A Numerical Approach to Discovering Relationships between Science Constants

Tamara Azzam*
North Hills Preparatory, 7116 Sugar Maple Drive, Irving, Tx, 75063, USA.

* Corresponding author. Email: tamarahkazzam@gmail.com
Manuscript submitted January 17, 2021; accepted March 29, 2021.
doi: 10.17706/ijapm.2021.11.3.50-60

Abstract: A numerical approach is proposed to explore relationships between constants. A subset of constants is assembled and a simple method is used to probe how each constant in the set might be related to a combination of the rest. The goal is to check if any of the constants can be expressed as a combination of two, three, or four constants using simple arithmetic operations (addition, subtraction, multiplication, division and power of positive and negative numbers). A software program, implemented in Python, was employed to search for such simple relationships. Some of the values found were within 10^{-11} accuracy. Others were simply esthetically pleasing like $\boldsymbol{\pi} \approx \frac{7^7}{8^6}$ and $\boldsymbol{e} \approx \frac{\pi^7}{1111}$. Last but not least, albeit long, we were able to find an expression for π with 1000 decimal places.

Key words: Numerical approximation, math constants, physics constants.

1. Introduction

Science constants are important fixed entities that are often linked to the foundations of nature. From π and e in math, to h and G in physics, these entities play an important role in helping us understand the world around us. They act as light beacons that connect experimental and theoretical domains and very often they become conduits to understanding the fundamentals of nature herself. Some of these constants, like π, have been around for thousands of years, but most were discovered within the past two centuries. Some represent tangible, comprehendible entities that we can relate to. π, for example, tells us that the circumference of the circle is almost three times its diameter. This is a statement that makes sense and can be intuitively grasped by just looking at a circle and a diameter (see Fig. 1).

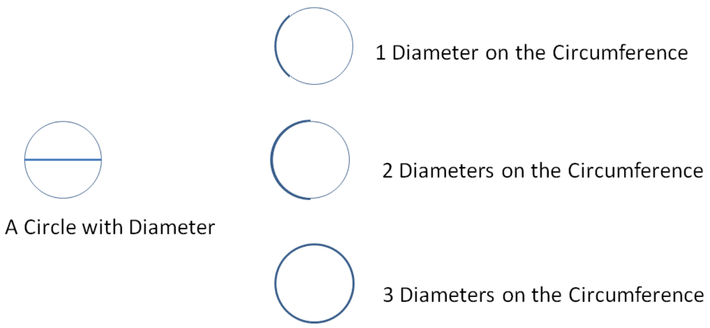

Fig. 1. Three Diameters + make a circumference.

Other constants, however, are abstract entities that have no intuitive or tangible reference. G, the gravity constant, is equal to $6.674 \times 10^{-11} \frac{m^3}{\mathrm{kg.s^2}}$, a very imperceptible value that makes no observational sense; it has no intuitive feel to it. Inspecting G closely does not get us any closer to understanding how the wheels of gravity turn. We know the value of this entity and we have tested its accuracy repeatedly and over decades, proving again and again its validity. Still, gravity remains an obscure abstract notion; scientific curiosity and philosophical arguments continue to rage on in the hope of understanding the true mechanics of the concept.

What is not a debate, however, is that these constants are an invaluable part of science and a triumph for humanity. They help us analytical establish models and mechanisms that allow us to theoretically represent the world around us. Many of the constants are mathematically derived from others. For example, Gelfond's constant is equal to e^{π}. Others are just truly fundamental. They seem to be an intrinsic part of the universe and it almost feels like they are pre-defined values set by a designer before construction. Some, however, can start as the latter (truly fundamental) but become the earlier (derived). For example, the mass of a Helium atom could have been thought of as a constant at one point in time. Now we know a Helium atom has two Protons and two Neutrons making it less fundamental than we thought. That's the idea behind this paper: to search diligently through the heaps of constants sprinkled across the sciences, looking for patterns that connect these constants together. We aim to bring constants from multiple fields together and then apply a simple computation to discover relationships between these constants, hoping to excavate hidden or undiscovered connections. Basically, we are trying to do the lazy work; instead of understanding the complex theory behind such connections, we are just going to look for these relationships without paying attention to theory.

The motivation of the work was the physics Standard Model [1], which describes the fundamental particles and forces that make up the known matter in the universe. The Standard Model contains a few classes of particles (Quarks, Leptons and Bosons) each with a number of particles (for example six types of Quarks: *up*, *down*, *charm*, *strange*, *top*, and *bottom*). Together these particles explain all the elements in the universe and the forces that act on them (at least three out of the four we know). If you look down the hierarchy from top, you will see that material around us are made of molecules which are an assembled structure of one or more elements. One step down the hierarchy and you see that all 118 elements of the periodic table are made of a smaller set of particles (neutrons, protons, electrons) some more fundamental than others. For example, an electron seems to be a fundamental entity (at least as of now), but a proton is made up of more fundamental particles called Quarks. What we wanted to do in this work is bring in all information we know about each particle (mass and charge for example) into a set, add all other constants we know in science and randomly and exhaustively look for relationships. The work in this paper does not address the physics part (investigation on that is still ongoing), but it was an offshoot of that research. Basically as we tackled the physics world, we spun off this math constants focused work that we discuss in this paper [2].

As you can imagine, the computations are iterative and therefore require trillions of mathematical operations, so the output produced in this paper will barely scratch the surface. With much better hardware capacity and better computation algorithms we can greatly improve on the results.

2. Computing π

The pursuit of π has been on the agenda of every math enthusiast since the dawn of science. Babylonians got as far as 3.125. Archimedes found it to be somewhere between 3.1408 and 3.125 [3]. Ptolemy, 139-161 AD, improved on that to find 3.14167 [4]. We now know π to two quadrillion digits; that's 2 with 15 zeros next to it. If you write this π value on a long string and stretch it into space (Star-Wars credits style) it will fill up 7 roundtrips to the moon!

As noble as the pursuit of π digits is, this is not what we address in this paper. What we set out to find is how to represent as many digits of π as we can using as small a formula as we can. We know that π is transcendental, which means there are no combinations of numbers, fractions or algebraic operations that can represent π. What we will do is try to find reasonable approximations of π in terms of basic numbers. The calculations we use in this paper are simple, but more complex computations might unravel better representations.

2.1. The Method

We took all prime numbers less than 1000 and the natural logarithms of these numbers and we computed all possible combinations of the following expressions:

$$\text{RandomComputation} = a^{p1}.\, b^{p2}$$

$$\text{RandomOutput} = a^{p1}.\, b^{p2}.\, c^{p3}$$

$$\text{RandomComputation} = a^{p1}.\, b^{p2}.\, c^{p3}.\, d^{p4}$$

where *a*, *b* and *c* and *d* are values picked from the constants in our list and *p1*, *p2*, *p3*, and *p4* iterate through different sets (sometimes we used a simple set like {-2, -1, -0.5, 0, 0.5, 1, 2}. Other runs were on sets containing all the integers between -10 and 10.) We wrote a Python program to iterate exhaustively through the list and compute the expressions.

Each *RandomComputation* was then compared to π and those within 10^{-6} accuracy were saved. In addition, a few numbers deemed of value were added to the list (like whole integers between 1 and 100), but the results of those were not included in this paper.

The program ran over days circling though all the numbers in the list. It was implemented in Python and a snippet from the code is show in Fig. 2. The run time of the program was very long so in order to speed up the process, multiple instances were run on two public clouds -- Amazon Web Services (AWS) and Google Cloud Platform (GCP) -- and a local instance was run on a gaming computer.

```python
cm.loadBasicMathConstants()
cm.loadAdvancedMathConstants()
#cm.loadPhysicsConstants()
cm.loadPrimeNumbers()

print ("number of Variables: %d" %(len(cm.listOfConstants)))
getcontext().prec = 50

Found = []
equationFound = []
matchesAtThisLevel = 0
equationNumber = 0
level0matches = 1

computedItemsArray = [dict() for x in range(len(cm.listOfConstants))]
makeComputations()
for m in loopRange:
    for n in loopRange:
        for p in  loopRange:
            twoVar(m,n, Decimal(math.e), -1, 1)
```

Fig. 2. Snippet of code from the Python program.

2.2. Findings: Accuracy

We found the usual known rational approximations like $\frac{22}{7}, \frac{33}{106}, \frac{355}{113}$ and $\frac{\sqrt{987}}{10}$. We also found many more. We chose two values in this paper that looked interesting. Each had a sizeable accuracy albeit not easily readable.

$$\pi = \frac{\sqrt{959587}}{(\ln(881))^3}$$ Accuracy=2.87x10-11

$$\pi = \frac{1277^3 . \ln(347)}{1577^3}$$ Accuracy=5.27x10-10

$$\pi = \left(\frac{8545}{4821}\right)^2$$ Accuracy=1.11x10-8

```
pi = (ln(2351))^-3*(ln(3469))^0.5*(ln(3023))^3
Accuracy=-2.4596817193554913424677589627311836590 57E-10

pi = 1571^-3*(ln(347))^1*1277^3
Accuracy=5.2730080536375766054746842155729044295 34E-10

pi = (ln(3673))^-2*(ln(1523))^-0.5*(ln(4049))^3
Accuracy=-1.2694540585904643074391398387059476829 35E-10

pi = (ln(1117))^-2*(ln(479))^0.5*(ln(2677))^2
Accuracy=4.1591587878758616322655109116246990352 3E-11

pi = (ln(2551))^-2*(ln(3929))^0.5*(ln(3631))^2
Accuracy=-3.9616257299756329505153753578954352018 91E-10
```

Fig. 3. π to 10 decimal digits.

We also found other values that are as accurate but perhaps not as read-able; some of these are shown in Fig. 3.

2.3. Findings: Esthetics

In addition, we found some interesting and pleasing to read values of π. The one that catches attention most is:

$$\pi = \frac{7^7}{8^6}$$

Which approximately is equal to 3.14157. We also found an equivalent one:

$$\pi = \frac{7^7}{4^9}$$

A few more made the headlines and are worth mentioning:

$$\pi = \left(\frac{11111}{1234}\right)^2 . \frac{1}{\sqrt{666}}$$

$$\pi = \frac{\sqrt{222 \times 2222222}}{7070}$$

$$\pi = 36. \sqrt{\frac{777}{102030}}$$

$$\pi = \frac{123456789}{1234^2 . \sqrt{666}}$$

$$\pi = \left(\frac{123}{22}\right)^2 . \frac{1}{\sqrt{99}}$$

Which is also equivalent to:

$$\pi = \left(\frac{123}{22}\right)^2 . \frac{\sqrt{11}}{33}$$

$$\pi = \frac{111\sqrt{8899}}{3333}$$

$$\pi = \frac{11\sqrt{1234}}{123}$$

This is the tip of the iceberg. Much better accuracy and more interesting results can be obtained by:

1. Employing more equations with more sophistication. The current implementation is a very simple multiplicative equation. More complex equations can potentially deliver more accuracy.
2. Expanding the set of constants. In this paper we only used primes and natural logarithms of primes. A bigger and more comprehensive constants set might yield better results
3. Introducing AI/ML capability to allow for feedback from each iteration. The idea is to use the output of every computation to fine tune the input to the next, potentially reducing the number of iterations used exponentially.

It is our opinion that less complex and more accurate π representations are possible.

2.4. More Accuracy

We also modified the above algorithm in order to find more accurate representations of π. We ran the algorithm mentioned in the previous section and compared each output to π and we then picked the ones that are within the accuracy needed (10^{-4}). For each result found (foundResult), we took the residue (π – foundResult) and ran the approximation algorithm again on that residue. Then we repeated for as much accuracy as needed. Fig. 4 shows an approximation of π to one hundred digits.

$$\pi = \frac{\sqrt{137}\times\sqrt{173}}{49} - \frac{23\times113\times10^{-4}}{31^2} + \frac{113\times197\times10^{-8}}{107^2} + \frac{\sqrt{61}\times29\times10^{-12}}{137}$$
$$+ \frac{\sqrt{109}\times47^2\times10^{-17}}{109^2} - \frac{\sqrt{167}\times79^2\times10^{-22}}{139^2} - \frac{\sqrt{53}\times89\times10^{-27}}{181} -$$
$$\frac{\sqrt{131}\times7\times10^{-31}}{61} + \frac{\sqrt{71}\times4\times10^{-36}}{\sqrt{13}} + \frac{\sqrt{197}\times83\times10^{-40}}{167} + \frac{\sqrt{59}\times79^2\times10^{-44}}{83^2}$$
$$- \frac{\sqrt{13}\times\sqrt{97}\times10^{-48}}{\sqrt{157}} - \frac{\sqrt{113}\times31\times10^{-52}}{173} - \frac{\sqrt{53}\times179\times10^{-57}}{193} +$$
$$\frac{\sqrt{2}\times\sqrt{137}\times10^{-61}}{\sqrt{191}} - \frac{17\times181\times10^{-66}}{31^2} - \frac{\sqrt{41}\times\sqrt{137}\times10^{-70}}{67}$$
$$+ \frac{\sqrt{11}\times49\times10^{-74}}{157} + \frac{17^2\times10^{-79}}{\sqrt{2}\times25} + \frac{\sqrt{197}\times41^2\times10^{-83}}{149^2} +$$
$$\frac{\sqrt{67}\times\sqrt{157}\times10^{-88}}{49} - \frac{\sqrt{2}\times\sqrt{151}\times10^{-92}}{\sqrt{149}} + \frac{\sqrt{197}\times73^2\times10^{-97}}{101^2} -$$
$$\frac{\sqrt{13}\times173^2\times10^{-101}}{193^2}$$

Fig. 4. π to one hundred digits.

The one thousand digits representation was longer than desired as shown below:

π =137^0.5*173^0.5*7^-2 - (23^1*113^1*31^-2)*(10^-4) +(113^1*197^1*107^-2)*(10^-8) +(61^0.5*29^1*137^-1)*(10^-12) +(109^0.5*47^2*109^-2)*(10^-17) - (167^0.5*79^2*139^-2)*(10^-22) - (53^0.5*89^1*181^-1)*(10^-27) - (131^0.5*7^1*61^-1)*(10^-31) +(13^-0.5*71^0.5*2^2)*(10^-36) +(197^0.5*83^1*167^-1)*(10^-40) +(59^0.5*79^2*83^-2)*(10^-44) - (13^0.5*97^0.5*157^-0.5)*(10^-48) - (113^0.5*31^1*173^-1)*(10^-52) - (53^0.5*179^1*193^-1)*(10^-57) +(2^0.5*137^0.5*191^-0.5)*(10^-61) - (17^1*181^1*31^-2)*(10^-66) - (41^0.5*137^0.5*67^-1)*(10^-70) +(11^0.5*7^2*157^-1)*(10^-74) +(2^-0.5*17^2*5^-2)*(10^-79) +(197^0.5*41^2*149^-2)*(10^-83) +(67^0.5*157^0.5*7^-2)*(10^-88) - (2^0.5*151^0.5*149^-0.5)*(10^-92) +(197^0.5*73^2*101^-2)*(10^-97) - (13^0.5*173^2*193^-2)*(10^-101) +(47^0.5*79^2*127^-2)*(10^-105) +(2^0.5*151^1*167^-1)*(10^-109) +(2^-0.5*79^1*43^-0.5)*(10^-114) - (137^0.5*167^0.5*5^-2)*(10^-118) - (61^0.5*71^2*103^-2)*(10^-122) +(41^0.5*113^2*149^-2)*(10^-127) - (11^1*151^1*29^-2)*(10^-131) - (89^0.5*151^2*181^-2)*(10^-136) +(29^-0.5*59^0.5*3^1)*(10^-140) -

(83^1*37^2*179^-2)*(10^-144) - (151^0.5*79^1*29^-2)*(10^-148) - (13^0.5*31^2*23^-2)*(10^-153) -
(19^0.5*7^1*197^-0.5)*(10^-157) +(13^2*13^2*97^-2)*(10^-162) - (41^0.5*41^2*83^-2)*(10^-166)
+(107^0.5*11^1*97^-1)*(10^-170) +(47^0.5*137^0.5*31^-1)*(10^-175) +(11^0.5*47^1*151^-1)*(10^-179)
- (13^-0.5*113^1*53^-0.5)*(10^-184) +(23^0.5*113^0.5*149^-0.5)*(10^-188) +(47^0.5*61^2*103^-2)*(10^-
192) +(13^0.5*29^0.5*181^-0.5)*(10^-196) - (101^0.5*19^2*59^-2)*(10^-200) - (43^1*29^2*73^-2)*(10^-
206) - (193^-0.5*2^1*11^1)*(10^-210) +(109^0.5*7^2*197^-1)*(10^-215) +(23^-0.5*13^2*193^-0.5)*(10^-
219) +(139^1*3^2*179^-1)*(10^-224) +(113^0.5*3^2*23^-1)*(10^-228) +(101^0.5*113^0.5*47^-1)*(10^-
232) - (163^0.5*5^2*149^-1)*(10^-236) +(61^0.5*103^2*197^-2)*(10^-240) +(3^0.5*101^1*109^-1)*(10^-
244) - (101^0.5*11^1*151^-0.5)*(10^-249) - (3^2*3^2*67^-1)*(10^-253) +(113^1*163^1*97^-2)*(10^-258)
+(43^0.5*103^2*191^-2)*(10^-262) - (97^0.5*127^1*139^-1)*(10^-267) - (11^0.5*139^0.5*103^-
0.5)*(10^-271) - (59^-0.5*79^1*13^-0.5)*(10^-275) +(7^0.5*163^2*167^-2)*(10^-279) -
(11^0.5*109^2*67^-2)*(10^-284) - (157^1*17^2*89^-2)*(10^-288) - (53^0.5*103^2*139^-2)*(10^-292) -
(13^0.5*83^2*79^-2)*(10^-296) - (163^0.5*191^1*31^-2)*(10^-300) - (3^0.5*179^1*131^-1)*(10^-304)
+(53^1*13^2*83^-2)*(10^-308) +(43^0.5*23^1*29^-1)*(10^-313) - (107^0.5*7^2*193^-1)*(10^-319) -
(5^0.5*43^1*109^-0.5)*(10^-324) - (193^1*31^2*157^-2)*(10^-328) - (151^-0.5*5^2*3^1)*(10^-332)
+(41^0.5*23^1*31^-1)*(10^-336) - (7^0.5*31^0.5*37^-0.5)*(10^-340) - (67^1*167^1*79^-2)*(10^-344) -
(197^0.5*19^2*71^-2)*(10^-349) +(17^0.5*197^2*131^-2)*(10^-354) +(31^1*79^2*173^-2)*(10^-358)
+(61^0.5*43^1*109^-1)*(10^-362) +(7^2*83^2*191^-2)*(10^-367) +(61^0.5*163^0.5*7^-2)*(10^-371)
+(43^-0.5*107^1*59^-0.5)*(10^-376) +(29^0.5*197^1*127^-1)*(10^-381) +(193^0.5*5^1*103^-0.5)*(10^-
385) +(101^0.5*3^2*149^-0.5)*(10^-390) +(67^0.5*3^2*101^-0.5)*(10^-394) - (31^0.5*5^1*53^-0.5)*(10^-
398) +(137^-0.5*137^2*179^-1)*(10^-403) +(181^0.5*7^2*181^-1)*(10^-407) - (31^0.5*59^2*101^-
2)*(10^-411) - (173^0.5*3^2*71^-1)*(10^-415) - (31^-0.5*137^1*17^-0.5)*(10^-420)
+(131^0.5*97^1*197^-1)*(10^-424) +(71^1*137^1*59^-2)*(10^-428) +(3^0.5*19^1*73^-0.5)*(10^-433)
+(107^1*193^1*139^-2)*(10^-437) - (83^1*179^1*67^-2)*(10^-442) - (79^-0.5*103^1*97^-0.5)*(10^-446)
+(109^1*181^1*101^-2)*(10^-451) - (29^0.5*31^1*67^-1)*(10^-456) - (3^0.5*181^2*167^-2)*(10^-460) -
(197^0.5*181^1*41^-2)*(10^-464) +(137^0.5*139^1*197^-1)*(10^-469) +(29^0.5*167^1*151^-1)*(10^-473)
- (29^0.5*179^0.5*167^-0.5)*(10^-477) - (67^0.5*149^2*157^-2)*(10^-482) - (193^0.5*79^2*113^-
2)*(10^-486) - (17^0.5*13^1*97^-0.5)*(10^-491) - (61^0.5*103^0.5*19^-1)*(10^-495)
+(13^0.5*61^1*197^-1)*(10^-499) +(181^0.5*107^2*137^-2)*(10^-504) - (103^1*43^2*157^-2)*(10^-508)
- (3^0.5*29^1*191^-0.5)*(10^-512) +(181^0.5*193^0.5*131^-1)*(10^-516) - (179^1*11^2*103^-2)*(10^-
521) - (11^1*89^1*23^-2)*(10^-525) +(103^0.5*167^0.5*37^-1)*(10^-530) - (13^2*13^2*163^-2)*(10^-
534) - (89^0.5*73^2*113^-2)*(10^-539) +(11^0.5*47^0.5*173^-0.5)*(10^-543) - (131^0.5*5^2*149^-
1)*(10^-548) - (17^0.5*137^2*197^-2)*(10^-553) - (23^0.5*31^1*107^-1)*(10^-558) - (157^0.5*41^1*67^-
1)*(10^-564) - (5^0.5*17^1*139^-0.5)*(10^-568) - (131^0.5*89^2*127^-2)*(10^-574) +(3^1*173^1*163^-
1)*(10^-578) - (151^0.5*197^0.5*139^-1)*(10^-582) +(131^0.5*13^2*197^-1)*(10^-587) -
(157^0.5*53^2*107^-2)*(10^-591) +(139^1*173^1*109^-2)*(10^-595) - (19^1*73^2*139^-2)*(10^-601)
+(53^0.5*41^1*107^-1)*(10^-605) +(71^0.5*73^2*181^-2)*(10^-609) +(19^1*73^2*101^-2)*(10^-614) -
(163^0.5*37^1*61^-1)*(10^-618) +(5^0.5*11^2*103^-1)*(10^-622) - (5^0.5*191^2*107^-2)*(10^-628) -
(73^0.5*83^2*97^-2)*(10^-632) - (13^0.5*3^1*53^-0.5)*(10^-636) - (191^-0.5*43^2*11^-2)*(10^-640) -
(2^0.5*113^2*131^-2)*(10^-644) +(167^0.5*41^1*53^-1)*(10^-649) +(191^0.5*37^1*53^-1)*(10^-653) -
(151^0.5*59^2*73^-2)*(10^-657) +(173^0.5*37^2*61^-2)*(10^-661) - (167^0.5*179^1*23^-2)*(10^-665) -
(3^1*139^1*109^-1)*(10^-669) - (131^0.5*53^2*103^-2)*(10^-673) - (61^-0.5*41^2*197^-1)*(10^-677)
+(79^0.5*7^2*53^-1)*(10^-682) - (59^1*37^2*107^-2)*(10^-687) - (7^0.5*151^1*11^-2)*(10^-691) - (29^-
0.5*71^1*31^-0.5)*(10^-695) +(197^-0.5*151^0.5*5^1)*(10^-700) - (5^2*53^2*137^-2)*(10^-704) - (11^-
0.5*103^1*109^-0.5)*(10^-708) +(13^0.5*127^2*167^-2)*(10^-712) +(31^0.5*59^0.5*5^-2)*(10^-716)
+(131^0.5*193^0.5*59^-1)*(10^-721) - (109^0.5*127^1*31^-2)*(10^-725) - (2^0.5*163^0.5*181^-
0.5)*(10^-729) +(89^0.5*137^0.5*107^-1)*(10^-733) - (47^-0.5*11^2*3^-2)*(10^-738)
+(179^1*193^1*73^-2)*(10^-743) +(5^1*47^2*73^-2)*(10^-747) +(179^0.5*107^1*151^-1)*(10^-752) -
(3^0.5*47^1*101^-0.5)*(10^-756) - (173^0.5*17^1*29^-1)*(10^-760) +(59^1*53^2*193^-2)*(10^-764)

+(43^0.5*53^1*179^-1)*(10^-768) +(7^1*37^2*83^-2)*(10^-772) - (23^1*97^2*181^-2)*(10^-777) - (67^0.5*19^2*29^-2)*(10^-781) +(89^-0.5*151^2*29^-2)*(10^-785) +(67^0.5*61^2*113^-2)*(10^-789) - (13^0.5*101^2*151^-2)*(10^-793) +(7^0.5*31^1*109^-0.5)*(10^-798) - (83^-0.5*67^2*13^-2)*(10^-802) +(3^0.5*37^1*31^-1)*(10^-806) +(71^0.5*107^2*181^-2)*(10^-811) - (5^0.5*157^2*103^-2)*(10^-816) - (43^0.5*31^1*113^-1)*(10^-820) - (83^-0.5*167^1*113^-0.5)*(10^-824) +(151^0.5*13^2*43^-2)*(10^-828) +(23^0.5*19^1*181^-0.5)*(10^-833) +(131^-0.5*67^2*103^-1)*(10^-837) +(19^-0.5*43^2*197^-1)*(10^-841) - (67^0.5*193^0.5*83^-1)*(10^-845) +(2^0.5*113^1*139^-1)*(10^-849) +(29^-0.5*157^2*43^-2)*(10^-854) +(127^0.5*29^1*191^-1)*(10^-858) +(7^1*5^2*149^-1)*(10^-862) - (157^-0.5*37^2*13^-1)*(10^-867) +(83^0.5*13^2*157^-1)*(10^-872) +(11^1*89^1*149^-1)*(10^-876) +(23^0.5*173^2*163^-2)*(10^-880) - (67^0.5*103^2*127^-2)*(10^-884) - (101^0.5*193^0.5*53^-1)*(10^-888) - (23^1*73^2*149^-2)*(10^-893) - (127^1*163^1*71^-2)*(10^-898) +(193^0.5*29^2*71^-2)*(10^-902) +(83^1*101^1*67^-2)*(10^-907) +(73^-0.5*2^1*7^1)*(10^-911) +(3^2*3^2*83^-0.5)*(10^-916) - (79^0.5*191^2*197^-2)*(10^-920) - (19^0.5*97^2*113^-2)*(10^-924) - (67^0.5*101^1*181^-1)*(10^-929) - (173^-0.5*151^2*193^-1)*(10^-934) - (3^1*151^1*149^-1)*(10^-938) +(79^0.5*5^1*19^-1)*(10^-942) - (191^-0.5*3^0.5*79^0.5)*(10^-947) - (131^1*13^2*67^-2)*(10^-952) +(11^1*31^1*83^-1)*(10^-956) - (137^1*157^1*89^-2)*(10^-961) - (113^-0.5*13^2*179^-0.5)*(10^-965) +(5^-0.5*11^2*41^-1)*(10^-970) +(5^0.5*103^2*97^-2)*(10^-975) - (151^1*181^1*107^-2)*(10^-979) - (43^0.5*11^2*151^-1)*(10^-984) +(47^0.5*73^2*191^-2)*(10^-988) - (11^1*3^2*139^-0.5)*(10^-993) +(53^1*23^2*67^-2)*(10^-997) - (43^0.5*3^2*23^-1)*(10^-1002) +(163^0.5*179^0.5*157^-1)*(10^-1006) - (37^0.5*191^0.5*103^-0.5)*(10^-1011) - (149^0.5*167^1*41^-2)*(10^-1015) +(173^0.5*11^1*131^-1)*(10^-1019) - (173^-0.5*31^2*71^-1)*(10^-1023) +(151^0.5*5^1*47^-0.5)*(10^-1028) - (2^-0.5*139^1*137^-0.5)*(10^-1032) +(193^-0.5*193^2*31^-2)*(10^-1036) +(43^1*179^1*73^-2)*(10^-1040) +(127^0.5*173^0.5*53^-1)*(10^-1045) - (53^1*41^2*179^-2)*(10^-1049) - (181^1*19^2*197^-2)*(10^-1053) - (167^0.5*11^1*89^-1)*(10^-1057) +(31^0.5*109^1*113^-1)*(10^-1062) - (3^2*5^2*61^-1)*(10^-1067) - (7^0.5*7^2*53^-1)*(10^-1072) +(151^0.5*163^0.5*71^-1)*(10^-1076) - (137^0.5*31^2*89^-2)*(10^-1080) - (61^1*193^1*101^-2)*(10^-1084) - (179^0.5*3^2*151^-0.5)*(10^-1089)

The main reason for the lengthy form was the restrictions on the run-time of the program. The higher we make the accuracy setting, the more compact the form found. We used 10^{-4} because it took more than two days to run one iteration. It would have taken more than a year to run with a 10^{-8} accuracy which would reduce the size of the expression dramatically. Of course more powerful machines can allow us to run with much higher accuracy and therefore produce much more compact representations.

3. Computing Euler's Number

The number e, called Euler's number is the limit of $(1+1/n)^n$ as n tends to infinity and is approximately equal to 2.7182. Euler's number, just like π, is transcendental which means it's not algebraic and therefore cannot be a solution to any polynomial whose coefficients are rational. Euler's number is ubiquitous and, also just like π, is featured elegantly in nature. For example, Normal distribution equation uses Euler's number. It's also the only number whose exponential derivative stays the same:

$$\frac{de^x}{dx}=e^x$$

There is no argument that it's a constant that have great value in Math. It's one of five constants that are featured in what arguably is the most beautiful formula in math, the Euler's identity, $e^{i\pi}+1=0$.

We applied the same algorithms that we did for π to find approximate algebraic expressions for Euler's number and we ran the program exhaustively for prime numbers and natural logarithms applying powers of not only whole integers, but also powers of decimals (not exceeding first decimal point.).

3.1. Findings: Accuracy

A good baseline is from Wolfram's e Approximations page, with accuracy to 7 digits [5]:

$$\mathrm{e} = 3 - \sqrt{\frac{5}{63}}$$

We found some values of e with higher accuracy (10-9) like the following two values:

$$e \approx \frac{433 \ln 461}{977} = 2.7182818$$

$$e \approx \frac{(\ln 29)^3}{\ln 881.\sqrt{\ln 73}} = 2.7182818$$

It's interesting to see ln(881) featured in both e and π representations.

A less accurate and easier to read values were also found like:

$$\mathrm{e} \approx \frac{\left(\frac{7\times 31}{13}\right)^4}{13^4} = 2.7182$$

Both values were obtained on datasets that included primes numbers less than 1000 and powers of 4 or less.

3.2. Findings: Esthetics

Also some nice looking but less accurate values were found for e. The most interesting ones are:

$$\mathrm{e} \approx \frac{\pi^7}{1111} = 2.718$$

And

$$\mathrm{e} \approx \frac{33333^4}{333^7} = 2.718$$

We have not attempted to compute e to a 1000 digit, but conceptually the same process that was applied to π can be used here.

4. Math Constants

Around 400 constants were used to run this exercises and they range from the popular, like π and e, to the less popular but fairly know in math-literate circles, like Viswanath Constant (1.1319882487943), to the obscure and only known to the experts in that specific domain, like Bloch–Landau Constant (0.54325 89653 42976 70695). This is not a comprehensive list and the results were used just to prove a concept. Much more work can be done here to extend the exercise to a wider data set.

4.1. The Method

Just like with π, we wrote a Python program to iterate exhaustively through the list and compute the following expressions:

$$\text{RandomComputation} = a^{p1}.b^{p2}$$

$$\text{RandomOutput} = a^{p1}.b^{p2}.c^{p3}$$

$$\text{RandomComputation} = a^{p1}.b^{p2}.c^{p3}.d^{p4}$$

However, instead of comparing each output to π, it was compared to all items in the list, which added two orders of magnitude to the size of the computation. We set an accuracy value to be 10-9 which means a match was found if any of the RandomComputations was within 10-9 of any of the values on the list. Basically, we are trying to find out if any of the constants can be obtained by combining three constants from the list, with different powers to each constant, in a simple multiplicative equation.

4.2. The Findings

Fig. 5 shows a summary of the results. In this section, we are going to pick the first result for analysis. Much more analysis can be done on the rest.

pi=17^-0.5*Foias^1*Bronze ratio^2

Wright constant=Sophomore's dream2^0.5*Calabi triangle^2*Primorial^1

resiprocal twin primes=Tetranacci^-0.5*Spiral of Theodorus^-0.5*Infinite Produt Alladi-Grinstead^1

Prévost constant=Bronze ratio^-0.5*Volume of Reuleaux tetrahedron^-2*Raabe^-1

e=Sophomore's dream2^-0.5*Laplace limit^-2*Bloch–Landau constant^-0.5

Somos' quadratic recurrence constant=Murata Constant^-0.5*Erdős–Borwein^-2*Khinchin's constant^2

(ln(3))=Khinchin-Lévy constant^0.5*Somos' quadratic recurrence constant^-1*Zeta(6)^0.5

Golomb–Dickman constant=Upper iterated exponentia^-0.5*Tribonacci constant^-2*Plastic number^2

Strongly Carefree constant=Golomb–Dickman constant^-0.5*Murata Constant^-1*Triangular root of 2^-1

Cahen's constant=Pell^-0.5*Viswanath constant^-1*Silver rootTutte–Beraha ^-0.5

Lower límit iterated exponential=Khinchin harmonic^-0.5*Salem Number^-1*Magic angle^-0.5

Omega constant=(ln(5))^-0.5*Exp.gamma, Barnes G-function^-1*Lebesgue constant L2^0.5

Euler–Gompertz constant=Exp.gamma, Barnes G-function^0.5*Triangular root of 2^-1*Continued fraction constant, Bessel function^1

Murata Constant=Machin–Gregory series^0.5*Landau–Ramanujan constant^-1*Exp.gamma, Barnes G-function^2

Nested radical S5=Lochs^-0.5*Gauss–Kuzmin–Wirsing ^-0.5*Universal parabolic^0.5

Infinite Produt Alladi-Grinstead=Primorial^-0.5*Backhouse's constant^-0.5*Grothendieck constant^1

9.6549037860879326248709038241476023773355483284371E-9

Bronze ratio=Foias^-0.5*Lebesgue^0.5*Brun 2 constant = Σ inverse of Twin primes^2

Alladi–Grinstead constant =Pólya Random walk^-0.5*Foias^0.5*Lower límit iterated exponential^2

Dottie number:=Chi Function^-0.5*Paris Constant^1*Continued fraction constant, Bessel function^2

moving sofa=Khinchin's constant^-0.5*Magic angle^1*(ln(7))^2

Landau–Ramanujan constant=Froda constant^-0.5*Sophomore's dream^2*Komornik–Loreti constant^2

Cubic recurrence constant=Lochs^0.5*Chi Function^-2*Module of Infinite Tetration^2

(ln(5))=Pell^0.5*Connective^-2*Khinchin's constant^2

Bernstein's constant=MRB constant^0.5*Raabe^2*Mills' constant^-1

Fig. 5. Math constants relationships.

The formula in the first result (first line in Fig. 5) is straightforward:

$$\pi \approx \frac{\text{Foias Constant.(Bronze Ratio)}^2}{\sqrt{17}} \tag{1}$$

Foias constant is named after Ciprian Foias who won the Norbert Wiener Prize in Applied Mathematics in 1995. The constant is defined as a real number x_1 such that

$$x_{n+1} = \left(1+\frac{1}{x_n}\right)^n$$

Tends to ∞ as n tends to ∞. The value of the constant is transcendental, just like π and e, and is approximately equal to: 1.187452351…

The Bronze Ratio is equal to $\frac{(3+\sqrt{13})}{2}$, which is around 3.30277563…

Rearranging Equation(1) above we get:

$$\pi \approx \frac{\left[\frac{3+\sqrt{13})}{2}\right]^2 . \text{Foias Constant}}{\sqrt{17}}$$

Or

$$\text{Foias Constant} = \frac{2\pi\sqrt{17}}{11+3\sqrt{3}}$$

The above form for Foias Constant is pretty interesting to read and there is potential to find more interesting relationships if the right computation are performed with the right datasets.

5. Further Research

More work can be done to improve accuracy on computation and find more elegant values for e, π, and many other constants. More importantly, more work will be done on science constants. The first run produced some interesting results for approximations, but a true breakthrough can be achieved if we manage to find exact values. In other words, if we truly find relationships that are not approximations, we could potentially discover fundamental insights on the behavior of these constants and their true meanings. One roadblock we faced is the wide range of numbers. Some of the numbers we looked at are infinitesimally small, like gravity constant G, or Planck's constant h. Others were tens of orders of magnitude bigger, like the speed of light, c. Comparing these entities can never produce precise results. We now know that we need to bucket each set of constants according to their size and apply the algorithms on each set separately.

Another idea worth pursuing is introducing Artificial Intelligence and Machine Learning to the computations. When we run the data exhaustively, and without any positive feedback after every loop, we are quickly faced with prohibitively large number of computations. For example, when we only had 100 numbers in the set, and we ran the programs with powers of 1 and 2 only, we ended up with a large number of computations (around 1.6 billion computations). When the number of constants was increased to 1000 and the number of powers was increased to 20, the number of computations became 8 quadrillion. It will therefore make sense to intelligently choose which computations to run by incorporating feedback from every loop directly into the new one.

6. Conclusion

A simple mathematical computation is used to exhaustively look for simple relationships between a set of constants. The constants span multiple fields but the focus of the paper was on math constants. A simple Python program was used to do these calculations and some very interesting values of both π and e were found like $\pi = \frac{123456789}{1234^2.\sqrt{666}}$, $\pi \approx \frac{7^7}{8^6}$ and $e \approx \frac{33333^4}{333^7}$. We also were able to compute π to one thousand digits by going through residue from every computation iteratively.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] The CERN Physics online. *The Standard Model*. Retrieved from http://www.home.cern/science/physics/standard-model

[2] Wikipedia. *List of Mathematical Constants*. Retrieved from http://www.en.wikipedia.org/wiki/List_of_mathematical_constants

[3] Strogatz, S. (2013). *The Joy of X* (pp. 126-127). Boston: Mariner Books.
[4] Becomann. (1971). *A History of π* (pp. 74-75). New York: St. Martin's Press.
[5] Wolfram Math World. *e Approximations.* Retrieved from http://www.mathworld.wolfram.com/eApproximations.html

Tamara Azzam was born on June 17, 2004 in Texas and is currently an 11th grade student at Uplift North Hills Preparatory. In school Tamara focuses on math and physics and plans to have a career in one of these fields. Tamara's areas of interest include math, technology, physics, music, and computer gaming.

Ms. Azzam is a member of the school band, and before COVID-19 was the lead flutist in the Lone Star Youth Orchestra. She made the regional band and qualified for state in multiple competitions. She currently lives with her family in Irving, Texas.