Programing in the Wolfram Language

O Fakinlede & O Adewumi
University of Lagos

Relevancy of today's tutorial class

- By a special request of your lecturer, an esteemed colleague, I am to lead this tutorial on programming.
  - There are two main platforms we have recommended on our three-year course in Continuum Mechanics. These are Mathematica and Fusion 360.
  - Of course, you can learn Continuum Mechanics without using the computer just as you can get from here to Ibadan without using a vehicle.
  - If the kidnappers don't see you as a worthy target, and no hungry animal finds you a sufficiently delicious lunch, you may get there.
- I am assuming you want the modern, efficient way to learn. We will provide it if you cooperate by obeying simple instructions.

Mechanical Engineering, What is it?

- Any engineering combines several skill sets to create a scientist specially equipped to orchestrate technology products and services.
  - The disciplines are not as distinct as you may be thinking. Modern technology mediates to blur the distinctions even further. There is more done by robots today than five year ago. Increasingly, some skills will diminish in value while those who embrace modern technology will keep finding themselves to be in demand.
  - The choice to become a workshop dinosaur or a modern engineer is yours to make! We are here to help you, if you allow us!

Fusion 360 & Mathematica

- If ALL you get from University of Lagos is proficiency in these two skill areas, you will never be unemployed! I can take your bet on that! Just obey simple instructions, you will KNOW how to program
  - If you waste your time here, and forfeit the opportunities we are giving you, those who pay attention may pity you and give some handouts later in life.
  - You may not remember to tell your unfortunate children that similar opportunities were yours to refuse!
  - I leave it at that!
- Let us take the dive into the wonderful world of programming with Mathematica and the Wolfram Language

Mathematica as a Calculator

Programming is the art of instructing the computer to do as you wish.

Mathematica is a special environment, using the Wolfram Language, that is exceedingly effective in instructing the computer to do many things in an engineer's wish list.

It is futile to pretend you are learning to program by reading a book. Programming is an art, it is pattern matching, it can ONLY be successfully learned by doing it!

To Program is to Do

- “Those who can, Do
  Those who cannot, Teach
  Those who cannot teach, Administrate
  If you fail in administration, go to Politics!”
  GB Shaw (Paraphrased)
- Make your mistakes, Learn from them, Copy other people’s work, Amend to your taste, Understand, … , You are programming!
- “Learn the rules like a pro, so you can break them like an artist.” Pablo Picaso

You already know Mathematica!

If you have ever successfully used a calculator, then you already know how to use Mathematica.

Just remember the following:

- All commands in Mathematica use the same words you are used to, capitalized first letters and spelled out in the most familiar way.
- You execute your command by holding the Shift key as you press enter.
- The entire session is saved and each command is saved

What is a Function?

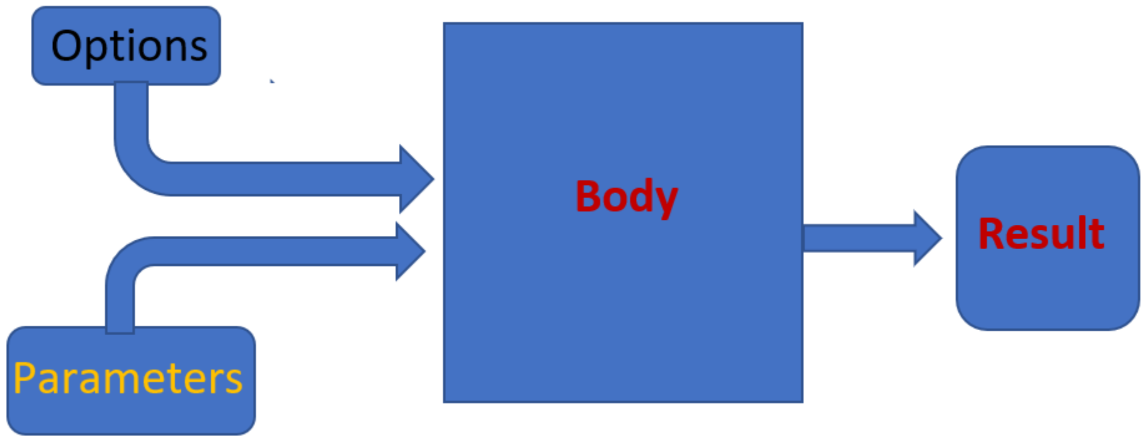
The Fundamental Construct in Mathematica is the Function.

- You can do a lot without fully understanding what a function is, and how it works.
- You will be like a Nigerian living in China for ten years and never learning the language. You are on a bus, you don't understand what jokes are being cracked at you; you see a road sign, you have no idea what it means. In fact, you can carry your death warrant to your executioner since you don't even know what you are carrying!

We will look at what a Function is in the context of Mathematica.

What is a Function?

- Usually, a Mathematica Function has a name, a body, a list of parameters with options that you supply to invoke the function just as much as a Sango man invokes Sango!
- The naming of the function with legal parameters in the correct order and all the necessary punctuations and delimiters is the way you invoke the function.
- The result is supplied by Mathematica after you have activated your invocation via the pressing of the enter key while holding Shift.



Symbolic Algebra & Numerical Analysis

- It could come as a shock to those who already know how to program the computer using numerical based compilers.
- In Mathematica, a variable does not have to be defined before usage.
- Remember, Mathematica is capable of Symbolic Algebra as well as Numerical analysis!

```
In[27]:= M = {{a, b}, {c, d}}
Out[27]= {{a, b}, {c, d}}

In[29]:= Inverse[M] // MatrixForm
Out[29]//MatrixForm=
```

$$\begin{pmatrix} \frac{d}{-b c+a d} & -\frac{b}{-b c+a d} \\ -\frac{c}{-b c+a d} & \frac{a}{-b c+a d} \end{pmatrix}$$

```
In[30]:= Det[M]
Out[30]= -b c + a d
```

Input	Interpretation	Comments
Sin [x]	$\sin x$	Ensure to Capitalize first letter and use square brackets
Integrate[a x^2,x]	$\int a\, x^2 dx$	Indefinite integral. It is still necessary to let Mathematica know which variable you are integrating with respect to after the comma.
Integrate[a x^2,{x, 0, 1}]	$\int_0^1 a\, x^2 dx$	Definite integral. The range is a list showing the variable of integration, beginning and end of domain.
Log[x,b]	$\log_x b$	
x y	$x \times y$	The space tells Mathematica you are multiplying the two variables whether you have declared them to be so or not.
TensorProduct[u,v]	$\mathbf{u} \otimes \mathbf{v}$	Tensor Product of two vectors. Mathematica expects the vectors to be defined as a list of numbers.

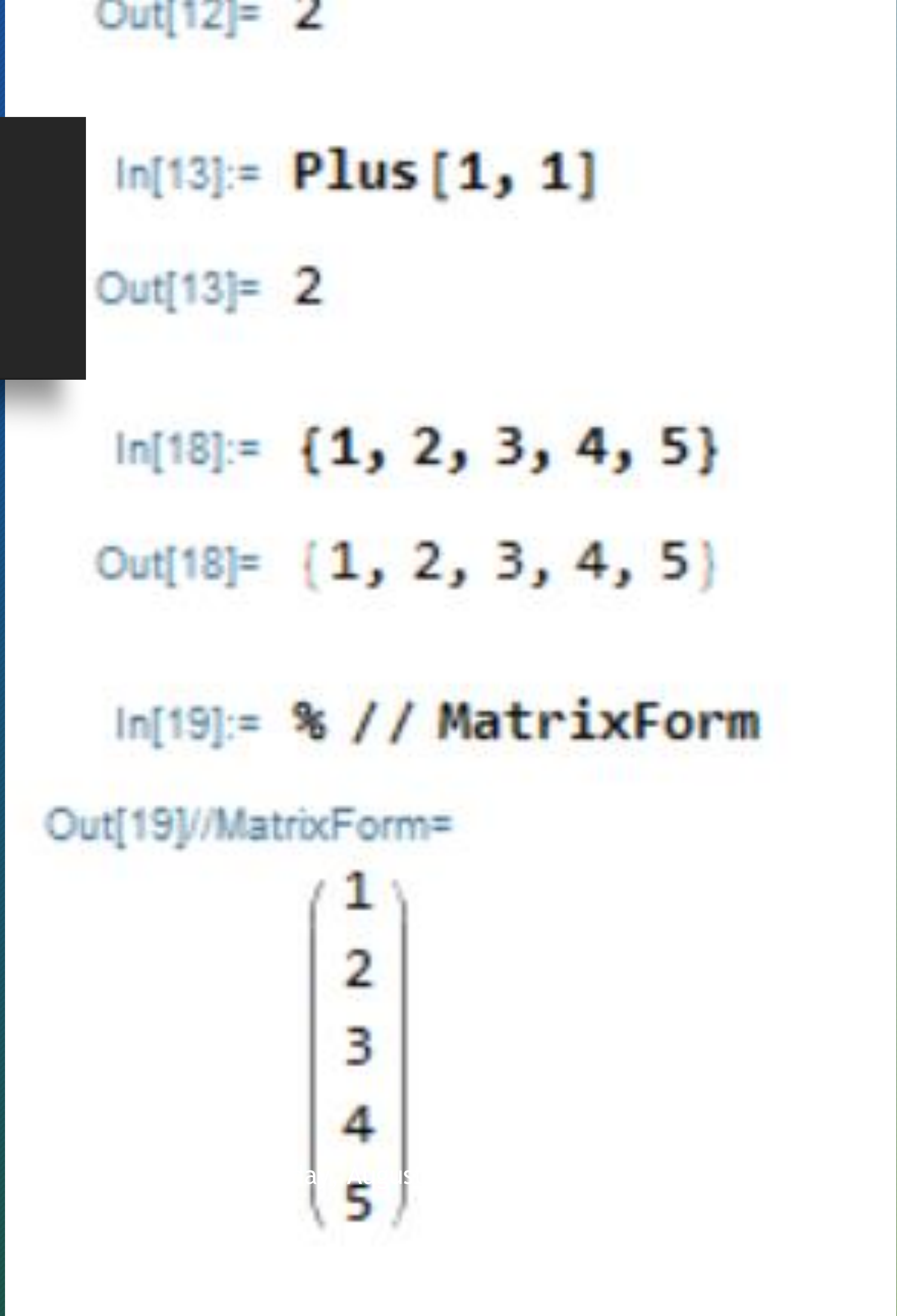
See How it Works

- See the Mathematica Session in the figure.
  - Of course, Mathematica will tolerate it when you type "1+1" like a primary school girl! It grudgingly tells the answer is two!
  - Internally, it is an alias for the invocation of the multi-parameter function Plus[]. I am sure you can see immediately that Times[3,8,-5] will work as you think! Mathematica is logical to the extremes!
  - Similarly, You are allowed to present your lists in the form of elements in braces, as shown. You can even ask it to help you present it in a matrix arrangement by calling the function, MatrixForm[] – here done in a "Postfix" method.
  - Internally, there is a function called List that your braces are simply a shorthand for, as you can see in the next call!

```
In[12]:= 1 + 1
Out[12]= 2

In[13]:= Plus[1, 1]
Out[13]= 2

In[18]:= {1, 2, 3, 4, 5}
Out[18]= {1, 2, 3, 4, 5}

In[19]:= % // MatrixForm
Out[19]//MatrixForm=
  (1)
  (2)
  (3)
  (4)
  (5)

In[20]:= Head[In[18]]
Out[20]= List

In[21]:= List[1, 2, 3, 4, 5]
Out[21]= {1, 2, 3, 4, 5}
```

See How it Works

- See the Mathematica Session in the figure.
  - Furthermore, that Mathematica keeps a good record of all that happens in a session. I requested the “Head” function that is behind the operation in Input 18: In[18]. It responded that the answer is “List”.
  - You are allowed to forget everything! Just ask Mathematica, and it will tell you.
  - The symbol % is for the last operation. %% is the second to the last operation. %n is the nth to the last operation or you can simply invoke directly by using the Input number as I did above.



Functions & Conventions

- Mathematica contains all the elementary functions you are already familiar with:
  - Trigonometric functions, Exponential and Logarithmic Functions, Hyperbolic functions.
  - Perhaps all the special functions you will likely need: Gamma Function, Error function,
  - Its commands and operators including control structures are all available as functions. Before we look at specific examples, observe the important issue of notation:
- ALL built in functions, constructs and structures are functions with capitalized first letters.
  - Consequently, Mathematica does NOT recognize that you want the following trigonometric functions: $\sin x$, $\cos x$, $\arcsin \vartheta$ unless they are properly capitalized

Calling Conventions

- The symbol π, for example, is recognized by its full name with the first letter capitalized. So are all the trig functions in their usual form capitalized.
  - Function arguments in Mathematical are supplied with square brackets, square brackets alone, and nothing but square brackets. The different brackets have specific uses.
  - Mathematica **will not** forgive your sins. You will get PUNISHED each and every time you forget these rules!
- We called the sine function in three different ways: Prefix, Infix and PostFix methods. All functions can be called in the three ways.
  - Lastly, you can generate numerical values to any number of decimals you want.

In[22]:= `x = Pi / 3`

Out[22]= $\frac{\pi}{3}$

In[23]:= `Sin[x]`

Out[23]= $\frac{\sqrt{3}}{2}$

In[24]:= `Sin@x`

Out[24]= $\frac{\sqrt{3}}{2}$

In[25]:= `x // Sin`

Out[25]= $\frac{\sqrt{3}}{2}$

In[26]:= `N[%, 20]`

Out[26]= `0.86602540378443864676`

The Rite of Passage

Your first sign of growth in any computer language is the day you know, that crazy computer people defy the world!

The sign that you have always known to mean Equality “=“ does NOT have the same meaning in most programming languages.

Before I tell you what it means, let me first tell you what you must type if what you want is the “Equality” that you have been carrying in your head since primary school!

In Mathematica, here is the sign that equates: “==“ the double equals. If you use the single one for this purpose, you are on your own!

The Set Symbol “=“


```
In[31]:= Set[a, b]

Out[31]= b

In[32]:= a = b

Out[32]= b
```

- Two statements in the picture are equivalent in Mathematica. In fact, the second, using you familiar equals sign, is the shorthand for the function call, Set[]:

Set (=)

lhs = *rhs*

evaluates *rhs* and assigns the result to be the value of *lhs*. From then on, *lhs* is replaced by *rhs* whenever it appears.

$\{l_1, l_2, \ldots\} = \{r_1, r_2, \ldots\}$

evaluates the r_i, and assigns the results to be the values of the corresponding l_i.

Simple Example: Simultaneous Equations

- In this example, you are in hot water if you mistakenly use the single equality here!
    - Even though the coefficients of your equation are not determined, the solution still comes out!
    - It is necessary to let Mathematica know that you are trying to get the values of x and y. It has no way of knowing as they are not the only things undeclared!
    - The solve function has a set of equations, and a set of variables as parameters or input arguments. These can either be placed in braces or listed as shown:

```
In[1]:= Solve[{a x + b y == 1, x - y == 2}, {x, y}]
```

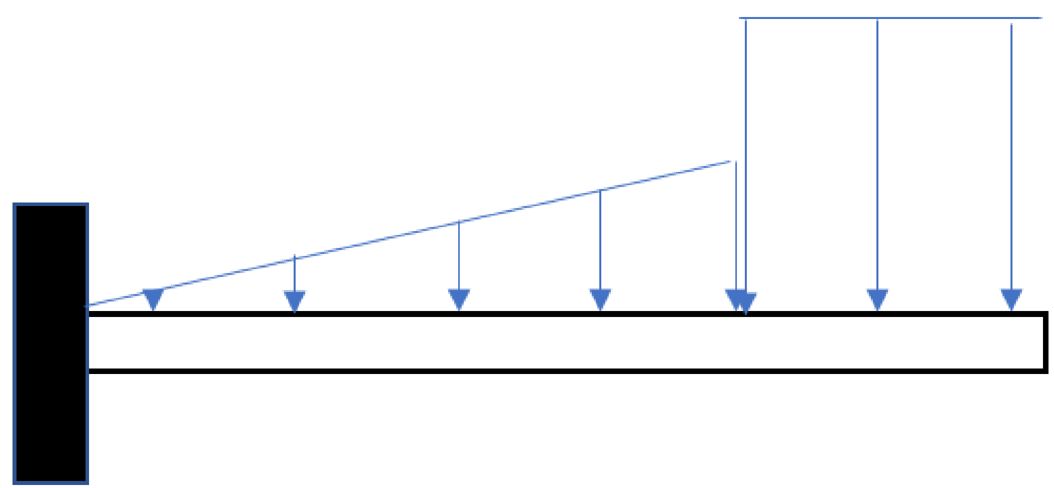
$$\text{Out[1]=}\ \left\{\left\{x \to -\frac{-1-2b}{a+b},\ y \to -\frac{-1+2a}{a+b}\right\}\right\}$$

```
Solve[List[a x + b y == 1, x - y == 2], List[x, y]]
```

$$\left\{\left\{x \to -\frac{-1-2b}{a+b},\ y \to -\frac{-1+2a}{a+b}\right\}\right\}$$

Discontinuous Loading

- Consider the classical Beam Bending problem where the loading is discontinuous.
- Suppose you want to find the Bending moment or the deflection of the elastica using the elementary Strength of Materials methods – a crazy thing to do in 2019 when there is 3D Solid Modeling and Simulation tools.
- With Fourier Series, you can turn this discontinuous load distribution to a continuous one.
- Mathematica takes this from just a theoretical idea to something you can see and feel! Shall we!

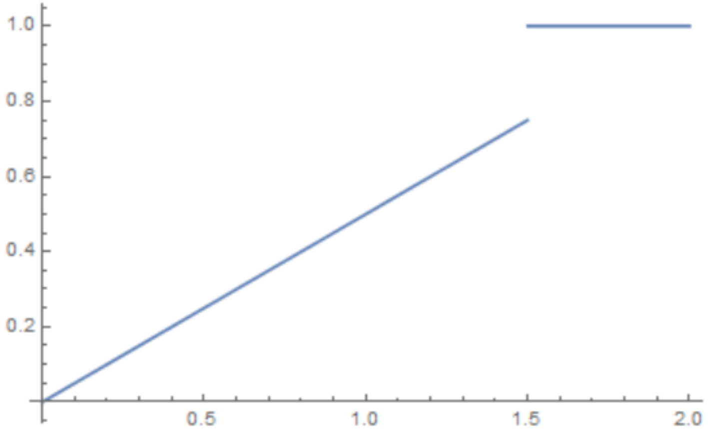


Fourier Method

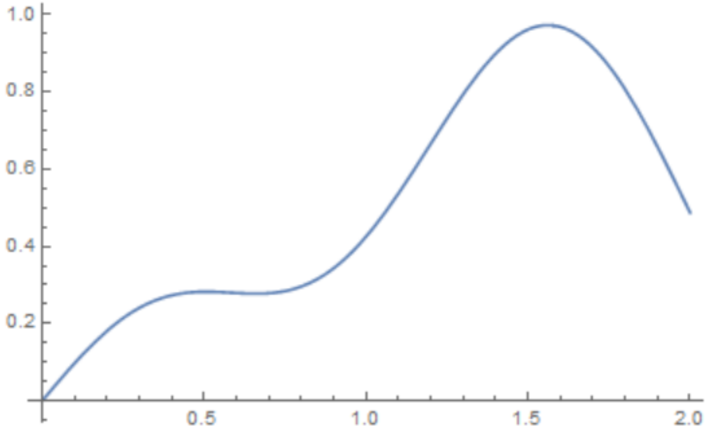
- Forcing a discontinuous function to become continuous!
- We have not done very well, have we? See the function we want to approximate, Compared to what we have!
- Next Step, we will do two things: Take more of the series and place the two curves on top of each other for a better comparison of the approximation.

www.oafak.com, www.s2pafrica.org, oafak@unilag.edu.ng

```
In[4]:= fF[x_] = Piecewise[{{x/2, 0 ≤ x ≤ 1.5}, {1, 1.5 < x ≤ 2}}]

Out[4]= { x/2   0 ≤ x ≤ 1.5
        { 1     1.5 < x ≤ 2
        { 0     True

Plot[fF[x], {x, 0, 2}]
```

Out[ ]=


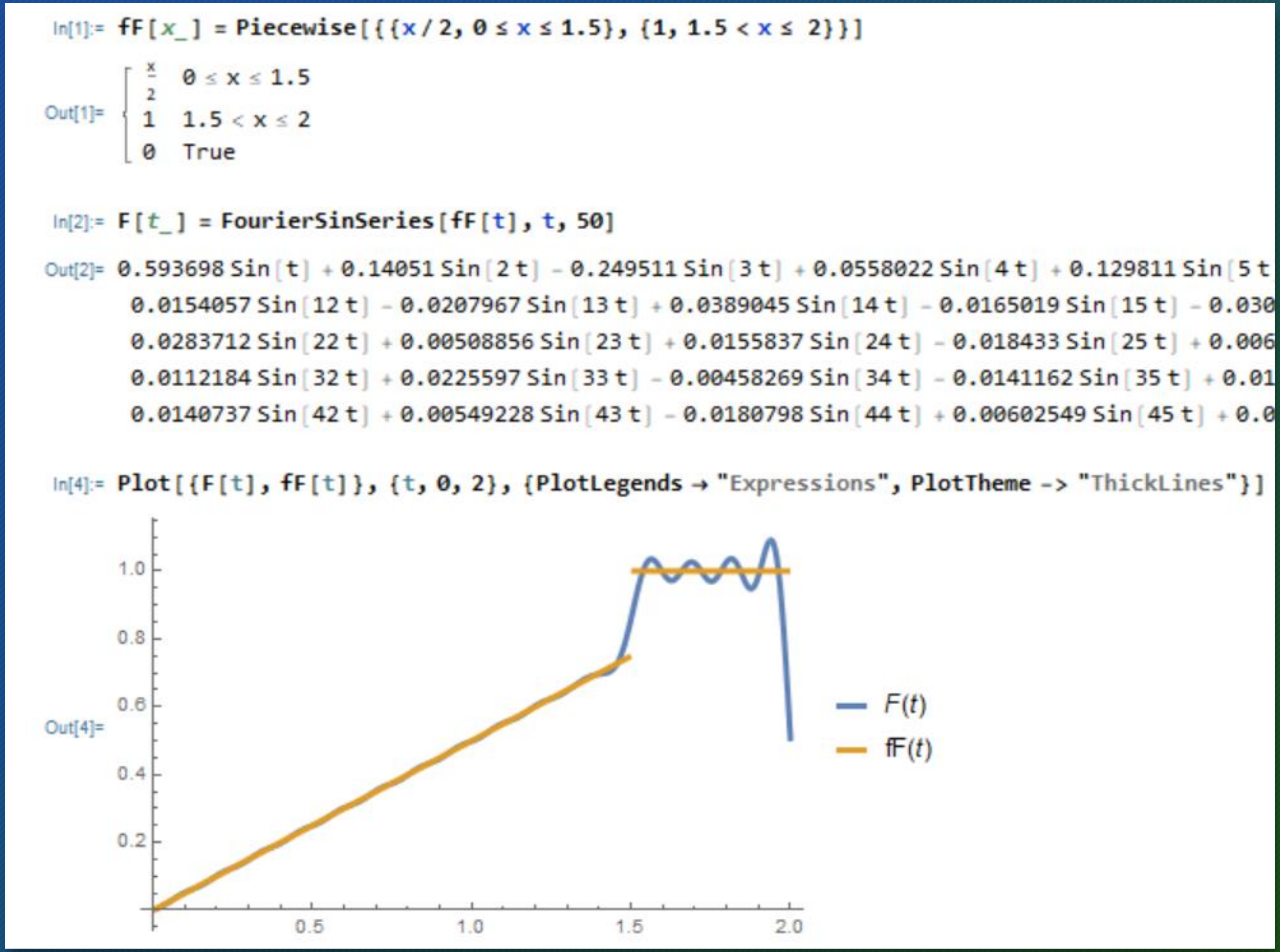

```
In[5]:= F[t_] = FourierSinSeries[fF[t], t, 5]

Out[7]= 0.593698 Sin[t] + 0.14051 Sin[2 t] - 0.249511 Sin[3 t] + 0.0558022 Sin[4 t] + 0.129811 Sin[5 t

In[6]:=
Plot[%, {t, 0, 2}]
```

Out[6]=



```
In[ ]:= F[t_] = FourierSinSeries[fF[t], t, 100]
```

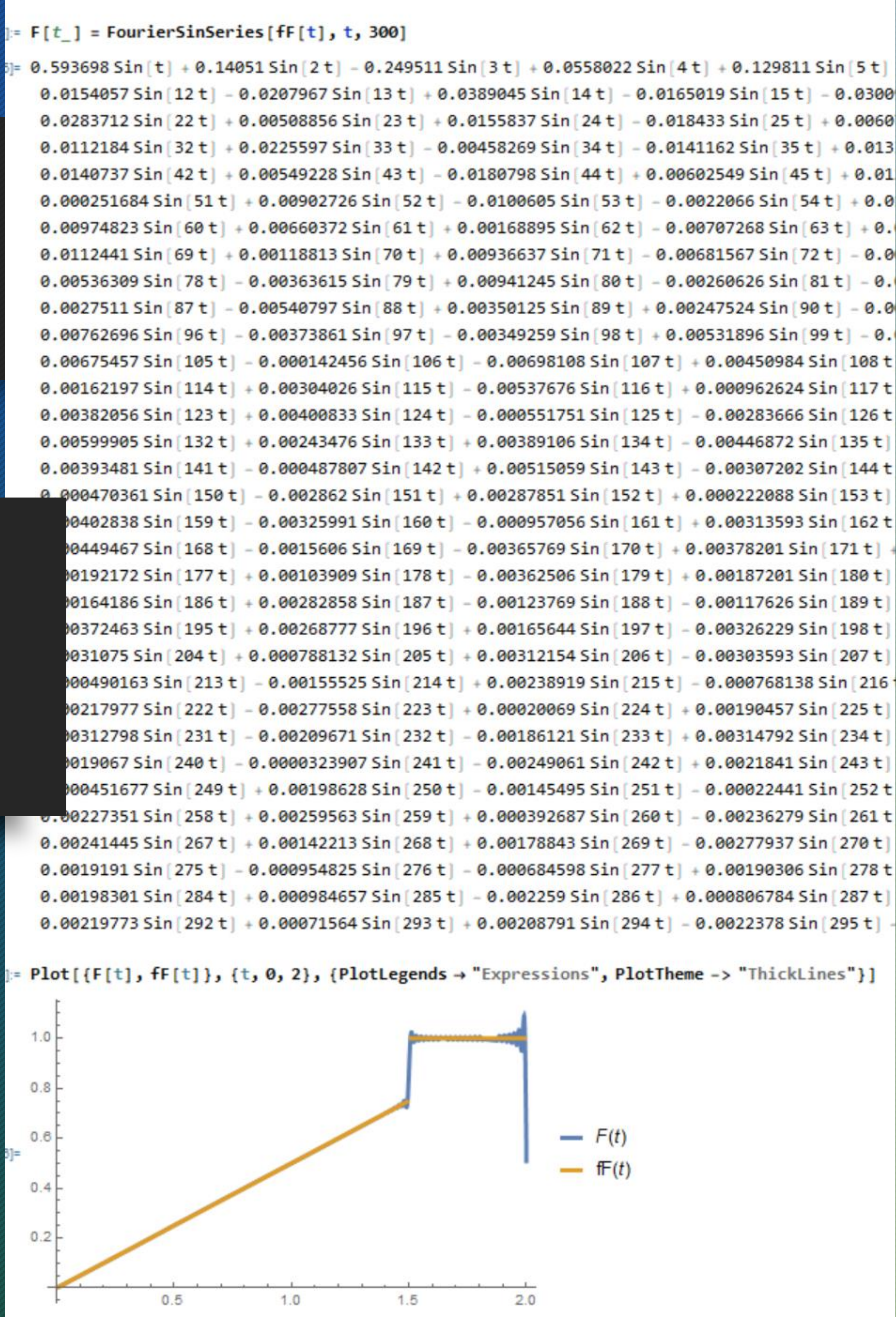
Improved Approximation

- If your computer is sufficiently fast, you may try 50 terms of the Fourier Series:
- You agree with me that we have a superior approximation now. I did not care for the details of the terms. The plot captures the approximations completely.



If you are ready to sleep here!

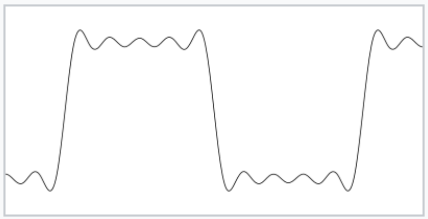
Do this!



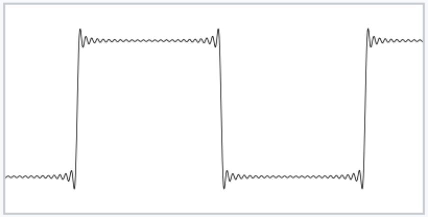
www.oafak.com, www.s2pafrica.org, oafak@unilag.edu.ng

Gibbs, the Phenomenon

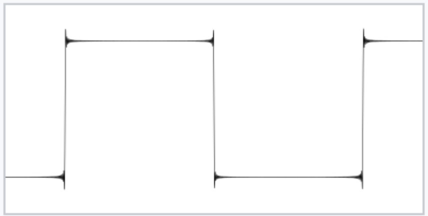
- When you try to compute Fourier approximations to functions there is a well-known error called the Gibbs Phenomenon. It was discovered by Henry Wilbraham (1848) and rediscovered by J. Willard Gibbs (1899)
  - Next Augustine Cauchy, Willard Gibbs is my favorite inspiration and Model.
  - When you realize that the modern digital equipment that we call computers were not invented until nearly 50 years after Gibbs had died, and that you need to compute nearly 20 Harmonics to see this effect, then you know some engineers really knew their Math! Gibbs was the first PhD Engineering awarded in the USA!



Functional approximation of square wave using 5 harmonics



Functional approximation of square wave using 25 harmonics



Functional approximation of square wave using 125 harmonics

Animation of the Square Function

- I want to conclude this Tutorial with an animation of the Square function to demonstrate the Gibb's Phenomenon.
  - The upper plot generates the Fourier approximation of the square function between 0-1.
  - The lower plot calculates the error in the approximation, displaying the Gibbs phenomenon at the edges.
- Some just know enough Math to make them sad! Others actually enjoy it and can make it count in their engineering!