Journal of Mathematics Research; Vol. 13, No. 3; June 2021
ISSN 1916-9795 E-ISSN 1916-9809
Published by Canadian Center of Science and Education

Pythagorean Triples Generator

Pranay Padavala

Correspondence: Chettinad Vidyashram, Chennai, India

Received: April 11, 2021 Accepted: May 20, 2021 Online Published: May 26, 2021

doi:10.5539/jmr.v13n3p63 URL: https://doi.org/10.5539/jmr.v13n3p63

Abstract

Given a leg of a right-angled triangle a, this formula gives the other leg b and the hypotenuse c by the usage of a pattern observed in Pythagorean triples. This is different as compared to Euclids method since Euclids method takes two arbitrary numbers as the input while this uses the side of the right-angled triangle as the input.

Keywords: triples generator, Pythagorean Triples Tree, PPT (Primitive Pythagorean Triples)

1. Introduction

1.1 The Current Problem

The Euclid's formula is the most commonly used formula for generating Pythagorean triples given an arbitrary pair of integers m and n with $m > n > 0$. The limitation in this formula is that we have no control over the values of the sides of the triangle. For instance, if we want to find a Pythagorean triple with side length of 156, this is quite a challenge with Euclid's method. However, the basis of the method presented in this paper has complete control over the side length.

1.2 Alternate Methods

There have been various other methods besides Euclid's formula that have been devised for generating Pythagorean triples such as:

- Fibonacci's Method - Fibonacci described a method for generating primitive triples for the sequence of odd natural numbers and the fact that the sum of the first n terms of this sequence is n^2. The method shared in this paper uses Fibonacci's Method as a source of inspiration.
- Dickson's Method - If we find positive integers r, s, and t such that $r^2 = 2st$ is a perfect square. Then $x = r + s$, $y = r + t$, $z = r + s + t$. where x, y, z are the side lengths and hypotenuse respectively.
- Michael Stifel's Method - It uses the progression of the $1\frac{1}{3}, 2\frac{2}{5}, 3\frac{3}{7}, 4\frac{4}{9}$... as the length of the sides of right angled triangle.

1.3 The Pattern

Lets look at the first few Pythagorean triples,

(3, 4, 5), (5, 12, 13), (7, 24, 25), (8, 15, 17), (9, 40, 41), (12, 35, 37)

Drawing inspiration from Fibonaccis method which described a method for odd first numbered triples, we categorize these triples based on whether the input is odd or even.

Let the first number be the input here,

Odd: (3, 4, 5), (5, 12, 13), (7, 24, 25), (9, 40, 41)

Even: (8, 15, 17), (12, 35, 37)

We now notice a pattern between the other leg b and hypotenuse c.

In the odd series, they differ by 1 and in the even series, they differ by 2.

2. The Formula

2.1 Analysing the Relation

Using the fact observed previously, we derive that,

$$c = b + 1 \text{ (odd)} \tag{1}$$

$$c = b + 2 \text{ (even)} \tag{2}$$

However, this pattern is not applicable to all Pythagorean triples since the next triple would be (28, 45, 53) which does not follow the pattern.

Therefore, this formula will generate infinite Pythagorean triples but does not generate **all** Pythagorean triples.

2.2 Derivation of the Formula

By plugging in the results developed previously into the Pythagoras theorem, we can derive the formulas.

For odd inputs,

$$a^2 + b^2 = c^2$$

since $c = b + 1$,

$$a^2 + b^2 = (b+1)^2$$

$$a^2 + b^2 = b^2 + 2b + 1$$

$$\mathbf{b = \frac{(a^2 - 1)}{2}} \tag{3}$$

$$b + 1 = \frac{(a^2 - 1)}{2} + 1$$

$$\mathbf{c = \frac{(a^2 + 1)}{2}} \tag{4}$$

For even inputs,

$$a^2 + b^2 = c^2$$

since $c = b + 2$,

$$a^2 + b^2 = (b+2)^2$$

$$a^2 + b^2 = b^2 + 4b + 4$$

$$\mathbf{b = \frac{a^2}{4} - 1} \tag{5}$$

$$b + 2 = \frac{a^2}{4} - 1 + 2$$

$$\mathbf{c = \frac{a^2}{4} + 1} \tag{6}$$

2.3 Final Result

Hence the final formula would be,

$$f(a) = \begin{cases} b = \frac{a^2-1}{2}, c = \frac{a^2+1}{2} & \text{where } a = 2n+1 \text{ and } n \in \mathbb{Z} \\ b = \frac{a^2}{4} - 1, c = \frac{a^2}{4} + 1 & \text{where } a = 2n \text{ and } n \in \mathbb{Z} \end{cases}$$

3. An Exploration

Now that the formula has been derived, we can try different inputs to receive interesting results.

3.1 The Base Examples

Exhibit 1 $f(11)$

$$b = \frac{120}{2} = 60; c = \frac{122}{2} = 61$$

The triplet is (11,60,61)

Exhibit 2 $f(14)$

$$b = 49 - 1 = 48; c = 49 + 1 = 50$$

The triplet is (14,48,50)

The above two exhibits show results as expected.

3.2 Few Special Cases

Exhibit 3 $f(2)$

$$b = 1 - 1 = 0; c = 1 + 1 = 2$$

The triplet is (2,0,2)

Exhibit 4 $f(1)$

$$b = \frac{0}{2} = 0; c = \frac{2}{2} = 1$$

The triplet is (1,0,1)

Exhibit 5 $f(0)$

0 can be represented as 2n where n = 0, the second set of formulas would be applied to it.

$$b = 0 - 1 = -1; c = 0 + 1 = 1$$

The triplet is (0,-1,1)

Hence, we can conclude that,

- Giving an input of 0,1 or 2 results in having one of the sides as a zero length.
- Zero is the only input thats able to produce a negative Hypotenuse.

3.3 The Negative Integers

Exhibit 6 $f(-5)$

-5 can be written as 2(-3) + 1, the first set of formulas would be applied to it.

$$b = \frac{24}{2} = 12; c = \frac{26}{2} = 13$$

The triplet is (-5,12,13)

Exhibit 7 $f(-6)$

-6 can be written as 2(-3), the second set of formulas would be applied to it

$$b = 9 - 1 = 8; c = 9 + 1 = 10$$

The triplet is (-6,8,10)

Hence, the sign of the input does not play a role here as the input is squared in every formula.

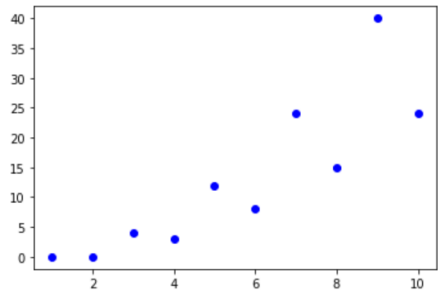
4. The Code

Given below is the algorithm written in the programming language Python.

The following code can be executed without any installation in Google Collab or can be executed in Jupyter Notebook if installed.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

def mypythagoreantriplet(a) :
    #Generates the triplet for the given input
        if(a%2 == 0):
            b = (a**2 / 4) - 1
            c = (a**2 / 4) + 1
            print(int(a), int(b), int(c))
```

```
        else:
            b = (a**2 - 1) / 2
            c = (a**2 + 1) / 2
            print(int(a), int(b), int(c))
def mypythagoreantripletgen(limit) :
    #Generates triples from 1 up to the given input
    for a in range(1,limit+1):
        if(a%2 == 0):
            b = (a**2 / 4) - 1
            c = (a**2 / 4) + 1
            print(int(a), int(b), int(c))
        else:
            b = (a**2 - 1) / 2
            c = (a**2 + 1) / 2
            print(int(a), int(b), int(c))
def mypythagoreantripletchart(limit) :
    #Provides a visual representation of the previous function
    a_values = []
    b_values = []
    c_values = []
    for a in range(1,limit+1):
        if(a%2 == 0):
            b = (a**2 / 4) - 1
            c = (a**2 / 4) + 1
        else:
            b = (a**2 - 1) / 2
            c = (a**2 + 1) / 2
        a_values.append(a)
        b_values.append(b)
        c_values.append(c)
    plt.plot(a_values,b_values,'bo')
    #Values of X and Y axes can be changed by easily tweaking the values in the function

mypythagoreantripletchart(10)
```

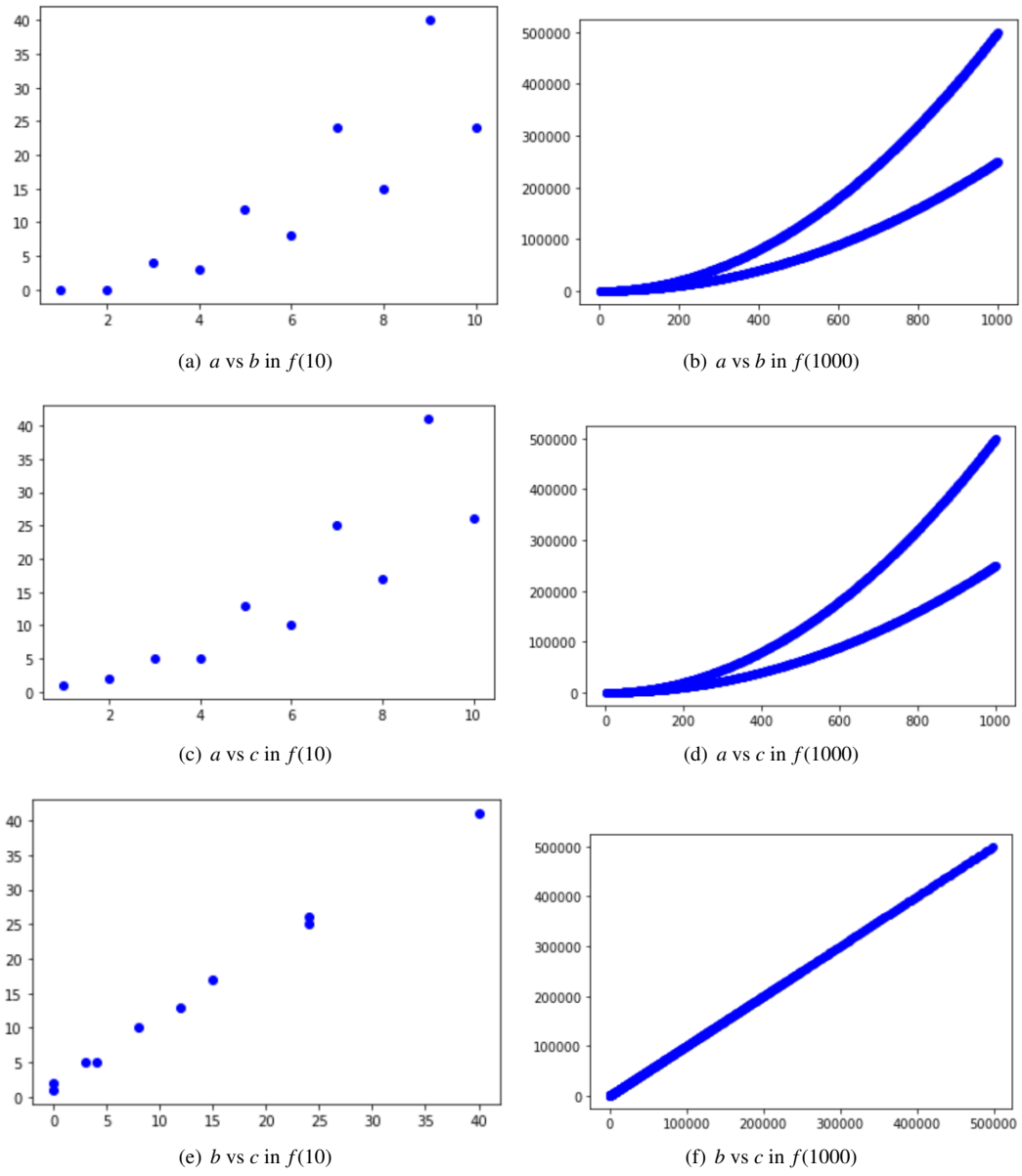



```
mypythagoreantriplet(7)

7 24 25

mypythagoreantripletgen(3)

1 0 1
2 0 2
3 4 5
```

5. The Graphs

A graphical representation can help us understand how the three quantities vary when compared to each other.

Therefore using the `mypythagoreantripletchart()` function shown before, we shall observe the a vs b, a vs c and b vs c graphs. We can observe the negligible difference between b and c.



(a) a vs b in $f(10)$ (b) a vs b in $f(1000)$

(c) a vs c in $f(10)$ (d) a vs c in $f(1000)$

(e) b vs c in $f(10)$ (f) b vs c in $f(1000)$

Figure 1. Graphs between the three sides a, b, c

6. Conclusion and the Way Forward

Prime numbers and Pythagorean Triples generation are a vital component in Cryptography. Developing methods that enable more flexibility in the generation process will benefit the Cryptography domain.

The Development of new methods of generating Pythagorean triples will not stop here. Although this paper showcases a method with flexible input values, The ability to generate all the Pythagorean triples is constrained.

Hence, methods that involve flexibility in input, ease of implementation and the ability to generate all Pythagorean Triples is the goal that we should try to achieve.

Acknowledgements

I would like to thank the anonymous reviewer and the editor for their valuable comments.

References

Fässler, A. (June 1991). *Multiple Pythagorean number triples*. American Mathematical Monthly. https://doi.org/10.2307/2324870

Joyce, D. E. (June 1997). *book X , Proposition XXIX*. Euclid's Elements, Clark University. Retrieved from https://mathcs.clarku.edu/ djoyce/java/elements/bookX/propX29.html

Posamentier, A. S. (2010). *The Pythagorean Theorem: The Story of Its Power and Beauty.* Retrieved from https://archive.org/details/pythagoreantheor0000posa/mode/2up

Saunders, R., & Randall, T. (1994). 78.12 The Family Tree of the Pythagorean Triplets Revisited. *The Mathematical Gazette, 78*(482), 190-193. https://doi.org/10.2307/3618576

Stillwell, J. (2002). *6.6 Pythagorean Triples*, Elements of Number Theory, Springer, pp. 110-116. Retrieved from https://books.google.com/books?id=LiAlZO2ntKAC&pg=PA110

Subhash, K. (2010). *Pythagorean Triples and Cryptographic Coding*. Oklahoma State University.

Vincent, P. M. D. R., Iqbal, S. A., Bhagat, K., & Kushwaha, K. K. (2013). Cryptography: A mathematical approach. *Indian Journal of Science and Technology, 6*(12), 5607-11. https://doi.org/10.17485/ijst/2013/v6i12.12