UNIVERSITY OF SRI JAYEWARDENEPURA

FINAL YEAR PROJECT-B.SC.(HONS) MATHEMATICS

Generalization of Trigonometric, Hyperbolic and Fibonacci-Lucas identities with Symbolic Computations

Gayanath Chandrasena

Faculty of Applied Sciences

Supervised by

Dr. Jayantha Senadheera

Department of Mathematics

12 August 2022

Declaration

I, MGS Chandrasena, declare that this dissertation is my own original work unless otherwise referenced or acknowledged and the work described in this report was carried out by me under the supervision of Dr. Jayantha Senadheera.This report has not been submitted in whole or in part to any university/institution for another qualification.

Name:-..

Index Number:-..

Signature:-...

Date:-..

Certification

I hereby certify that the dissertation entitled "Generalization of Trigonometric, Hyperbolic and Fibonacci-Lucas Identities with Symbolic Computations" submitted in partial fulfillment for the BSc(Hons) degree in Mathematics,was prepared under my supervision by MGS Chandrasena(AS2017333).

Name:-..

Signature:-...

Date:-..

Acknowledgments

I would like to acknowledge and give my warmest thanks to my supervisor, Dr. Jayantha Senadheera who made the completion of this work possible. His advices and guidance carried me through all the stages of writing and doing my project.

And also I would like to thank Dr. Prabath Silva, My second supervisor for helping me throughout this project.

I am also grateful to Mr.Sarath Kumara, Head of the Department of Mathematics as well as the Department of Mathematics, University of Sri jayewardenepura for giving me this valuable opportunity.

Lastly, I would be remiss in not mentioning my family whose belief in me has kept my spirits and motivation high during this process.

Contents

Abstract

Using a kind of symbolic generalization of Fibonacci-Lucas' numbers and sine, cosine functions, we try to bridge two areas. It seems all of this generalization reduces to five fundamental equations. Using these five fundamental equations we derive identities in the symbolic generalization. By converting these identities into Fibonacci-Lucas sequences and trigonometry, we derive meaningful identities in those areas. Using SymPy library of python we developed two programs to generate the identities in symbolic generalization.

1 Introduction

Number sequences and trigonometry are two different areas in mathematics. Fibonacci and Lucas' sequences are well known sequences which are closely related. Again cosine function and sin function are closely related periodic functions in trigonometry. In this study using a "symbolic method" we are bridging the two areas. We call this generalization as "Symbolic Trigonometry".

John Conway pointed out this idea in his article in *mathematical gazette* and also Jayantha Senadheera developed this idea and used it to solve some problems in Elementary problem session and Advanced problem session in *Fibonacci Quarterly*. Basically this project is based on those studies.

Our main objectives were to use the definitions of Symbolic Trigonometry to generalize some trigonometric, hyperbolic and Fibonacci-Lucas' identities and use python to build computer program to establish trigonometric, hyperbolic and Fibonacci-Lucas' identities. Apart from the theoretical aspect we have built a computer program using python, to compute and establish the identities in these areas.

In this study we have developed some symbolic proofs and symbolic computations on this symbolic generalization.We have built two separate computer programs for two structures, trigonometric structure and Fibonacci-Lucas' structure. In future we hope to unify those two separate programs into one single program.

The main purpose of this study is to establish some identities in Symbolic Trigonometry and convert them into as Fibonaccci-Lucas' identities and as trigonometric identities.In future one can extend this study for establishing more advanced identities.

1.1 Fibonacci Sequence

"Fibonacci Sequence" is consisted of fibonacci numbers[1] where fibonacci numbers are starting from 0 and 1. Generally n^{th} term of the fibonacci sequence is denoted by F_n.

$$F_0 = 0$$

$$F_1 = 1$$

The n^{th} term of the fibonacci sequence can be obtained by the recurrence relation,

$$F_n = F_{n-1} + F_{n-2} \qquad for\, n \geq 2$$

1.2 Lucas Sequence

"Lucas Sequence"[1] is defined using the same recurrence relation as Fibonacci sequence but with different initial values.The n^{th} term of Lucas' sequence is denoted by L_n.

$$L_0 = 2$$

$$L_1 = 1$$

$$L_n = L_{n-1} + L_{n-2} \qquad for\, n \geq 2$$

.

1.3 Closed Form Formula(Binet Formula)

Even though one can find the n^{th} term of the Fibonacci sequence or Lucas sequence using the recurrence relations, one disadvantage is, one should know the previous two terms to do so. To overcome this disadvantage it can be used the closed form formula called the Binet formula.

$$F_n = \frac{\alpha^n - \beta^n}{\sqrt{5}}$$

$$L_n = \alpha^n + \beta^n$$

where α=$\frac{1+\sqrt{5}}{2}$ and β=$\frac{1-\sqrt{5}}{2}$.

So by substituting values for n it can be found any Fibonacci number or Lucas number.

This closed form expression first established by Jacques Philippe Marie Binet in 1843.

Lets establish the closed form formula using the recurrence relations.

The characteristic equation for the recurrence relations of Fibonacci and Lucas' equations is

$$x^2 - x - 1 = 0$$

And the roots were $\frac{1 \pm \sqrt{5}}{2}$.
Since the roots are real and distinct n^{th} term of the Fibonacci sequence is of the form $a\alpha^n$+$b\beta^n$ and n^{th} term of the Lucas' sequence is of the form $c\alpha^n$+$d\beta^n$, where a, b, c, d are constants and α and β are roots.
By substituting the initial conditions F_0=0, F_1=1 and solving the simultaneous equations we get $a = \frac{1}{\sqrt{5}}$ and $b = -\frac{1}{\sqrt{5}}$.
Then

$$F_n = \frac{1}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^n - \frac{1}{\sqrt{5}}(\frac{1-\sqrt{5}}{2})^n$$

$$F_n = \frac{\alpha^n - \beta^n}{\sqrt{5}}$$

In a similar way one can find the closed form formula for Lucas' numbers too.

1.4 Euler's formula for $Sin\theta$ and $Cos\theta$

Euler's formula, which is named after Leonard Euler, is a formula which gives the relationship between trigonometric functions and exponential functions. Usually it is denoted by,

$$e^{i\theta} = cos\theta + isin\theta$$

where e is the base of the natural logarithm. By simple calculations we can establish two formulas for $sin\theta$ and $cos\theta$.

$$sin\theta = \frac{e^{i\theta} - e^{-i\theta}}{2i} \quad and \quad cos\theta = \frac{e^{i\theta} + e^{-i\theta}}{2}$$

Hyperbolic sine and hyperbolic cosine functions[4] are denoted as $sinh\theta$ and $cosh\theta$ respectively, which are defined as.

$$sinh\theta = \frac{e^{\theta} - e^{-\theta}}{2} \quad and \quad cosh\theta = \frac{e^{\theta} + e^{-\theta}}{2}$$

2 Literature Review

"Symbolic Trigonometry" is not a very eluded area since only few researches or projects are done about it. In this study, I have dug deeper to strengthen the bridge between trigonometry and Fibonacci-Lucas number sequences. I referred past studies which are related to my study.

This study is mainly based on the article "Fibonometry" by John Conway in *Mathematical Gazette* and the problems and solutions given in the Elementary and Advanced Problems sessions in Fibonacci Quarterly. Specially the solution published as *"An Unusual Generalization"* by Jayantha Senadheera.

In [2] Conway says that "Every trigonometric identity relating sums of products of sines and cosines also gives a corresponding identity in which sines and cosines are replaced by Fibonacci and Lucas numbers, respectively. Only the constants are changed".This statement leads to find some further identities and relations between these two areas. Further he has linked some identities in Fibonacci-Lucas number sequences and trigonometry.

In [5], author has solved problems using the symbolic trigonometry approach. This method facilitate not only solve the problems but also generalize the problems. More examples in this context will be discussed in this thesis.

3 Generalization(Symbolic Trigonometry)

Symbolic Generalization is the most important part in this study. This generalization establish a bridge between two structures, trigonometry and Fibonacci-Lucas numbers. We named this procedure as "*Symbolic Trigonometry*".

Let's define two symbolic sequences as follows.

$$\mathbf{u_n} = \mathbf{x^n} - \frac{\varepsilon^{\mathbf{n}}}{\mathbf{x^n}} \quad \textit{and} \quad \mathbf{v_n} = \mathbf{x^n} + \frac{\varepsilon^{\mathbf{n}}}{\mathbf{x^n}}$$

where $\varepsilon \in \{-1, 1\}$ and $n \in N$.

Why we can call this as a "Generalization"?

Simply because these symbolic definitions paving the path to *sine* and *cosine* functions as well as to Fibonacci-Lucas' sequences. By using above mentioned Binet formula and Euler's formula the task can be easily done.

- case 1-Trigonometry

  Substitute x=e$^{i\theta}$ $\quad and \quad \varepsilon = 1$
  $u_n = 2isin(n\theta) \quad and \quad v_n = 2cos(n\theta)$

- case 2-Fibonacci-Lucas

  Substitute x=$\frac{1+\sqrt{5}}{2}$ $\quad$ and $\quad \varepsilon = -1$
  $u_n = \sqrt{5}F_n \quad and \quad v_n = L_n$

Why above generalization is useful? From this, we can prove the identities in each field in more symmetrical way. Also we can convert identities in one field to another field using this symbolic bridging. Moreover this symbolic approach facilitate to establish the identities using symbolic computations.

Straight forward calculations lead to following results.

By substituting $n = 0$,

1.
$$\begin{aligned} u_0 &= x^0 - \frac{\varepsilon^0}{x^0} \\ &= 1 - 1 \\ &= 0 \end{aligned}$$

$$\begin{aligned} v_0 &= x^0 + \frac{\varepsilon^0}{x^0} \\ &= 1 + 1 \\ &= 2 \end{aligned}$$

2.
$$\begin{aligned} u_{-n} &= x^{-n} - \frac{\varepsilon^{-n}}{x^{-n}} \\ &= -\varepsilon^n u_n \end{aligned}$$

$$\begin{aligned} v_{-n} &= x^{-n} + \frac{\varepsilon^{-n}}{x^{-n}} \\ &= \varepsilon^n v_n \end{aligned}$$

3.

$$u_m.u_n = (x^m - \frac{\varepsilon^m}{x^m})(x^n + \frac{\varepsilon^n}{x^n})$$

$$= x^{m+n} + \frac{\varepsilon^n}{x^{n-m}} - \frac{\varepsilon^m}{x^{n-m}} - \frac{\varepsilon^{m+n}}{x^{m+n}}$$

$$= v_{m+n} - \varepsilon^n.(x^{m-n} + \frac{\varepsilon^{m-n}}{x^{m-n}})$$

$$= v_{m+n} - \varepsilon^n v_{m-n}$$

4.

$$v_m.v_n = (x^m + \frac{\varepsilon^m}{x^m})(x^n + \frac{\varepsilon^n}{x^n})$$

$$= x^{m+n} + \frac{\varepsilon^n}{x^{n-m}} + \frac{\varepsilon^m}{x^{m-n}} + \frac{\varepsilon^{m+n}}{x^{m+n}}$$

$$= v_{m+n} + \varepsilon^n.(x^{m-n} + \frac{\varepsilon^{m-n}}{x^{m-n}})$$

$$= v_{m+n} + \varepsilon^n v_{m-n}$$

5.

$$u_n.v_m = (x^n - \frac{\varepsilon^n}{x^n})(x^m + \frac{\varepsilon^m}{x^m})$$

$$= x^{m+n} + \frac{\varepsilon^m}{x^{m-n}} - \frac{\varepsilon^n}{x^{n-m}} - \frac{\varepsilon^{m+n}}{x^{m+n}}$$

$$= u_{m+n} - \varepsilon^n.(x^{m-n} - \frac{\varepsilon^{m-n}}{x^{m-n}})$$

$$= u_{m+n} - \varepsilon^n u_{m-n}$$

We consider these as the main five identities in Symbolic Trigonometry. We will use above identities for calculations in Trigonometric structure and Fibonacci-Lucas Structure of symbolic computations.

3.1 Conversion to Trigonometry

As I mentioned above, in case 1, by substitution to x and ε, we achieve the following.

we substitute

$$x = e^{i\theta} \quad and \quad \varepsilon = 1$$

Then we obtain

$$u_n = 2i.sin(n\theta) \qquad\qquad v_n = 2cos(n\theta)$$

1.

$$u_0 = 2isin(0) = 2i.0 = 0$$

$$u_0 = 0$$

$$v_0 = 2cos(0) = 2.1 = 2$$

$$v_0 = 2$$

2.

$$u_{-n} = 2i.sin(-n\theta)$$

$$= -2i.sin(n\theta)$$

$$= -2i.sin(n\theta)$$

$$v_{-n} = 2cos(-n\theta)$$

$$= 2cos(n\theta)$$

$$= 2cos(n\theta)$$

3. Now since u_n=2i.sin(nθ), v_n=2cos(nθ), u_m=2i.sin(mθ) and v_m=2cos(mθ))

$$u_m.u_n = v_{m+n} - v_{m-n}$$

$$\begin{aligned} \text{u}_m.u_n &= 2i.sin(m\theta).\text{2i.sin(n}\theta) \\ &= -4sin(m\theta)\text{sin(m}\theta) \end{aligned} \tag{1}$$

$$\begin{aligned} \text{v}_{m+n}\text{-}v_{m-n} &= 2cos((m+n)\theta)\text{-2cos((m-n)}\theta) \\ &= -4sin(m\theta)\text{sin(m}\theta) \end{aligned} \tag{2}$$

By (1) and (2),

$$2sin(n\theta).sin(m\theta) = cos((m+n)\theta) - cos((m-n)\theta)$$

4.

$$v_m.v_n = v_{m+n} + v_{m-n}$$

$$\begin{aligned} \text{v}_m.v_n &= 2cos(m\theta).\text{2cos(n}\theta) \\ &= 4cos(m\theta)\text{cos(m}\theta) \end{aligned} \tag{3}$$

$$\begin{aligned} \text{v}_{m+n}\text{+}v_{m-n} &= 2cos((m+n)\theta)\text{+2cos((m-n)}\theta) \\ &= 2[2cos(m\theta)\text{cos(n}\theta)]\text{=4cos(m}\theta\text{)cos(n}\theta) \end{aligned} \tag{4}$$

again by (3) and (4),

$$2cos(n\theta).cos(m\theta) = cos((m+n)\theta) + cos((m-n)\theta)$$

5.

$$u_n.v_m = u_{m+n} - u_{m-n}$$

$$\begin{aligned} \mathrm{u}_n.v_m &= 2i.sin(n\theta).2\cos(\mathrm{m}\theta) \\ &= 4i.sin(n\theta)\cos(\mathrm{m}\theta) \end{aligned} \tag{5}$$

$$\begin{aligned} \mathrm{v}_{m+n}+v_{m-n} &= 2cos((m+n)\theta)+2\cos((\mathrm{m}-\mathrm{n})\theta) \\ &= 2[2cos(m\theta)\cos(\mathrm{n}\theta)]=4\cos(\mathrm{m}\theta)\cos(\mathrm{n}\theta) \end{aligned} \tag{6}$$

By (5) and (6),

$$2i.sin(n\theta).cos(m\theta) = cos((m+n)\theta) + cos((m-n)\theta)$$

3.2 Conversion to Fibonacci-Lucas' Sequences

According to case 2 mentioned above in this chapter, by substituting $x = \frac{1+\sqrt{5}}{2}$ and ε=-1 we get the following.

$$u_n = \sqrt{5}F_n \qquad\qquad v_n = L_n$$

1.
$$u_0 = \sqrt{5}.F_0 = 0$$
$$u_0 = 0$$

$$v_0 = L_0 = 2$$
$$v_0 = 2$$

2.
$$u_{-n} = \sqrt{5}.F_{-n}$$
$$v_{-n} = L_{-n}$$

3.
$$u_m.u_n = v_{m+n} - (-1)^n v_{m-n}$$
$$u_m.u_n = 5F_mF_n \tag{7}$$

$$v_{m+n} - (-1)^n v_{m-n} = L_{m+n} + (-1^n)L_{m-n} \tag{8}$$

By (7) and (8),
$$5F_mF_n = L_{m+n} + (-1^n)L_{m-n}$$

4.
$$v_m.v_n = v_{m+n} + (-1)^n v_{m-n}$$

$$v_m.v_n = L_m L_n \tag{9}$$

$$v_{m+n} + (-1)^n v_{m-n} = L_{m+n} - (-1^n) L_{m-n} \tag{10}$$

By (9) and (10),

$$L_m L_n = L_{m+n} - (-1^n) L_{m-n}$$

5.

$$u_n.v_m = u_{m+n} - (-1)^n u_{m-n}$$

$$u_n.v_m = \sqrt{5} F_n L_m \tag{11}$$

$$u_{m+n} - (-1)^n u_{m-n} = \sqrt{5} F_{m+n} - (-1^n)\sqrt{5} F_{m-n} \tag{12}$$

By (11) and (12)

$$F_n L_m = F_{m+n} - (-1^n) F_{m-n}$$

3.3 Theorem

$$u_{a_1}^{\alpha_1}.u_{a_2}^{\alpha_2}...u_{a_n}^{\alpha_n}.v_{b_1}^{\beta_1}.v_{b_2}^{\beta_2}...v_{b_m}^{\beta_m} = \begin{cases} c_1v_{i_1}+c_2v_{i_2}+...+c_kv_{i_k} & if\ \sum_{h=1}^{n}\alpha_h\ is\ even \\ d_1u_{j_1}+d_2u_{j_2}+...+d_lv_{j_l} & if\ \sum_{h=1}^{n}\alpha_h\ is\ odd \end{cases}$$

where $\alpha_1,\alpha_2,...,\alpha_n,\beta_1,\beta_2,...,\beta_m \in \mathbb{N},\ a_1,a_2,...,a_n,b_1,b_2,...,b_m \in \mathbb{Z}$

$c_1,c_2,...c_k,d_1,d_2,...,d_l \in \mathbb{Z},\ i_1,i_2,...,i_k,j_1,j_2,...,j_l \in \mathbb{N}\cup\{0\}$

proof

Recall our main identities in the *Generalization*..

$$u_0 = 0,\ v_0 = 2$$

$$u_{-n} = -\varepsilon^n u_n,\ v_{-n} = \varepsilon^n v_n$$

$$u_m.u_n = v_{m+n} - \varepsilon^n v_{m-n}$$

$$v_m.v_n = v_{m+n} + \varepsilon^n v_{m-n}$$

$$u_n.v_m = u_{m+n} - \varepsilon^n u_{m-n}$$

Observe that from the fourth equation, if the product has any ***v*** terms, product of those ***v*** terms reduces to linear combination of ***v*** terms. From the second equation if the ***v*** terms have negative subscripts those can be transformed into positive subscripts. Now if the product has even number of ***u*** terms, from the third and fourth equations product of those even number of ***u*** terms reduce to linear combination of ***v*** terms. Finally from the last equation it is clear that if the product consist of odd number of ***u*** terms then the final answer reduces to linear combination of ***u*** terms. As in the earlier argument all the negative subscripts can be reduced to positive subscripts. Hence the theorem.

Here are some of the numerical examples I have computed using the computer code I mentioned in the chapter 5.This is this code is for $\varepsilon = 1$, in other words for the trigonometric case.

$$\begin{array}{ll}
\mathrm{u}_1 = u_1 & v_1 = v_1 \\
u_1^2 = v_2 - 2 & v_1^2 = v_2 + 2 \\
u_1^3 = -3u_1 + u_3 & v_1^3 = 3v_1 + v_3 \\
u_1^4 = -4v_2 + v_4 + 6 & v_1^4 = 4v_2 + v_4 + 6 \\
u_1^5 = 10u_1 - 5u_3 + u_5 & v_1^5 = 10v_1 + 5v_3 + v_5 \\
u_1^6 = 15v_2 - 6v_4 + v_6 - 20 & v_1^6 = 15v_2 + 6v_4 + v_6 + 20
\end{array}$$

Furthermore Consider the following examples.

$$\begin{array}{ll}
u_1 v_1 = u_2 & u_1 v_1 = u_2 \\
u_1^2 v_1 = -v_1 + v_3 & u_1 v_1^2 = u_1 + u_3 \\
u_1^3 v_1 = -2u_2 + u_4 & u_1 v_1^3 = 2u_2 + u_4 \\
u_1^4 v_1 = 2v_1 - 3v_3 + v_5 & u_1 v_1^4 = 2u_1 + 3u_3 + u_5
\end{array}$$

Observe that by substituting $U_n = 2i.sin(n\theta)$ and $v_n = 2cos(n\theta)$ above identities transform into familiar trigonometric identities[5].

Now lets observe some examples computed using the second computer code in chapter 5. This is this code is for $\varepsilon = -1$, in other words for the Fibonacci-Lucas' case.

$$\begin{aligned} \mathrm{u}_1 &= u_1 & v_1 &= v_1 \\ u_1^2 &= v_2 + 2 & v_1^2 &= v_2 - 2 \\ u_1^3 &= 3u_1 + u_3 & v_1^3 &= -3v_1 + v_3 \\ u_1^4 &= 4v_2 + v_4 + 6 & v_1^4 &= -4v_2 + v_4 + 6 \\ u_1^5 &= 10u_1 + 5u_3 + u_5 & v_1^5 &= -10v_1 - 5v_3 + v_5 \\ u_1^6 &= 15v_2 + 6v_4 + v_6 + 20 & v_1^6 &= 15v_2 - 6v_4 + v_6 - 20 \end{aligned}$$

Furthermore Consider the following examples.

$$\begin{aligned} u_1v_1 &= u_2 & u_1v_1 &= u_2 \\ u_1^2v_1 &= v_1 + v_3 & u_1v_1^2 &= -u_1 + u_3 \\ u_1^3v_1 &= 2u_2 + u_4 & u_1v_1^3 &= -2u_2 + u_4 \\ u_1^4v_1 &= 2v_1 + 3v_3 + v_5 & u_1v_1^4 &= 2u_1 - 3u_3 + u_5 \end{aligned}$$

3.4 Symbolic Proofs

3.4.1 Symbolic proof for $sin3\theta$

Generally from our trigonometric knowledge we know that

$$sin3\theta = 3sin\theta - 4sin^3\theta$$

But we will try to approch this from our symbolic aspect.

consider u_1^3. Note that for the trigonometric case $\varepsilon = 1$. By applying the third and fifth formulas of the fundamental equations, we get

$$u_1^3 = u_3 - 3u_1$$

By substituting u_n=2i.sin(nθ) for $n = 1$ and $n = 3$ we get

$$\begin{aligned} -8i.sin3\theta &= 2i.sin3\theta - 6i.sin\theta \\ sin3\theta &= 3sin\theta - 4sin^3\theta \end{aligned}$$

Hence proof is complete.

3.4.2 Symbolic proof for Cassini formula

Let's consider the "Cassini formula" which is related to Fibonacci numbers. Cassini formula is given by,

$$F_{n-1}.F_{n+1} - F_n^2 = (-1)^n$$

Consider $\frac{1}{5}(u_{n-1}u_{n+1} - u_n u_n)$.

$$\begin{aligned}
&= \frac{1}{5}(u_{n-1}u_{n+1} - u_n u_n) \\
&= \frac{1}{5}[v_{2n} - (-1)^n v_2 - v - 2n + (-1)^n v - 0] \\
&= \frac{1}{5}(-1)^n [v_2 + v_0] \\
&= \frac{1}{5}(-1)^n [3 - 2] \\
&= (-1)^n
\end{aligned}$$

using our symbolic definitions and substituting to the left side of the above equation($u_n = \sqrt{5}F_n$ we get,

$$F_{n-1}.F_{n+1} - F_n^2 = (-1)^n$$

Hence the proof is complete.

From above example we can observe that not only the Cassini formula of Fibonacci numbers can be derived but also a generalization of it can be verified.

4 Computational Examples

In this chapter we will discuss some of the basic computational problems and the identities they imply.

Recall our main identities in the *Generalization*..

$$u_0 = 0, \;\; v_0 = 2$$

$$u_{-n} = -\varepsilon^n u_n, \;\; v_{-n} = \varepsilon^n v_n$$

$$u_m.u_n = v_{m+n} - \varepsilon^n v_{m-n}$$

$$v_m.v_n = v_{m+n} + \varepsilon^n v_{m-n}$$

$$u_n.v_m = u_{m+n} - \varepsilon^n u_{m-n}$$

(a)

Symbolic Trigonometry Identity

$$u_n^2 + v_n^2 = 2v_{2n}$$

By third and fourth equations $u_n.u_n = v_{n+n} - \varepsilon^n v_{n-n}$. Then $u_n^2 = v_{2n} - \varepsilon^n v_0$. Similarly $v_n^2 = v_{2n} + \varepsilon^n v_0$. By adding these two equations we get $2v_{2n}$.

Trigonometric Identity

By substituting $u_1 = 2i.sin\theta$, $v_1 = 2cos\theta$, $v_2 = 2cos2\theta$ we have the trigonometric identity $cos^2\theta - sin^2\theta = cos(2\theta)$.

Fibonacci-Lucas' Identity

By substituting u_n=$\sqrt{5}F_n$, $v_n = L_n$, $v_{2n} = L_{2n}$ we have the Fibonacci-Lucas' identity $5F_n^2 + L_n^2 = 2L_{2n}$.

(b)

Symbolic Trigonometry Identity

$$u_n v_n = u_{2n}$$

Trigonometric Identity

By substituting $u_1 = 2i.sin\theta$, $v_1 = 2cos\theta$, $v_2 = 2cos2\theta$ we have the trigonometric identity $sin2\theta = 2sin\theta.cos\theta$

Fibonacci-Lucas' Identity

By substituting u_n=$\sqrt{5}F_n$, $v_n = L_n$, $v_{2n} = L_{2n}$ we have the Fibonacci-Lucas' identity $F_n L_n = F_{2n}$

(c)

Symbolic Trigonometry Identity

$$u_1^3 = u_3 - 3\varepsilon u_1$$

Trigonometric Identity

By substituting $u_1 = 2i.sin\theta$, $u_3 = 2sin3\theta$ we have the trigonometric identity $sin3\theta = 3sin\theta - 4sin^3\theta$

Fibonacci-Lucas' Identity

By substituting u_n=$\sqrt{5}F_n$, $F_{3n} = F_{3n}$ we have the Fibonacci-Lucas' identity $5F_n^3 = F_{3n} + 3F_n$

(d)

Symbolic Trigonometry Identity

$$v_{2n}^2 = v_{4n} - 2$$

Trigonometric Identity

By substituting $v_2 = 2cos2\theta$, $v_4 = 2cos4\theta$ we have the trigonometric identity $cos4\theta = 2cos^2 2\theta - 2$

Fibonacci-Lucas' Identity

By substituting $v_{2n} = L_{2n}, v_{4n} = L_{4n}$ we have the Fibonacci-Lucas' identity $L_{2n}^2 = L_{4n} - 2$

5 Computations with Python

5.1 Python-SymPy

It can be observed that this symbolic approach facilitate to build a computer program for the computations. In my research I have used a Computer Algebra System(CAS) library called **Sympy**, a library used in python programming language for symbolic computations. Which is basically built on top of python.

SymPy is an open-source Python library which provides computer algebra capabilities either as a standalone application, as a library to other applications, or live on the web as SymPy Live.Wikipedia[6] says SymPy includes features ranging from basic symbolic arithmetic to calculus, algebra, discrete mathematics, and quantum physics.

In my research I have built two separate computer programs for the cases of $\varepsilon = 1$ and $\varepsilon = -1$. But in future I hope to unify those two into a single program.

5.2 Sympy Code for Trigonometry Structure ($\varepsilon = 1$)

$$u_n = x^n - \frac{1}{x^n} \qquad v_n = x^n + \frac{1}{x^n}$$

```
# importing sympy library
import sympy as sp
# defining symbols
x, n, k = sp.symbols("x n k")




# defining 2 functions u(n) and v(n)
def u(n):
    return x**n -(x**-n)

def v(n):
    return x**n + (x**-n)




# defining function/expression
def function(expr):
    expansion = sp.expand(expr)
    if isinstance(expansion, sp.Pow):
raise Exception("Failed to expand given expression:", expansion)

    # analyse all the powers of x available
    defaulters = []
    terms_with_powers = [[],[]]
    twp_constants = [[],[]]

    output = sp.S.Zero

    if expansion.is_constant():
        args = [expansion]
    else:
        args = expansion.args
    for i in args:
        var = 0 if i.subs({x:0}) == 0 else 1
        if i.is_constant():
            output += i
        elif isinstance(i, sp.Mul) and not
        isinstance
        (i/i.subs({x:1}), sp.Pow):
```

```
        terms_with_powers
        [var].append(1 if i.subs({x:0})
        == sp.S.Zero else -1)
            twp_constants[var].append(i.subs({x:1}))
        elif isinstance(i, sp.Symbol):
            terms_with_powers[var].append(1)
            twp_constants[var].append(1)
        elif isinstance(i, sp.Pow):
            terms_with_powers[var].append(i.args[1])
            twp_constants[var].append(1)
        elif isinstance(i, sp.Mul):
            constant = i.subs({x:1})
            terms_with_powers[var].append((i/constant).args[1])
            twp_constants[var].append(constant)

        else:
            defaulters.append(i)
    if defaulters != []:
raise Exception("Failed to comprehend these terms:" + str(defaulters))

    # start applying relations available

    used_powers = []
    for i in terms_with_powers[0]:
        if -i in terms_with_powers[1]:
     used_powers.append(abs(i))
     positive = twp_constants[0][terms_with_powers[0].index(i)]
     negative = twp_constants[1][terms_with_powers[1].index(-i)]
         if positive == negative:
        output += positive*sp.symbols('V_'+str(i))
      elif positive == -negative:
                output += positive*sp.symbols('U_'+str(i))
    if set(used_powers) != set(terms_with_powers[0]):
message = "All terms could not be converted to
U_n and V_n
\n terms_with_powers: " +\
str(terms_with_powers)+"\n used_powers:
"+str(used_powers)
        raise Exception(message)

    return output

# Enter your input here in the form of u(2)**3*v(2)**2
input_expr = "u(1)*u(2)*u(4)*u(9)"


# Output of the fucntion is stored in answer
answer = function(eval(input_expr, {'u':u, 'v':v,}))
print("input expression:", input_expr, "\n\nOutput:", answer)
```

5.3 Sympy Code for Fibonacci-Lucas' Structure ($\varepsilon = -1$)

$$u_n = x^n - \frac{(-1)^n}{x^n} \qquad v_n = x^n + \frac{(-1)^n}{x^n}$$

```
import sympy as sp
x, n, k = sp.symbols("x n k")

def u(n):
    return x**n - (-1**n)*x**-n

def v(n):
    return x**n + (-1**n)*x**-n

def function(expr):
    expansion = sp.expand(expr)
    if isinstance(expansion, sp.Pow):
        raise Exception("Failed to expand given expression:",
        expansion)

    # analyse all the powers of x available
    defaulters = []
    terms_with_powers = [[],[]]
    twp_constants = [[],[]]

    output = sp.S.Zero

    if expansion.is_constant():
        args = [expansion]
```

```
else:
    args = expansion.args
for i in args:
    var = 0 if i.subs({x:0}) == 0 else 1
    if i.is_constant():
        output += i
    elif isinstance(i, sp.Mul)
    and not
    \n isinstance(i/i.subs({x:1}), sp.Pow):
    terms_with_powers[var].append
    (1 if i.subs({x:0}) == sp.S.Zero else -1)
    twp_constants[var].append(i.subs({x:1}))
    elif isinstance(i, sp.Symbol):
        terms_with_powers[var].append(1)
        twp_constants[var].append(1)
    elif isinstance(i, sp.Pow):
        terms_with_powers[var].append(i.args[1])
        twp_constants[var].append(1)
    elif isinstance(i, sp.Mul):
        constant = i.subs({x:1})
 terms_with_powers[var].append((i/constant).args[1])
 twp_constants[var].append(constant)

    else:
        defaulters.append(i)
if defaulters != []:
    raise Exception("Failed to comprehend these terms:
    " + str(defaulters))

# start applying relations available

used_powers = []
```

```
    for i in terms_with_powers[0]:
        if -i in terms_with_powers[1]:
            used_powers.append(abs(i))
            positive = twp_constants[0][terms_with_powers[0].index(i)]
            negative = twp_constants[1][terms_with_powers[1].index(-i)]
            if positive == negative:
                output += positive*sp.symbols('V_'+str(i))
            elif positive == -negative:
                output += positive*sp.symbols('U_'+str(i))
    if set(used_powers) != set(terms_with_powers[0]):
        message =
        "All terms could not be converted to
        U_n and V_n
        \n terms_with_powers: " +\
        str(terms_with_powers)+"\n used_powers: "+str(used_powers)
        raise Exception(message)

    return output

# Enter your input here in the form of u(2)**3*v(2)**2
input_expr = " "

# This code gets the output and presents it.
# Output of the fucntion is stored in answer.
answer = function(eval(input_expr, {'u':u, 'v':v}))
print("input expression:", input_expr, "\n\nOutput:", answer)
```

6 References

[1] Burton, D., 2010. EBOOK: Elementary Number Theory. McGraw Hill.

[2] Conway, J. and Ryba, A., 2013. 97.39 Fibonometry. The Mathematical Gazette, 97(540), pp.494-495.

[3] Loney, S.L., 1895. Plane trigonometry.

[4] Rudin, W., 1976. Principles of mathematical analysis (Vol. 3). New York: McGraw-hill.

[5] Senadheera, J., 2021. An Unusual Generalization. Fibonacci Quarterly, [online] 59(4). Available at: <https://www.fq.math.ca/Problems/ElemProbSolnNov2021.pdf> [Accessed 9 August 2022].