3. \(L_{p,2}\) is not a perfect square.

In other words, \(L_{p,2}\) is a square modulo all primes up to \(p\), but is not a square. We found the following new pseudosquares:

| \(p\) | \(L_{p,2}\) |
|------|----------|
| 367  | 34429 74705 474600 46489 |
| 373  | 25223 08059 75035 19329 |
| 379  | \(> 10^{25}\) |

The two pseudosquares listed were found in 2008 in a computation that went up to \(5 \times 10^{24}\), taking roughly 3 months wall time. The final computation leading to the lower bound of \(10^{25}\) ran for about 6 months, in two 3-month pieces, the second of which finished on January 1st, 2010.

Wooding and Williams \([12]\) had found a lower bound of \(L_{367,2} > 120120 \times 2^{64} \approx 2.216 \times 10^{24}\). (Note: a complete table of pseudosquares, current as of this writing, is available at [http://cr.yp.to/focus.html](http://cr.yp.to/focus.html) care of Dan Bernstein).

Similarly, for an odd prime \(p\), let \(L_{p,3}\), the *pseudocube* for \(p\), be the smallest positive integer such that

1. \(L_{p,3} \equiv \pm 1 \pmod{9}\),
2. \(L_{p,3}^{(q-1)/3} \equiv 1 \pmod{q}\) for every prime \(q \leq p\), \(q \equiv 1 \pmod{3}\),
3. \(\gcd(L_{p,3}, q) = 1\) for every prime \(q \leq p\), and
4. \(L_{p,3}\) is not a perfect cube.

We found the following new pseudocubes (only listed for \(p \equiv 1 \pmod{3}\)):

| \(p\) | \(L_{p,3}\) |
|------|----------|
| 499  | 21674 16551 89317 |
| 523,541 | 21674 16551 89317 |
| 547  | 15947 |
| 571,577 | 27899 |
| 601,607 | 90677 |
| 613  | 90677 |
| 619  | \(> 10^{27}\) |

These pseudocubes were found in about 6 months of total wall time in 2009. Wooding and Williams \([12]\) had found a lower bound of \(L_{499,3} > 1.45152 \times 10^{22}\). For a complete list of known pseudocubes, see \([12,10]\).
3 The Distribution of Pseudosquares and Pseudocubes

Let $p_i$ denote the $i$th prime, and $q_i$ denote the $i$th prime such that $q_i \equiv 1 \pmod{3}$. In [6] it was conjectured that, for a constant $c_2 > 0$, we have

$$L_{p_n,2} \approx c_2 2^n \log p_n.$$  \hspace{1cm} (1)

Using similar methods, in [4] it was conjectured that, for a constant $c_3 > 0$, we have

$$L_{q_n,3} \approx c_3 3^n (\log q_n)^2.$$  \hspace{1cm} (2)

In a desire to test the accuracy of these conjectures, for integers $n > 0$ let us define

$$c_2(n) := \frac{L_{p_n,2}}{2^n \log p_n}, \hspace{1cm} (3)$$

$$c_3(n) := \frac{L_{q_n,3}}{3^n (\log q_n)^2}. \hspace{1cm} (4)$$

We calculated $c_2(n)$ and $c_3(n)$ from known pseudosquares and pseudocubes. We present these computations in Table 1 for pseudosquares, and in Table 2 for pseudocubes, below.

From Table 1 we readily see that $c_2(n)$ appears to be bounded between roughly 5 and 162, with an average value near 45. There is no clear trend toward zero or infinity. Due to the common occurrence of values of $n$ where $L_{p_n,2} = L_{p_{n+1},2}$ (for example, $n = 56$), it should also be clear $c_2(n)$ does not have a limit.

Similarly for the pseudocubes, in Table 2 we see that $0.05 < c_3(n) < 6.5$ for $10 \leq n \leq 53$, with an average value of roughly 1.22. And again, there is no clear trend toward zero or infinity, nor can there be a limit for $c_3(n)$.

This leads us to the following refinements, if you will, of the conjectures (1),(2) above.

**Conjecture.** For the pseudosquares, we conjecture that

$$\lim\inf_{n \to \infty} \frac{L_{p_n,2}}{2^n \log p_n} > 0,$$  \hspace{1cm} (5)

$$\lim\sup_{n \to \infty} \frac{L_{p_n,2}}{2^n \log p_n} < \infty.$$  \hspace{1cm} (6)

Similarly, for the pseudocubes, we conjecture that

$$\lim\inf_{n \to \infty} \frac{L_{q_n,3}}{3^n (\log q_n)^2} > 0,$$  \hspace{1cm} (7)

$$\lim\sup_{n \to \infty} \frac{L_{q_n,3}}{3^n (\log q_n)^2} < \infty.$$  \hspace{1cm} (8)

Our data also has implications on the relative efficiency of primality testing. In particular, several researchers have pointed out that if conjectures (1),(2) are
| \(n\) | \(p_n\) | \(L_{p_n,2}\) | \(c_2(n)\) | \(n\) | \(p_n\) | \(L_{p_n,2}\) | \(c_2(n)\) |
|-----|--------|----------|--------|-----|--------|----------|--------|
| 2   | 3      | 16.61    | 39     | 167 | 112434732901969 | 39.96  |
| 3   | 5      | 18.72    | 40     | 173 | 178936222537081 | 31.58  |
| 4   | 7      | 32.41    | 41     | 179 | 178936222537081 | 15.69  |
| 5   | 11     | 34.42    | 42     | 181 | 696161110020049 | 30.45  |
| 6   | 13     | 49.28    | 43     | 191 | 696161110020049 | 15.07  |
| 7   | 17     | 49.64    | 44     | 193 | 2854909648103881 | 30.84  |
| 8   | 19     | 71.48    | 45     | 197 | 6450045516630769 | 34.70  |
| 9   | 23     | 54.49    | 46     | 199 | 6450045516630769 | 17.32  |
| 10  | 29     | 33.95    | 47     | 211 | 11641399247947921 | 15.46  |
| 11  | 31     | 73.34    | 48     | 223 | 11641399247947921 | 7.65   |
| 12  | 37     | 73.24    | 49     | 227 | 190621428905186449 | 62.42  |
| 13  | 41     | 105.41   | 50     | 229 | 196640148121928601 | 32.14  |
| 14  | 43     | 61.97    | 51     | 233 | 71262435095093521 | 58.06  |
| 15  | 47     | 73.38    | 52     | 239 | 177385579187750321 | 71.92  |
| 16  | 53     | 84.55    | 53     | 241 | 2327687064124474441 | 47.12  |
| 17  | 59     | 90.70    | 54     | 251 | 6384991873059836689 | 64.15  |
| 18  | 61     | 44.98    | 55     | 257 | 801920466130519761 | 40.11  |
| 19  | 67     | 79.49    | 56     | 263 | 101981058204267869 | 25.40  |
| 20  | 71     | 95.70    | 57     | 269 | 101981058204267869 | 12.65  |
| 21  | 73     | 47.54    | 58     | 271 | 101981058204267869 | 6.32   |
| 22  | 79     | 49.04    | 59     | 277 | 6984288832000186969 | 21.54  |
| 23  | 83     | 75.69    | 60     | 281 | 20893636579904975961 | 32.14  |
| 24  | 89     | 37.25    | 61     | 283 | 5335266333982803681 | 40.99  |
| 25  | 97     | 18.28    | 62     | 293 | 9366407926671697089 | 35.76  |
| 26  | 101    | 33.29    | 63     | 307 | 9366407926671697089 | 17.73  |
| 27  | 103    | 37.96    | 64     | 311 | 214220286037026991629 | 20.23  |
| 28  | 107    | 67.89    | 65     | 313 | 214220286037026991629 | 10.10  |
| 29  | 109    | 33.81    | 66     | 317 | 214220286037026991629 | 5.04   |
| 30  | 113    | 38.67    | 67     | 331 | 13649154491585298803281 | 15.94  |
| 31  | 127    | 18.87    | 68     | 337 | 345948580167012778801 | 20.14  |
| 32  | 131    | 137.15   | 69     | 347 | 99492945930479213334049 | 28.81  |
| 33  | 137    | 67.95    | 70     | 349 | 99492945930479213334049 | 14.39  |
| 34  | 139    | 33.88    | 71     | 353 | 29536318740900310880401 | 21.32  |
| 35  | 149    | 152.68   | 72     | 359 | 29536318740900310880401 | 10.63  |
| 36  | 151    | 76.14    | 73     | 367 | 3655334242970546004689 | 65.54  |
| 37  | 157    | 161.79   | 74     | 373 | 4235025223080597503519329 | 37.86  |
| 38  | 163    | 80.30    |         |     |                    |       |
Table 2. Values of $c_3(n)$ based on known pseudocubes.

| $n$ | $q_n$ | $L_{q_n,3}$ | $c_3(n)$ | $n$ | $q_n$ | $L_{q_n,3}$ | $c_3(n)$ |
|-----|-------|-------------|----------|-----|-------|-------------|----------|
| 10  | 79    | 7255857     | 6.42     | 32  | 337   | 75017625272879381 | 1.2      |
| 11  | 97    | 8721539     | 2.35     | 33  | 349   | 75017625272879381 | 0.394    |
| 12  | 103   | 8721539     | 0.764    | 34  | 367   | 996438651365898469 | 1.71     |
| 13  | 109   | 91246121    | 2.6      | 35  | 373   | 2152984914389968651 | 1.23     |
| 14  | 127   | 91246121    | 0.813    | 36  | 379   | 12403284862819956587 | 2.34     |
| 15  | 139   | 98018803    | 0.281    | 37  | 397   | 3760527410547922861 | 2.33     |
| 16  | 151   | 1612383137  | 1.49     | 38  | 409   | 3760527410547922861 | 0.77     |
| 17  | 157   | 1612383137  | 0.488    | 39  | 421   | 3760527410547922861 | 0.254    |
| 18  | 163   | 7991083927  | 0.795    | 40  | 433   | 205830039006337114403 | 0.459    |
| 19  | 181   | 7991083927  | 0.254    | 41  | 439   | 1845193818928603436441 | 1.37     |
| 20  | 193   | 7991083927  | 0.0827   | 42  | 457   | 7854338425385225902393 | 1.91     |
| 21  | 199   | 20365764119 | 0.0695   | 43  | 463   | 12904554928068268848739 | 1.04     |
| 22  | 211   | 2515598768717 | 2.8 | 44  | 487   | 133848095485212757303 | 0.355    |
| 23  | 223   | 6440555721601 | 2.34 | 45  | 499   | 60125695216741655189317 | 0.527    |
| 24  | 229   | 29135874901141 | 3.49 | 46  | 523   | 116614853914870278915947 | 0.336    |
| 25  | 241   | 29135874901141 | 1.14 | 47  | 541   | 116614853914870278915947 | 0.111    |
| 26  | 271   | 29135874901141 | 0.365 | 48  | 547   | 4139150561509947885227899 | 1.31     |
| 27  | 277   | 406540676672677 | 1.69 | 49  | 571   | 16248573199879956914339717 | 1.69     |
| 28  | 283   | 406540676672677 | 0.558 | 50  | 577   | 16248573199879956914339717 | 0.56     |
| 29  | 307   | 406540676672677 | 0.181 | 51  | 601   | 24191374719361484275890677 | 0.274    |
| 30  | 313   | 406540676672677 | 0.0598 | 52  | 607   | 24191374719361484275890677 | 0.0912   |
| 31  | 331   | 75017625272879381 | 3.61 | 53  | 613   | 674441580981249129037406633 | 0.845    |
true, then the running time of the pseudocube prime test, which depends on the value of $L_{q_n,3}^{2/3}$, should eventually outperform the pseudosquare prime test, whose running time depends on $L_{p_n,2}$. In particular, one infers from conjectures (1) and (2) that

$$\frac{L_{q_n,3}^{2/3}}{L_{p_n,2}} \gg \left(\frac{3^{2/3}}{2}\right)^n > 1$$

(9)

for sufficiently large $n$ (see [12, §9.1]). This inference follows from our refined conjectures as well.

We have our first specific value of $n$ to support (9), namely with $n = 48$, where $L_{q_n,3}^{2/3} \approx 2.214 \cdot L_{p_n,2}$. However, given that $c_2(n)$ averages about 45, and $c_3(n)$ averages just over 1.2, we would reasonably expect (9) to largely be true only for $n$ larger than about 75, under the assumption these averages are maintained. To test this, more pseudosquares and, in particular, more pseudocubes are needed.

4 Algorithm Details

We begin with a review of doubly-focused enumeration, explain how we employ parallelism, and how the space-saving wheel data structure is utilized. We also discuss the details of our implementation, including the hardware platform and software used.

4.1 Doubly-Focused Enumeration

The main idea is that every integer $x$, with $0 \leq x \leq H$, can be written in the form

$$x = t_p M_n - t_n M_p$$

(10)

where

$$\gcd(M_p, M_n) = 1, \quad 0 \leq t_p \leq \frac{H + M_p M_n}{M_n}, \quad \text{and} \quad 0 \leq t_n < M_n.$$  

(11)

(See [2] or [12, Lemma 1].) This is an explicit version of the Chinese Remainder Theorem.

To find pseudosquares, we set $M_n$ and $M_p$ to be products of small odd primes and 8, choose $t_p$ to be square modulo $M_p$, and $-t_n$ to be square modulo $M_n$. To be precise, in our implementation we set

$$M_p = 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 29 \cdot 31 \cdot 37 \cdot 41 \cdot 43 \cdot 53 \cdot 89 = 2057046173382917717 \quad \text{and}$$

$$M_n = 8 \cdot 3 \cdot 5 \cdot 47 \cdot 59 \cdot 61 \cdot 67 \cdot 71 \cdot 73 \cdot 79 \cdot 83 \cdot 97 = 4483259527721526840.$$

Note that both $M_p, M_n < 2^{64}$, allowing us to work in 64-bit machine arithmetic.
To find pseudocubes, the same idea applies, only note that if \(-t_n\) is a cube modulo \(M_n\), so is \(t_n\). We used only 2, 9 and primes congruent to 1 (mod 3) for better filter rates:

\[
M_p = 2 \cdot 7 \cdot 13 \cdot 31 \cdot 43 \cdot 73 \cdot 79 \cdot 127 \cdot 139 \cdot 157 \cdot 181 = 701\,856\,356\,110\,394\,022\quad \text{and} \\
M_n = 9 \cdot 19 \cdot 37 \cdot 61 \cdot 67 \cdot 97 \cdot 103 \cdot 109 \cdot 151 \cdot 163 = 693\,110\,504\,329\,192\,503
\]

### 4.2 Parallelism and Main Loop

Each processor core was assigned an interval of \(t_p\) values to process by giving it values of \(H^-\) and \(H^+\).

For finding pseudosquares, \(H^+ - H^- \approx M_n \cdot 4.76 \times 10^{11}\). For finding pseudocubes, \(H^+ - H^- \approx M_n \cdot 4.99 \times 10^{12}\).

Parallelism was achieved by having different processors working on different intervals simultaneously. Once all processors had finished their current intervals, the work was saved to disk (allowing restarts as needed) and new intervals were assigned.

To process an interval, each processor core did the following:

1. Using the wheel datastructure, generate all square or cube values of \(t_p\) with \(H^- \leq t_p M_n \leq H^+\), and store these in an array \(A[]\).
2. The wheel datastructure does not generate the \(t_p\) values in order, so sort \(A[]\) in memory using quicksort. Note that \(H^-\) and \(H^+\) are chosen close enough together so that this array held no more than 40 million integers, using at most 320 megabytes of RAM per processor core.
3. Using the first and last entries in \(A[]\), compute a range of valid \(t_n\) values to process, and then use a wheel datastructure to generate all \(t_n\) values in that range such that \(-t_n\) is square modulo \(M_n\) for pseudosquares, or \(t_n\) is a cube modulo \(M_n\) for pseudocubes.

We use an outer loop over \(t_n\) values in the order enumerated by the wheel data structure for \(M_n\), and an inner loop over consecutive \(t_p\) values drawn from \(A[]\).

4. For each \(t_n\) generated, we normalize sieve tables for the next 4 primes (101, 103, 107, 109 for pseudosquares, and 193, 199, 211, 223 for pseudocubes) to allow for constant-time table lookup to see if an \(x\)-value (see below) is a square/cube modulo these primes, indexed by \(t_p\) value.

The number of primes to use for this depends on how many \(t_p\) values will be processed for each \(t_n\) – in our case, it was several hundred on average, so this step improves performance. If it were fewer, say 50, then normalizing the sieve tables would require more work than is saved by having constant-time lookup.

5. For each \(t_n\) generated, using binary search on \(A[]\) to find all the \(t_p\) values it can match with, generate an \(x = t_p M_n - t_n M_p\) within our global search
range. (For example, in our last run for pseudosquares, we searched for \(x\) values between \(7.5 \times 10^{24}\) and \(10^{25}\).)

Note: at this point we do not actually compute the value of \(x\).

6. Lookup each \(t_p\) value in the normalized tables mentioned above. If it fails any of the 4 sieve tests, move on to the next \(t_p\) value. For pseudosquares, a \(t_p\) values passes these tests with probability roughly \((1/2)^4 = 1/16\), and for pseudocubes, roughly \((1/3)^4 = 1/81\).

Note that this step is the running time bottleneck of the algorithm.

7. The next batch of primes \(q\) have precomputed sieve tables that are not normalized, but we precompute \(M_p\) and \(M_n\) modulo each \(q\) so the we can compute \(x \mod q\) without exceeding 64-bit arithmetic. Continue only if our \(t_p\) value passes all these sieve tests as well. The expected number of primes \(q\) used in this step is constant.

8. Finally, compute \(x\) using 128-bit hardware arithmetic, and see if it is a perfect square or perfect cube. If it passes this test, append \(x\) to the output file for this processor core.

We had two wheel datastructures, one each for \(M_p\) and \(M_n\). For details on how this datastructure works, see \cite{9}. We leave the details for how to modify the datastructure to handle cubes in place of squares to the reader.

4.3 Implementation Details

To compute the tables presented in \cite{2}, we used Butler University’s cluster supercomputer, BigDawg, which has 24 compute nodes, each of which has four AMD Opteron 8354 quad-core CPUs at 2.2GHz with 512KB cache, for a total of 384 compute cores. As might be expected, we did not have sole access to this machine for over a year, so the code was designed, and ran, using anywhere from 10 to 24 nodes, or from 160 to 384 cores, depending on the needs of other users. This flexibility is one advantage of our parallelization method – by \(t_p\) intervals. In \cite{12}, they parallelized over residue classes, which restricts the CPU count to a fixed number (180 in their case).

BigDawg runs a Linux kernel on its head node and compute nodes, and the code was written in C++ using the gnu compiler (version 4.1.2) with MPI. It has both 10GB ethernet and Infiniband interconnect, but inter-processor communication was not a bottleneck for our programs.

We tested our code by first finding known pseudosquares (all but the highest few) and known pseudocubes, in the process verifying previous results.

5 Future Work

We plan to port our code to work with 8 NVidia GPUs recently added to Butler’s supercomputer, giving it roughly 2-3 times the raw computing power. This will require a major restructuring of the code, and the removal of recursion in the wheel datastructure.
References

1. Agrawal, M., Kayal, N., Saxena, N.: PRIMES is in P. Ann. of Math. (2) 160(2), 781–793 (2004), \url{http://dx.doi.org/10.4007/annals.2004.160.781}

2. Bernstein, D.J.: Doubly focused enumeration of locally square polynomial values. In: High primes and misdemeanours: lectures in honour of the 60th birthday of Hugh Cowie Williams, Fields Inst. Commun., vol. 41, pp. 69–76. Amer. Math. Soc., Providence, RI (2004)

3. Bernstein, D.J.: Proving primality in essentially quartic random time. Math. Comp. 76(257), 389–403 (electronic) (2007), \url{http://dx.doi.org/10.1090/S0025-5718-06-01786-8}

4. Berrizbeitia, P., Müller, S., Williams, H.C.: Pseudocubes and primality testing. In: Algorithmic number theory, Lecture Notes in Comput. Sci., vol. 3076, pp. 102–116. Springer, Berlin (2004)

5. Hardy, G.H., Wright, E.M.: An Introduction to the Theory of Numbers. Oxford University Press, 5th edn. (1979)

6. Lukes, R.F., Patterson, C.D., Williams, H.C.: Some results on pseudosquares. Math. Comp. 65(213), 361–372, S25–S27 (1996)

7. Pomerance, C., Shparlinski, I.E.: On pseudosquares and pseudopowers. In: Combinatorial number theory, pp. 171–184. Walter de Gruyter, Berlin (2009)

8. Schinzel, A.: On pseudosquares. New Trends in Prob. and Stat. 4, 213–220 (1997)

9. Sorenson, J.P.: The pseudosquares prime sieve. In: Hess, F., Pauli, S., Pohst, M. (eds.) Proceedings of the 7th International Symposium on Algorithmic Number Theory (ANTS-VII). pp. 193–207. Springer, Berlin, Germany (July 2006), LNCS 4076, ISBN 3-540-36075-1

10. Stephens, A.J., Williams, H.C.: An open architecture number sieve. In: Number theory and cryptography (Sydney, 1989), London Math. Soc. Lecture Note Ser., vol. 154, pp. 38–75. Cambridge Univ. Press, Cambridge (1990)

11. Williams, H.C.: Édouard Lucas and primality testing. Canadian Mathematical Society Series of Monographs and Advanced Texts, 22, John Wiley & Sons Inc., New York (1998), a Wiley-Interscience Publication

12. Wooding, K., Williams, H.C.: Doubly-focused enumeration of pseudosquares and pseudocubes. In: Proceedings of the 7th International Algorithmic Number Theory Symposium (ANTS VII). Berlin, Germany (2006)