Sums of Two Squares Visualized

Ishan Banerjee and Amites Sarkar

September 6, 2023

Abstract

In 1972, John Brillhart described an algorithm for expressing a prime $p \equiv 1 \pmod 4$ as the sum of two squares. Brillhart's algorithm, which is based on the Euclidean algorithm, is simplicity itself. However, Brillhart's proof of his algorithm's correctness uses several previous results, and subsequent simplifications of his argument still retain something of an air of mystery. We provide a geometric interpretation of Brillhart's algorithm, which not only proves that it works, but also sheds light on a surprising palindromic property proved by Perron, and used by Brillhart in his proof.

1 Introduction.

On Christmas Day 1640, Pierre de Fermat stated his famous "two-square" theorem in a letter to Marin Mersenne. This theorem, which was first observed by Girard in 1625, and first proved by Euler in 1749, states (in Gauss's notation) that an odd prime p is the sum of two squares if (and only if) $p \equiv 1 \pmod 4$. Among the many admirers of the theorem was G.H. Hardy, who wrote in *A Mathematician's Apology* that the theorem is one of the finest of arithmetic, and also that "there is no proof within the comprehension of anybody but a fairly expert mathematician" [6].

Hardy's famous textbook with E. M. Wright, *An Introduction to the Theory of Numbers*, contains a neat geometrical proof of the two-square theorem, due to J.H. Grace [5]. Grace's proof can be summarized as follows. First, we solve the congruence $u^2 + 1 \equiv 0 \pmod p$. (No major expertise required here: writing $p = 4k+1$, a short argument using Wilson's theorem shows that $u = (2k)!$ is a solution. The proof is due to Lagrange.) Next consider the integer lattice L given by

$$L = \{(x, y) \in \mathbb{Z}^2 : x \equiv uy \pmod p\}. \tag{1}$$

L is a lattice containing a proportion $1/p$ of integer points; more precisely, the area of any fundamental parallelogram of the lattice is p. But L is also a *square lattice*, since if $P = (g, h)$ is a non-zero point of L closest to the origin O, then $g \equiv uh \pmod p$, so that $ug \equiv -h \pmod p$,

and $(-h, g) \in L$ too. These two facts together imply that the area $g^2 + h^2$ of the square on side OP is p, proving the theorem.

There are also many proofs of Fermat's theorem that do not use the number u defined above. For instance, in 1971, Heath-Brown [9] found a proof using involutions on finite sets, which he published 13 years later. Then, in 1990, Zagier [13] condensed Heath-Brown's argument into a single (long) sentence. Very recently, Dolan [3] obtained an extremely short and elementary proof, based on computing the parity of the number of positive integer solutions (x, y, z) to

$$p = (x + y + z)^2 - 4xz = (x + y - z)^2 + 4yz$$

in two different ways. Indeed, the equation $p = (x + y - z)^2 + 4yz$ shows first that the number of such solutions is finite, and then that there are an even number of solutions with $y \neq z$. But the equation $p = (x + y + z)^2 - 4xz$ shows that, for a prime $p \equiv 1 \pmod 4$, the total number of solutions is odd. Therefore, there must be a solution with $y = z$, so that $p = x^2 + 4y^2$. These proofs are all non-constructive.

All well and good. But is there a fast algorithm for finding g and h such that $p = g^2 + h^2$? Indeed there is. In 1972, John Brillhart [2], building on earlier work of Legendre, Serret, and Hermite, proposed the following method, starting with p and the number u defined above. If $p = u^2 + 1$, stop. Otherwise, simply apply the Euclidean algorithm to p and u, and stop when the first remainder r satisfies $r < \sqrt{p}$. If the *next* remainder is r', then $(g, h) = (r, r')$. (The fly in the ointment is that we first need to find u. Lagrange's formula $u = (2k)!$ is computationally inefficient, but Shiu [10] provides a non-deterministic algorithm with a short expected running time: find a quadratic non-residue s modulo p, and compute $u \equiv s^k \pmod p$ using the fast exponentiation algorithm based on repeated squaring. Euler's criterion now shows that $u^2 \equiv s^{2k} \equiv -1 \pmod p$. Shiu also gives an explicit formula, involving factorials and due to Gauss, for g and h themselves. Alas, Gauss's formula, like Lagrange's formula for u, is computationally inefficient.)

Here's an example. Let $p = 73$, and solve $u^2 + 1 \equiv 0 \pmod p$ to get $u = \pm 27$. Taking $u = 27$ (although $u = 46 \equiv -27 \pmod{73}$ also works), the Euclidean algorithm now yields:

$$\begin{aligned} 73 &= 2 \cdot 27 + 19 \\ 27 &= 1 \cdot 19 + \mathbf{8} \\ 19 &= 2 \cdot 8 + \mathbf{3} \end{aligned}$$

and, lo and behold, $8^2 + 3^2 = 73$.

Magic? Maybe. Brillhart's proof of his algorithm's correctness used the palindromic property, proved by Perron, of the continued fraction expansion of p/u. Wagon [11] simplified the proof,

employing Euler's formula $a = f(q_1, q_2, \ldots, q_n)$, where the q_i are the quotients in the Euclidean algorithm applied to compute $\gcd(a, b)$, and f is a certain sum of products. Shiu [10] presents a proof using *continuants*, quantities introduced by Euler in his investigation of continued fractions. These proofs, while natural, are all algebraic. By contrast, a closer look at Grace's proof reveals that Brillhart's algorithm has a secret geometric twin: *Lagrange's algorithm.*

2 Lattice reduction.

Two integer vectors in $\mathbb{R}^2$, such as $\mathbf{b_1} = (3,1)^\mathsf{T}$ and $\mathbf{b_2} = (5,3)^\mathsf{T}$, generate a lattice L in the plane, consisting of all vectors of the form $m\mathbf{b_1} + n\mathbf{b_2}$, where m and n are integers. Noting that $||\mathbf{b_1}|| = \sqrt{10}$ and $||\mathbf{b_2}|| = \sqrt{34}$, we might wonder if there is a better way of generating the same lattice. What do we mean by better? One answer is that we should try to minimize $||\mathbf{b_1}||$ and $||\mathbf{b_2}||$, subject to the condition that $\mathbf{b_1}$ and $\mathbf{b_2}$ are linearly independent. It turns out that there is a simple way to do this, which was discovered independently by Lagrange (first) and Gauss (second). We will call it *Lagrange's algorithm.* It is similar to the Gram-Schmidt process, but more complicated, since we are only allowed to adjust vectors by *integer* multiples of other vectors, because we need to stay inside the lattice L. Indeed, given any two linearly independent vectors in $\mathbb{R}^2$, the Gram-Schmidt process terminates after one step, while Lagrange's algorithm will terminate, but, depending on the initial two vectors, could take arbitrarily many steps to do so.

Figure 1 illustrates how Lagrange's algorithm works. We start with $\mathbf{b_1}$ (shorter) and $\mathbf{b_2}$ (longer, or, for the first step only, at least as long as $\mathbf{b_1}$). Next, we reduce the longer vector $\mathbf{b_2}$ by adding or subtracting multiples of the shorter vector $\mathbf{b_1}$, to get $\mathbf{b_3} = \mathbf{b_2} - q\mathbf{b_1}$, where the integer q is chosen to make $||\mathbf{b_3}||$ as short as possible. For $\mathbf{b_1} = (3,1)^\mathsf{T}$ and $\mathbf{b_2} = (5,3)^\mathsf{T}$, illustrated on the left of Figure 1, we get $\mathbf{b_3} = (-1,1)^\mathsf{T}$. Notice that $||\mathbf{b_3}|| = \sqrt{2}$, so that $\mathbf{b_1}$ and $\mathbf{b_2}$ have exchanged places; the shortened version $\mathbf{b_3}$ of $\mathbf{b_2}$ is now *strictly* shorter than $\mathbf{b_1}$. This means that we should exchange the vectors, setting $\mathbf{b_1'} = \mathbf{b_3}$ and $\mathbf{b_2'} = \mathbf{b_1}$, and repeat. The second step of the algorithm is shown on the right of Figure 1. The new shortened vector is $\mathbf{b_3'} = (2,2)^\mathsf{T}$, of length $\sqrt{8}$.

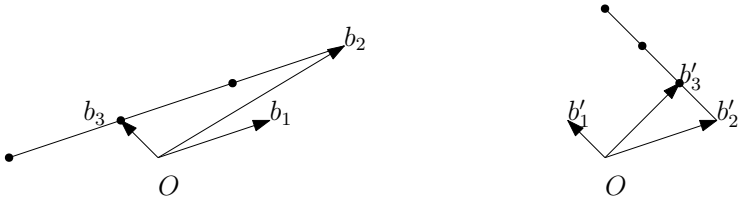

Figure 1: Two steps of Lagrange's algorithm.

But now there is a problem. This time, $\mathbf{b_1'}$ and $\mathbf{b_2'}$ have *not* exchanged places; the adjusted longer vector $\mathbf{b_3'}$ is *still* longer than (or, in general, at least as long as) the shorter vector $\mathbf{b_1'}$. The resulting basis, consisting of $\mathbf{b_1''} = \mathbf{b_1'}$ and $\mathbf{b_2''} = \mathbf{b_3'}$, is said to be *Lagrange-reduced*. This means that $\mathbf{b_1''}$ and $\mathbf{b_2''}$ satisfy

$$||\mathbf{b_1''}|| \leq ||\mathbf{b_2''}|| \leq ||\mathbf{b_2''} - \mathbf{q}\mathbf{b_1''}||$$

for all integers q. Lagrange's algorithm terminates here. (In this example, the final basis is orthogonal, but in general it need not be.)

It is nontrivial that Lagrange's algorithm does yield two basis vectors of minimal length, i.e., $\mathbf{b_1}$ is a minimal non-zero element of L, and $\mathbf{b_2}$ is of minimal length in $L \setminus \mathbb{Z}\mathbf{b_1}$. The proof, left as an exercise in [4], and presented in full in [1], is about half a page long.

Lagrange's algorithm is very reminiscent of the Euclidean algorithm, with two quantities leapfrogging each other in a race to the bottom. Consequently, it may now be clear to the reader that (as stated without proof in [12]) Brillhart's algorithm is just Lagrange's algorithm applied to Grace's lattice (1), starting with $\mathbf{b_1} = (u, 1)^\mathsf{T}$ and $\mathbf{b_2} = (p, 0)^\mathsf{T}$. But the algorithms are in fact different, even for the case $p = 73, u = 27$ considered earlier. Before proceeding, the reader is encouraged to run the two algorithms for this special case.

3 The plot thickens.

The basic connection between Lagrange's algorithm and Brillhart's is that, *at least initially*, the x-components of the sequence of vectors produced by Lagrange's algorithm (applied to $(p, 0)^\mathsf{T}$ and $(u, 1)^\mathsf{T}$) are the remainders in the Euclidean algorithm (applied to p and u). Indeed, if the y-components are small, we can neglect them, so that Lagrange's algorithm is approximated by its one-dimensional version, which is precisely the Euclidean algorithm.

Or is it? Figure 2 shows the result of applying Lagrange's algorithm to the vectors $(73, 0)^\mathsf{T}$ and $(27, 1)^\mathsf{T}$. It does indeed produce (essentially) the same answer as Brillhart's. But look at the x-components. Instead of the expected sequence $73, 27, 19, 8, 3$, we see $73, 27, 3, -8$. The number 19 is absent entirely, and 8 and 3 are replaced by 3 and -8. So the two algorithms are different. They are also different when, for example, $p = 277$.

The resolution to this paradox is that the one dimensional version of Lagrange's algorithm is *not* the ordinary version of the Euclidean algorithm with non-negative remainders. Instead, it is the version where the remainders are chosen to minimize their *absolute values*. So the obvious way to connect the algorithms is to modify Brillhart's algorithm to use this version of Euclid's algorithm instead. Then it will run even faster.

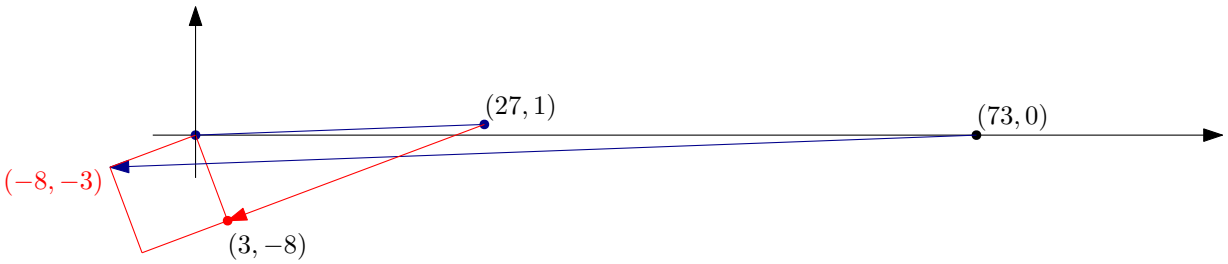

Figure 2: Lagrange's algorithm.

The catch is that the fast Brillhart algorithm doesn't work, in the sense that it doesn't produce the required goods! Taking $p = 277$ and $u = 60$, we get

$$\begin{aligned} 277 &= 5 \cdot 60 - 23 \\ 60 &= 3 \cdot 23 - 9 \\ 23 &= 3 \cdot 9 - 4 \end{aligned}$$

which misses the solution $9^2 + 14^2 = 277$.

Fortunately, there is another possibility. We can modify *Lagrange's* algorithm instead. And the obvious way to do this is to insist that every vector it generates has positive x-component. Figure 3 shows the result of applying this modification of Lagrange's algorithm with initial vectors $(73, 0)^\mathsf{T}$ and $(27, 1)^\mathsf{T}$. This time, the sequence of x-components does match the (positive) remainders. And, finally, it is not hard to modify the proof from [1] to show that the modified Lagrange algorithm also terminates with a Lagrange-reduced basis, at least for the lattices under consideration.

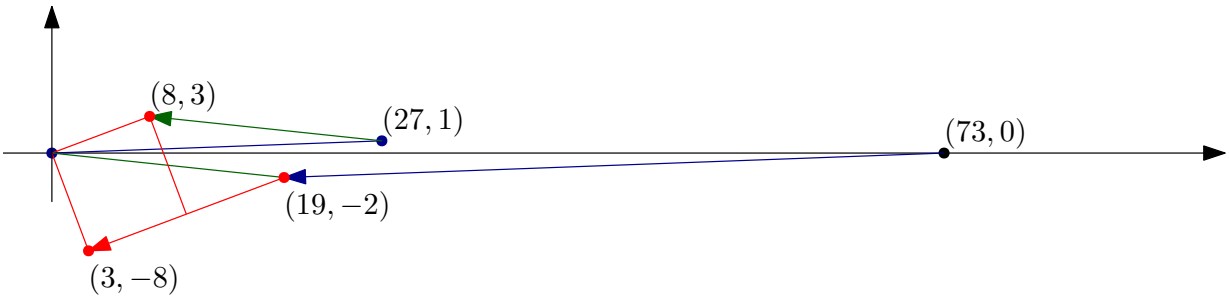

Figure 3: Lagrange's algorithm, restricted to $x > 0$.

4 A calculation.

There is one final hurdle to clear. These approximations are only valid when the y-components of the relevant vectors are small. Once they start growing, the shortest vector $\mathbf{b_3}$ from (the modified) Lagrange algorithm might not be the one with the smallest x-component. We will show that, in fact, these vectors coincide, until both algorithms terminate with the same result.

To do this, it will be convenient to change notation slightly, and to work with points in $\mathbb{R}^2$, instead of their position vectors. Let's introduce the notation by restating both algorithms.

Ingredients. $p \equiv 1 \pmod 4$ is a prime, and u satisfies $u^2 \equiv -1 \pmod p$ with $1 \leq u < p/2$.

Initialization. We start with $\mathbf{x}_0 = (x_0, y_0) = \mathbf{x}'_0 = (p, 0)$ and $\mathbf{x}_1 = (x_1, y_1) = \mathbf{x}'_1 = (u, 1)$.

The Extended Brillhart (EB) Algorithm [2]. For $i = 1, 2, \ldots$, we write $x_{i-1} = q_i x_i + r_i$ with $0 \leq r_i < x_i$, and set

$$\mathbf{x}_{i+1} = \mathbf{x}_{i-1} - q_i \mathbf{x}_i = (x_{i-1} - q_i x_i, y_{i-1} - q_i y_i) = (r_i, y_{i-1} - q_i y_i).$$

Note that here we have extended Brillhart's algorithm to produce points in $\mathbb{R}^2$, and we have also extended it to run until Euclid's algorithm terminates. The sequence (x_i) is just the sequence of remainders (r_i) from the Euclidean algorithm applied to p and u, while the sequence (y_i) is also related to the Euclidean algorithm: we have $r_i = s_i p + y_i u$ for another sequence (s_i). Indeed the sequences (s_i) and (y_i) are just the coefficients in the recursive algorithm for expressing $1 = \gcd(p, u)$ as an integer linear combination of p and u; they are defined by the same recursion $y_{i+1} = y_{i-1} - q_i y_i$ (respectively $s_{i+1} = s_{i-1} - q_i s_i$), but with different starting values (we have $s_0 = y_1 = 1$ and $y_0 = s_1 = 0$).

For example, with $p = 193$ we have $u = 81$, and successive $\mathbf{x}_i$ are given by

$$(193, 0), (81, 1), (31, -2), (19, 5), (12, -7), (7, 12), (5, -19), (2, 31), (1, -81), (0, 193).$$

The symmetry is related to the fact that $u^2 \equiv -1 \pmod p$; we will derive it as a consequence of the arguments below.

The Modified Lagrange (ML) Algorithm. For $i = 1, 2, \ldots$, as long as $||\mathbf{x}'_i|| \leq ||\mathbf{x}'_{i-1}||$, write

$$q'_i = \left\lfloor \frac{\mathbf{x}'_i \cdot \mathbf{x}'_{i-1}}{||\mathbf{x}'_i||^2} \right\rfloor,$$

and set

$$\mathbf{x}'_{i+1} = \mathbf{x}'_{i-1} - q'_i \mathbf{x}'_i.$$

Our aim is to show that, with the above choices of p and u, the initial points $\mathbf{x_0}, \mathbf{x_1}, \ldots$ produced by the EB algorithm do indeed coincide with the complete sequence $\mathbf{x'_0}, \mathbf{x'_1}, \ldots, \mathbf{x'_t}$ of points computed by the ML algorithm ending with $\mathbf{x'_t} = (x'_t, y'_t)$. (In the example above, $(x_t, y_t) = (x'_t, y'_t) = (7, 12)$.) In other words, we need to show that $q_i = q'_i$, until the ML algorithm terminates.

We recall some details from earlier. First, the integer lattice L given by

$$L = \{(x, y) \in \mathbb{Z}^2 : x \equiv uy \pmod{p}\}$$

is a square lattice with determinant p, so that, for all $(x, y) \in L$, p divides $x^2 + y^2$, and, if $(x, y) \in L$ is a point closest to the origin, then $p = x^2 + y^2$. Consequently, if the ML algorithm terminates at $\mathbf{x'}_t = (x'_t, y'_t)$, then $p = ||\mathbf{x'}_t||^2 = (x'_t)^2 + (y'_t)^2$.

We now observe that each of the points $\mathbf{x}_i$ (and $\mathbf{x}'_i$) lies on L. Consequently, for all $i \geq 1$,

$$p \mid (x_i^2 + y_i^2). \tag{2}$$

Also, since p divides the right hand side of the equation

$$2(x_{i-1}x_i + y_{i-1}y_i) = ||\mathbf{x}_{i-1} + \mathbf{x}_i||^2 - ||\mathbf{x}_{i-1}||^2 - ||\mathbf{x}_i||^2,$$

it must also divide the left hand side, and so, since p is odd,

$$p \mid (x_{i-1}x_i + y_{i-1}y_i). \tag{3}$$

Moreover, it is easy to see by induction that, for all $i \geq 1$,

$$x_{i-1}y_i - x_iy_{i-1} = (-1)^{i+1}p. \tag{4}$$

We can use these properties to establish the symmetry observed earlier. Suppose the EB algorithm terminates at (x_T, y_T). Then, since the x_i are just the reminders r_i in the Euclidean algorithm, we must have $x_T = 0$ and $x_{T-1} = 1$. By (4) we get $y_T = p$, and, since $(x_{T-1}, y_{T-1}) \in L$, we then obtain $y_{T-1} = -u$. We can now establish the required symmetry by induction (see also [11]). Moreover, since the continued fraction expansion of p/u is precisely the sequence of quotients q_i from the EB algorithm, this symmetry furnishes the palindromic property proved by Perron.

Now, it is easy to show, by induction, that the y_i alternate in sign, and that the sequence $|y_i|$ is increasing. From the remarks above, the sequence $(|y_i|)$ is just the sequence (x_i) in reverse order. Consequently, we can stop iterating once $x_i \leq |y_i|$. Thus, in the following, we will assume that $x_i > |y_i|$.

Claim. For all $i \geq 1$, $q_i = q'_i$.

Proof. We use induction on i. The induction starts. For the induction step, we need to show that

$$q_i := \left\lfloor \frac{x_{i-1}}{x_i} \right\rfloor = \left\lfloor \frac{x_{i-1}x_i + y_{i-1}y_i}{x_i^2 + y_i^2} \right\rfloor =: q_i'.$$

For simplicity of notation, write a, b, c, d for $x_{i-1}, y_{i-1}, x_i, y_i$ respectively. Then we need that

$$\left\lfloor \frac{a}{c} \right\rfloor = \left\lfloor \frac{ac + bd}{c^2 + d^2} \right\rfloor.$$

Note that a and c are positive, while b and d have opposite signs, so that $bd < 0$. Therefore

$$\frac{ac + bd}{c^2 + d^2} < \frac{a}{c},$$

so that $q_i' \leq q_i$. Suppose that $q_i' < q_i$. Then there is an integer m with

$$\frac{ac + bd}{c^2 + d^2} < m \leq \frac{a}{c}. \tag{5}$$

Now, from (2), (3), and (4) above, we know there are integers K and L such that

$$ac + bd = pK,\ c^2 + d^2 = pL \text{ and } d(ad - bc) = |d|p.$$

Multiplying (5) by $c(c^2 + d^2)$, and noting that $|d| = |y_i| < x_i = c$, we get

$$\begin{aligned} cpK &= c(ac + bd) < mc(c^2 + d^2) \\ &= mcpL \leq a(c^2 + d^2) \\ &= c(ac + bd) + d(ad - bc) \\ &= cpK + |d|p < cpK + cp \\ &= cp(K + 1), \end{aligned}$$

so that, dividing both sides by cp, we have

$$K < mL < K + 1,$$

a contradiction.

Exercise. Explain why the fast Brillhart algorithm doesn't work!

5 Epilogue.

There is another connection between Lagrange reduction and the Euclidean algorithm. Identify $\mathbb{R}^2$ with $\mathbb{C}$. A short calculation reveals that, for Grace's lattice L, *unmodified* Lagrange reduction, starting with initial points $(p, 0) \equiv p$ and $(u, 1) \equiv u + i$, is just the Euclidean algorithm applied in $\mathbb{Z}[i]$ to compute a greatest common divisor $x + yi$ of p and $u + i$.

As mentioned by Wagon [11], Brillhart's algorithm works for composite n, as long as one can solve $u^2 \equiv -1 \pmod{n}$. And indeed, Grace's proof works in this case too. The catch is that there is now no good algorithm for finding u.

Finally, we should mention that Brillhart's algorithm has been generalized by Hardy, Muskat, and Williams [7, 8] to solve both $n = ax^2 + by^2$ and $n = ax^2 + bxy + cy^2$ (for appropriate a, b, c and composite n). These algorithms are considerably more complicated, and the palindromic property no longer holds.

6 Acknowledgment.

Both authors are extremely grateful to the second author's teacher and mentor, Peter Shiu, whose paper [10] was the inspiration for this project, and whose encouragement and feedback has been indispensable.

References

[1] C. Bright, Algorithms for lattice basis reduction, available at:
https://cs.uwaterloo.ca/∼cbright/reports/latticealgs.pdf.

[2] J. Brillhart, Note on representing a prime as a sum of two squares, *Math. Comp.* **26** (1972), 1011–1013.

[3] S. Dolan, A very simple proof of the two-squares theorem, *Math. Gaz.* **105** (2021), p. 511.

[4] S. Galbraith, Mathematics and Public Key Cryptography, Cambridge University Press, 2012.

[5] J. Grace, The four square theorem, *J. Lond. Math. Soc.* **2** (1927), 3–8.

[6] G.H. Hardy, A Mathematician's Apology, Cambridge University Press, 1940.

[7] K. Hardy, J. Muskat and K. Williams, A deterministic algorithm for solving $n = fu^2 + gv^2$ in coprime integers u and v, *Math. Comp.* **55** (1990), 327-–343.

[8] K. Hardy, J. Muskat, and K. Williams, Solving $n = au^2 + buv + cv^2$ using the Euclidean algorithm, *Utilitas Math.* **38** (1990), 225-–236.

[9] D.R. Heath-Brown, Fermat's two-squares theorem, *Invariant* (1984), 3–5.

[10] P. Shiu, The two squares theorem of Fermat for a prime $p \equiv 1 \pmod 4$, *Newsletter Lond. Math. Soc.* **494** (2021), 26–31.

[11] S. Wagon, The Euclidean algorithm strikes again, *Amer. Math. Monthly* **97** (1990), 125–129.

[12] B. de Weger, Lagrange's algorithm strikes again, available at:
https://www.win.tue.nl/∼bdeweger/downloads/sumof2squares.pdf.

[13] D. Zagier, A one-sentence proof that every prime $p \equiv 1 \pmod 4$ is a sum of two squares, *Amer. Math. Monthly* **97** (1990), p. 144.