Generating Pythagorean Quadruples with One Variable

Nadav Voloch[1]

[1] IMT School of advanced studies, Lucca, Italy
voloch@post.bgu.ac.il

Abstract. This paper introduces a novel method for generating Pythagorean quadruples using a single-variable algebraic approach derived from the classical Platonic sequence. While traditional methods for generating Pythagorean triples are well known and deeply rooted in number theory, their extension to quadruples has remained less accessible. By building on the established formulas for triples and applying recursive logic, this work presents a set of formulas that yield valid Pythagorean quadruples with only one input variable. The approach distinguishes between even and odd values of the input variable and constructs each quadruple component step-by-step using simple algebraic expressions. Although the method does not account for all primitive quadruples, it offers a highly efficient and elegant means to generate a wide range of valid solutions. Additionally, the paper provides full algebraic proofs for correctness and includes a Python implementation that can be used to automate the generation and verification process. The simplicity of the method makes it particularly useful for educational, computational, and exploratory purposes. This work also forms a basis for potential future developments in higher-dimensional Diophantine analysis and applications in cryptography, where integer-based geometric structures play an important role.

Keywords: Pythagorean quadruples, Pythagorean triples, Number theory, Cryptography, Algebraic Geometry.

1 Introduction

Pythagorean quadruples are sets of four positive integers (a, b, c, d) that satisfy the Diophantine equation $a^2+b^2+c^2=d^2$.
These quadruples can be viewed as a natural extension of the well-known Pythagorean triples (a, b, c), which satisfy $a^2+b^2=c^2$.While Pythagorean triples have been extensively studied due to their historical and theoretical significance in number theory and geometry, Pythagorean quadruples offer a higher-dimensional generalization with intriguing algebraic and geometric properties.
The study of Pythagorean quadruples lies at the intersection of number theory, algebraic geometry, and the theory of quadratic forms. These tuples not only serve as solutions to Diophantine equations ([1]) but also have applications in areas such as lattice point geometry, cryptography, and theoretical physics. Investigating their structure, methods of generation, and classification yields insight into the deeper patterns and symmetries underlying integer solutions to quadratic equations.

Figure 1 illustrates a rectangular box with side lengths 2, 3, and 6 units. The diagonal connecting two opposite corners of the box, shown as a dashed line, measures 7 units.

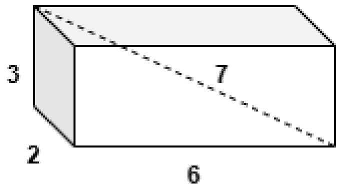

Fig. 1. A rectangular box with sides 2, 3, and 6 units and a space diagonal of 7 units, forming a Pythagorean quadruple.

This visual serves as a geometric representation of a Pythagorean quadruple $(2, 3, 6, 7)$, where the sum of the squares of the three edge lengths equals the square of the space diagonal: $2^2+3^2+6^2 = 4+9+36 = 49 = 7^2$.

This configuration highlights a practical, three-dimensional interpretation of the Pythagorean quadruple equation $a^2+b^2+c^2=d^2$. As shown in Fig. 1, the diagonal across the box (also known as the space diagonal) corresponds to the hypotenuse in three dimensions, derived by applying the Pythagorean theorem successively in two orthogonal planes.

In this research a unique method is shown for generating Pythagorean Quadruples with only one variable. The method is based on the Platonic sequence ([2]), and using the basic formulas accordingly, with some algebraic development. The methos does not cover all primitive cases, but the uniqueness of it is of using one variable, and more importantly, its simplicity. A Python software was also devised for the purpose of this research, that can generate as many Pythagorean Quadruples as one wishes and checks their correctness in adherence to the equation.

2 Background and related work

Pythagorean quadruples, defined as integer solutions (a, b, c, d) to the Diophantine equation $a^2+b^2+c^2=d^2$ represent a natural generalization of the classical Pythagorean triples. These tuples have been studied within the broader context of Diophantine equations and quadratic forms, providing insights into number theory and lattice geometry [3]. Historically, Pythagorean triples received extensive treatment in ancient mathematics, but higher-dimensional generalizations like Pythagorean quadruples remained less explored until more recent studies. The foundational methods for generating such quadruples include both algebraic parameterizations and geometric interpretations, often relying on extensions of Euclid's formula for generating triples [4].

In contemporary research, Pythagorean quadruples are analyzed not only for their intrinsic mathematical interest but also for applications in computer science, particularly in optimization problems and cryptography, where lattice-based structures are relevant [5]. Recent studies have developed algorithms for efficiently enumerating and generating Pythagorean n-tuples using recursive techniques and matrix transformations [6]. These approaches often relate to representations of integers as sums of squares, a subject extensively treated in analytic number theory [7].

Furthermore, research has explored generalizations to Pythagorean n-tuples, where the sum of $n-1$ squares equal the square of the n^{th} term, expanding the scope of study and enabling applications in higher-dimensional geometry and theoretical physics [8]. The computational generation and analysis of Pythagorean quadruples have received attention in the context of both number-theoretic research and algorithmic development. Several studies have focused on efficient methods for generating quadruples using parametric formulas, recursive techniques, and computer algebra systems [6]. A notable implementation strategy involves extending Euclidean parametrization methods used for Pythagorean triples to higher dimensions, often requiring symbolic computation and number-theoretic libraries. For example, [3] proposed algorithmic techniques for enumerating quadruples with specific constraints, demonstrating implementations using procedural programming to optimize performance. Recent work by [9] introduced a matrix-based approach for generating generalized Pythagorean tuples, including quadruples. Their implementation leverages linear algebraic transformations and recursive iteration to efficiently compute valid tuples within a bounded range. The study also evaluates computational complexity and provides code-level optimizations suitable for large-scale enumeration tasks. In educational and recreational mathematics, implementations of such generators have been developed in software environments like Python, Mathematica, and MATLAB, enabling exploration of these number sets through visualization and simulation [10]. These implementations serve both research and pedagogical purposes, often illustrating connections between geometry and number theory in three-dimensional space. Moreover, the application of Pythagorean quadruple generation in cryptography, particularly in constructing hard lattice problems and obfuscation techniques, has been considered a promising direction, though still in early development [11].

3 A simple formula for generating Pythagorean Quadruples

3.1 Preliminaries – The Platonic sequence

The formula for generating Pythagorean triplets (The Platonic sequence) is inherently straightforward. Pythagoras himself developed the formula for odd lengths of the a side (perpendicular) around 540 BC, while Plato later introduced the formula for even a side length around 380 BC. The fundamental premise is that for any given integer a with a length of at least 3, a Pythagorean triplet can be derived using this formula for odd 'a' side length:

$$b = \frac{a^2-1}{2}\ ; c = b + 1 \tag{1}$$

And for the case of even 'a' side length, the formula is:

$$b = \left(\frac{a}{2}\right)^2 - 1\ ; c = b + 2 \tag{2}$$

As can be seen, these formulas can be manufactured by using on one chosen variable, that is a with a length of at least 3.

On the premise of these basic formulas, a bit more complex formulas can be developed for generating Pythagorean Quadruples.

3.2 Generating the formulas

For simplifying, let x be the variable for which the algebraic calculation is performed. x is with a length of at least 3. For all formulas the first side a is set as:

$$\boldsymbol{a = x} \tag{3}$$

At this point, two options exist for x (and a): x is odd, or x is even.

<u>x is odd</u>:

The b side is calculated as regularly in the Platonic sequence for odd a:

$$\boldsymbol{b = \frac{x^2-1}{2}} \tag{4}$$

To continue a viable Platonic sequence for a quadruple, the c side from the Pythagorean triples, that is $c = b + 1 \rightarrow c = \frac{x^2+1}{2}$ is now used as the free variable ($a = x$) on the b side formula of the triples:

$$\boldsymbol{c} = \frac{\left(\frac{x^2+1}{2}\right)^2 - 1}{2} = \frac{\frac{x^4+2x^2+1}{4} - 1}{2} = \boldsymbol{\frac{x^4}{8} + \frac{x^2}{4} - \frac{3}{8}} \tag{5}$$

And the last formula for the case of x is odd, for the d side, as c side was the equivalent of b side formula of the triples, the d side is the equivalent of c side formula of the triples:

$$\boldsymbol{d} = c + 1 = \boldsymbol{\frac{x^4}{8} + \frac{x^2}{4} + \frac{5}{8}} \tag{6}$$

<u>x is even</u>:

The b side is calculated as regularly in the Platonic sequence for even a:

$$\boldsymbol{b = \left(\frac{x}{2}\right)^2 - 1} \tag{7}$$

At this point, two options exist for the c side and the d side, as the b side can be either odd or even.

x is even, b is even:

To continue a viable Platonic sequence for a quadruple, the c side from the Pythagorean triples, that is $c = b + 2 \rightarrow c = \left(\frac{x}{2}\right)^2 + 1$ is now used as the free variable ($a = x$) on the b side formula of the triples on the even part:

$$\boldsymbol{c} = \left(\frac{\left(\frac{x}{2}\right)^2 + 1}{2}\right)^2 - 1 = \frac{\left(\frac{x^2}{4}\right)^2 + 2\frac{x^2}{4} + 1}{4} - 1 = \frac{\boldsymbol{x^4}}{\boldsymbol{64}} + \frac{\boldsymbol{x^2}}{\boldsymbol{8}} - \frac{\boldsymbol{3}}{\boldsymbol{4}} \tag{8}$$

And the last formula for the case of x is even, b is even, for the d side, as c side was the equivalent of b side formula of the triples, the d side is the equivalent of c side formula of the triples:

$$\boldsymbol{d} = c + 2 = \frac{\boldsymbol{x^4}}{\boldsymbol{64}} + \frac{\boldsymbol{x^2}}{\boldsymbol{8}} + \boldsymbol{1}\frac{\boldsymbol{1}}{\boldsymbol{4}} \tag{9}$$

***x* is even, *b* is odd:**

To continue a viable Platonic sequence for a quadruple, the c side from the Pythagorean triples, that is $c = b + 2 \rightarrow c = \left(\frac{x}{2}\right)^2 + 1$ is now used as the free variable $(a = x)$ on the b side formula of the triples on the odd part:

$$\boldsymbol{c} = \frac{\left(\left(\frac{x}{2}\right)^2 + 1\right)^2 - 1}{2} = \frac{\left(\frac{x^2}{4} + 1\right)^2 - 1}{2} = \frac{\boldsymbol{x^4}}{\boldsymbol{32}} + \frac{\boldsymbol{x^2}}{\boldsymbol{4}} \tag{10}$$

And the last formula for the case of x is even, b is even, for the d side, as c side was the equivalent of b side formula of the triples, the d side is the equivalent of c side formula of the triples:

$$\boldsymbol{d} = c + 1 = \frac{\boldsymbol{x^4}}{\boldsymbol{32}} + \frac{\boldsymbol{x^2}}{\boldsymbol{4}} + \boldsymbol{1} \tag{11}$$

3.3 Proof

In all the cases presented above the equations need to be checked in terms of adherence to the Pythagorean Quadruple equation of $a^2+b^2+c^2=d^2$.

***x* is odd:**

$$\boldsymbol{a^2} + \boldsymbol{b^2} + \boldsymbol{c^2} = x^2 + \left(\frac{x^2-1}{2}\right)^2 + \left(\frac{x^4}{8} + \frac{x^2}{4} - \frac{3}{8}\right)^2 = \frac{\boldsymbol{x^8+25}}{\boldsymbol{64}} + \frac{\boldsymbol{7x^4}}{\boldsymbol{32}} + \frac{\boldsymbol{x^6+5x^2}}{\boldsymbol{16}} \tag{12}$$

$$\boldsymbol{d^2} = \left(\frac{x^4}{8} + \frac{x^2}{4} + \frac{5}{8}\right)^2 = \frac{\boldsymbol{x^8+25}}{\boldsymbol{64}} + \frac{\boldsymbol{7x^4}}{\boldsymbol{32}} + \frac{\boldsymbol{x^6+5x^2}}{\boldsymbol{16}} \tag{13}$$

***x* is even, *b* is even:**

$$\boldsymbol{a^2} + \boldsymbol{b^2} + \boldsymbol{c^2} = x^2 + \left(\left(\frac{x}{2}\right)^2 - 1\right)^2 + \left(\frac{x^4}{64} + \frac{x^2}{8} - \frac{3}{4}\right)^2 = \frac{\boldsymbol{x^8}}{\boldsymbol{4096}} + \frac{\boldsymbol{x^6}}{\boldsymbol{256}} + \frac{\boldsymbol{7x^4}}{\boldsymbol{128}} + \frac{\boldsymbol{5x^2+25}}{\boldsymbol{16}} \tag{14}$$

$$\boldsymbol{d^2} = \left(\frac{x^4}{64} + \frac{x^2}{8} + 1\frac{1}{4}\right)^2 = \frac{\boldsymbol{x^8}}{\boldsymbol{4096}} + \frac{\boldsymbol{x^6}}{\boldsymbol{256}} + \frac{\boldsymbol{7x^4}}{\boldsymbol{128}} + \frac{\boldsymbol{5x^2+25}}{\boldsymbol{16}} \tag{15}$$

***x* is even, *b* is odd:**

$$\boldsymbol{a^2} + \boldsymbol{b^2} + \boldsymbol{c^2} = x^2 + ((\frac{x}{2})^2 - 1)^2 + \left(\frac{x^4}{32} + \frac{x^2}{4}\right)^2 = \frac{\boldsymbol{x^8}}{\boldsymbol{1024}} + \frac{\boldsymbol{x^6}}{\boldsymbol{64}} + \frac{\boldsymbol{x^4}}{\boldsymbol{8}} + \frac{\boldsymbol{x^2}}{\boldsymbol{2}} + \boldsymbol{1} \quad (16)$$

$$\boldsymbol{d^2} = \left(\frac{x^4}{32} + \frac{x^2}{4} + 1\right)^2 = \frac{\boldsymbol{x^8}}{\boldsymbol{1024}} + \frac{\boldsymbol{x^6}}{\boldsymbol{64}} + \frac{\boldsymbol{x^4}}{\boldsymbol{8}} + \frac{\boldsymbol{x^2}}{\boldsymbol{2}} + \boldsymbol{1} \quad (17)$$

4 Results

For this research, a Python-based program that implements the formulas presented here on all their variants, was developed. The details of the software are presented in [12], where there are also more results presented in a data file. Examples of its operation, showcasing various variables and the corresponding results for their equations generated by the formulas, are provided in Table 1 for Pythagorean Quadruples in the range of a < 100, Table 2 for Pythagorean Quadruples in the range of 100 < a < 200, and in Table 3 for Pythagorean Quadruples in the range of 200 < a < 300.

Table 1. Pythagorean Quadruples in the range of a < 100.

a	*b*	*c*	*d*
51	1300	846300	846301
88	1935	1875984	1875985
72	1295	841104	841105
53	1404	987012	987013
8	15	144	145
30	224	12768	12770
67	2244	2520012	2520013
27	364	66612	66613
66	1088	297024	297026
59	1740	1515540	1515541
95	4512	10183584	10183585
45	1012	513084	513085
57	1624	1320312	1320313

Table 2. Pythagorean Quadruples in the range of 100 < a < 200.

a	*b*	*c*	*d*
187	17484	152862612	152862613
180	8099	32813100	32813101
151	11400	64991400	64991401
106	2808	1974024	1974026
148	5475	14998764	14998765
198	9800	24019800	24019802
174	7568	14326224	14326226
178	7920	15689520	15689522
164	6723	22612812	22612813
161	12960	83993760	83993761
169	14280	101973480	101973481
190	9024	20367168	20367170
124	3843	7392012	7392013

5 Conclusion and Future Work

This paper presented a novel and accessible method for generating Pythagorean quadruples using a single-variable approach rooted in the classical Platonic sequence. By systematically extending the formulas traditionally used for generating Pythagorean triples, we derived compact expressions for quadruples under both even and odd conditions. These formulas, although not exhaustive in covering all primitive solutions, offer a valuable balance between algebraic simplicity and computational utility.

The method's strength lies in its ability to generate valid quadruples through straightforward algebraic manipulation, making it suitable for educational purposes and algorithmic implementation alike. The accompanying Python-based tool demonstrated the practicality of the formulas by generating a wide range of valid quadruples and verifying their correctness through symbolic computation.

Table 3. Pythagorean Quadruples in the range of 200 < a < 300.

a	*b*	*c*	*d*
260	16899	142821900	142821901
242	14640	53597040	53597042
215	23112	267105384	267105385
281	39480	779374680	779374681
288	20735	215011584	215011585
226	12768	40768224	40768226
267	35644	635283012	635283013
239	28560	407865360	407865361
245	30012	450390084	450390085
284	20163	203313612	203313613
272	18495	171069504	171069505
227	25764	331917612	331917613
293	42924	921277812	921277813
263	34584	598061112	598061113

This work contributes to the broader understanding of Diophantine equations and integer solutions in higher dimensions, while also laying a foundation for potential applications in cryptography and algorithmic number theory. The simplicity and elegance of the approach underscore the enduring power of classical mathematical techniques when extended with modern computational insights.

While considerable progress has been made in understanding and generating Pythagorean quadruples, the exploration of higher-dimensional analogues remains a fertile ground for future research. One natural extension involves the development of a general formula for Pythagorean 5-tuples—ordered sets of five integers (a, b, c, d, e) satisfying the equation $a^2 + b^2 + c^2 + d^2 = e^2$.

In the initial stages of this research, a foundational scheme plan was proposed in [13], with several ideas from the scheme further developed in [14] and [15]. The basis for this research was done in [15], where the general algorithmic heuristic was established along with several initial results.

Unlike the well-known parametrizations for Pythagorean triples and the more recent work on quadruples, there is currently no widely accepted closed-form formula or generation method for 5-tuples that systematically produces all primitive solutions. Identifying such a formula would not only extend the existing taxonomy of Diophantine identities but could also lead to new insights in higher-dimensional geometry and the theory of quadratic forms.

Moreover, an even more ambitious goal is to define a general construction or parametrization for Pythagorean n-tuples, where $n-1$ squared terms sum to the square of the n^{th} term: ${a_1}^2 + {a_2}^2 + \cdots + {a_{n-1}}^2 = {a_n}^2$.

This problem grows rapidly in complexity with increasing n, both algebraically and computationally. Future work may involve the use of matrix transformations, higher-dimensional lattice theory, or recursive algorithms to develop generalized generation schemes. Another promising direction includes leveraging symbolic computation and machine learning to search for patterns in known solutions, potentially leading to new conjectures or partial parametrizations.

The development of such formulas would not only enhance our understanding of Diophantine equations in higher dimensions but also broaden the applicability of these structures in areas such as cryptography, error-correcting codes, and theoretical physics. A promising avenue for future exploration involves investigating diverse communication systems requiring encryption solutions, particularly those employing complex and non-trivial communication protocols, as highlighted in [16].

References

1. Mordell, L. J. (1969). *Diophantine Equations: Diophantine Equations* (Vol. 30). Academic press.
2. Barnett, J. H. (2017). Generating Pythagorean triples: the methods of Pythagoras and of Plato via Gnomons.
3. Vashisht, L., & Sharma, V. (2023). On the generation of Pythagorean quadruples. Procedia Computer Science, 218, 748–757. https://doi.org/10.1016/j.procs.2023.01.127
4. Stillwell, J. (2010). Mathematics and Its History (3rd ed.). Springer.
5. Nguyen, P. Q., & Vallée, B. (Eds.). (2010). The LLL Algorithm: Survey and Applications. Springer.
6. Weisstein, E. W. (n.d.). Pythagorean n-Tuple. MathWorld. Retrieved from https://mathworld.wolfram.com/PythagoreanNTuple.html
7. Hardy, G. H., & Wright, E. M. (2008). An Introduction to the Theory of Numbers (6th ed.). Oxford University Press.
8. Dickson, L. E. (2005). History of the Theory of Numbers, Vol. II: Diophantine Analysis. Dover Publications.
9. Ghosh, M., & Pal, S. (2023). Generalized Pythagorean tuples via matrix transformations. *arXiv preprint arXiv:2303.05776*. https://arxiv.org/abs/2303.05776

10. Elkin, M. (2018). Integer lattice visualization for Pythagorean tuples. *Mathematics in Computer Science*, *12*(2), 235–248. https://doi.org/10.1007/s11786-017-0345-5
11. Nguyen, P. Q., & Regev, O. (2010). Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *Journal of Cryptology*, *23*(1), 1–25. https://doi.org/10.1007/s00145-009-9048-5
12. https://github.com/nadavvoloch/PythagoreanQuadruples
13. Voloch, Benjamin (1996), Ben Gurion University of the Negev, & Baruch, Moshe (2005), The Hebrew University of Jerusalem, Israel. Private communication.
14. Voloch, N., Birnbaum, E., & Sapir, A. (2014, December). Generating error-correcting codes based on tower of Hanoi configuration graphs. In 2014 IEEE 28th Convention of Electrical & Electronics Engineers in Israel (IEEEI) (pp. 1-4). IEEE.
15. Voloch, N., & Raicol, N. V. B. (2024, September). A Cryptographic Encryption Scheme based on a Pythagorean Triplets Manufacturing Formula. In *2024 14th International Conference on Advanced Computer Information Technologies (ACIT)* (pp. 529-534). IEEE.
16. Voloch, N., & Hajaj, M. M. (2022, November). Handling Exit Node Vulnerability in Onion Routing with a Zero-Knowledge Proof. In International Conference on Information Integration and Web (pp. 399-405). Cham: Springer Nature Switzerland.