

Phys 304: Assignment 3

Mary Smith*

Haverford College Department of Physics

(Dated: March 1, 2024)

This problem set took me an absurd amount of time because of the approximation method I chose for the first question. I estimate that it took me about 20 hours to do this problem set. I did, however, learn a lot about recursive functions, functions within functions, and when to call those functions. My collaborators for this problem set were Luke Smithberg and Melanie Santiago.

1. PROBLEM 1: THE TRIG LIBRARY

In order to approximate the trigonometric functions, my collaborators and I chose to use Bhaskara's Sine Approximation. Bhaskara was a 7th century Indian mathematician who created an incredibly accurate approximation for the sine function[1]. His derivation of the approximation was not well documented, although the approximation itself has survived. Stroethoff[1] goes through a possible derivation for this. Bhaskara's approximation formula is as follows:

$$\sin \theta^\circ \approx \frac{4\theta(180 - \theta)}{40500 - \theta(180 - \theta)}. \quad (1)$$

Using this, approximations of the sine, cosine, and tangent functions can be graphed with their error.

1.1. The sine function

The pseudocode to create a plot of the sine function is shown in Figure 1. The input angle $x < 180$ is halved until machine precision dictates the Bhaskara function of the next half is equivalent to the Bhaskara function of the previous value. Then using the double angle formula,

$$\sin 2x = 2 \sin x \sqrt{1 - \sin^2 x}, \quad (2)$$

previous angles are recursively added to generate the Bhaskara function of the original angle. This is necessary to obtain the smallest error possible in the calculation of the sine.

Using the method in Figure 1, a graph of the approximation of the sine function can be generated.

1.2. The cosine function

Using the identity

$$\cos x = \sqrt{1 - \sin^2 x}, \quad (3)$$

```
import libraries
define Bhaskara function sine
define double angle function
define marysin(x):
    h=x, hold=2x
    counter:=0
    while |sbbas(h)-sbbas(hold)|>0.001:
        hold=h
        h=h/2
        i+=1
    call h: s=sbbas(h)
    double back to get: for loop i (1,i+1):
        s=doubles
    return s,i
plot the function:
x=linspace for 5 periods
empty list for sine values
for n in x:
    n>360
    subtract down to n≤360
    n>180
    call marysin
    make s negative
    append empty list for sine
plot(x, Bine)
title
show
```

FIG. 1: [The pseudocode for the creation of the sine plot.]

cosine values can be created from the sine values. The pseudocode to create a plot of the cosine function can be found in Figure 3.

From this method in Figure 3, code to graph the approximation of the cosine function can be generated.

1.3. The tangent function

Using the identity

$$\tan x = \frac{\sin x}{\cos x} \quad (4)$$

*Electronic address: masmith@haverford.edu

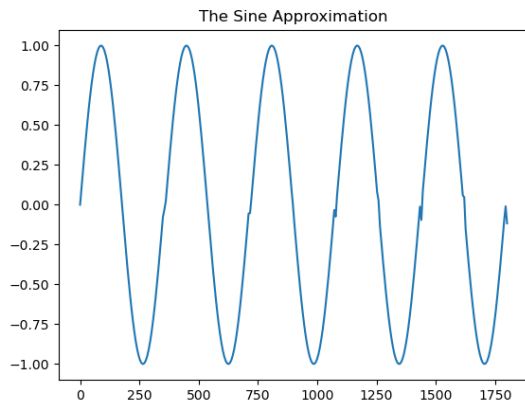


FIG. 2: [Approximation of the sine function using Bhaskara's function.]

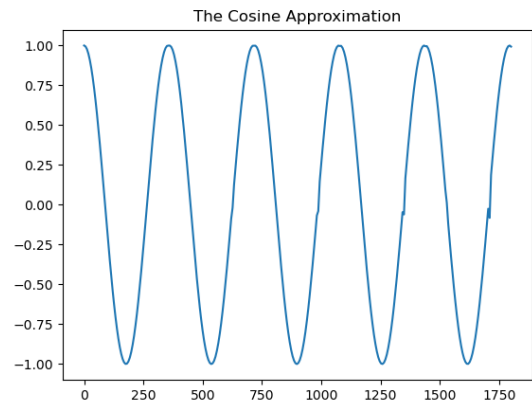


FIG. 4: [Approximation of the cosine function using Bhaskara's function.]

```

1. cosine function:
def mary cos(x):
    call s, i from mary sin
    cosine identity cos(x) = sqrt(1-sin^2 x)
    return cos, i

plot mary cos(x):
x = linspace(0, 1800)
empty list for values of cosine
keep x within domain:
    trig fn periodic every 360 deg, subtract
    fix domains to dec. over right areas
    -> if n > 90 and n < 270
    else return normal
    append cos values for n
plot
title
show

```

FIG. 3: [The pseudocode for the creation of the cosine plot.]

```

tangent function:
def mary tan(x):
    call s, i, c from sine, cosine
    tangent identity: tan(x) = sin(x)/cos(x)
    return t, i

plot mary tan(x):
    same x domain as sin
    empty list for tan vals:
    copy domain structure for sine
    replace + for s
    plot
    title
    show

```

FIG. 5: [The pseudocode for the creation of the tangent plot.]

tangent values can be generated from the original sine values and the cosine values found in Section 1.2. The pseudocode to create a plot of the tangent function can be found in Figure 5.

Using this method in Figure 5, a graph of the approximation of the tangent function can be generated.

What is interesting to note from Figure 8 is how the computer processes the asymptotic behavior of the tangent function. The asymptote at 270° reaches a value of almost -300. This is due to the fact that this function is based off an approximation for sine and cosine. So, the cosine values aren't actually reaching precisely zero

where it should be. This means the tangent function is dividing a number by a very tiny number, which yields values reaching almost -300.

1.4. The error function

The error function

$$\epsilon = \frac{|calc - obs|}{obs} \quad (5)$$

can be used to generate the error between observed and calculated values to find the error in calculated values. This method can be applied to this approximation by taking the angle past the most accurate halved angle of the Bhaskara function as the calculated value, and taking

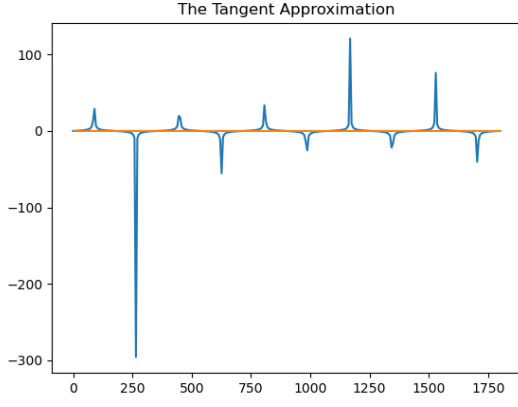


FIG. 6: [Approximation of the tangent function using Bhaskara's function.]

the most accurate halved angle as the observed value. This method is described in Figure 7.

```
error function:
def error(x):
    call s,i from sin
    go one value past precision determined in sine
    in range of i, double values back to original #
    return error:  $\epsilon = \frac{|s_i - s|}{s}$ 
```

FIG. 7: [The pseudocode for the creation of the error plot.]

A plot of the error approximation can then be generated.

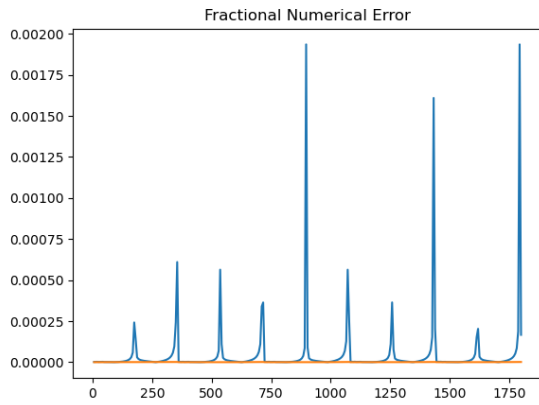


FIG. 8: [The plot of the error function from using Bhaskara's function.]

2. PROBLEM 2: THE MADELUNG CONSTANT

The Madelung constant is a quantity related to the geometric structure of materials[2]. Ionic solids are made up of a lattice structure of charged particles each with their own electrostatic potential. The charged particles, or ions, each have alternating charges in order to maintain their shape. This creates an interesting effect, as oppositely charged ions in the structure will attract and the same charges will repel. So, it is useful to know information about the Madelung constant for a solid. The constant is then used to derive the Madelung energy, which yields the lattice energy of the structure. The lattice energy determines many physical properties of the material.

2.1. Plotting the constant

The Madelung constant is given by:

$$M = \sum_{j,k,l=-\infty}^{\infty} \frac{(-1)^{j+k+l}}{\sqrt{j^2 + k^2 + l^2}} \quad (6)$$

Using the process described in Figure 9, code to generate a plot of the Madelung constant can be created.

```
2. Plotting Madelung constant:
import libraries
define Madelung function
set M as float 0
create nested loops in summation range for different variables
for l in range
    for k in range
        for j in range
            exclude cases where denom. = 0
empty list for values of Madelung
set x range
append values list for each # in range
set fonts
plot, title, show
finding the expression:
import libraries
set M, n
while loop for summation
for when num in summation index
expression we are summing over
break at machine precision
keep sum over odd values
calc. full expression and print
```

FIG. 9: [The pseudocode for the creation of the Madelung plot and expression comparison.]

The plot of the Madelung constant is given in Figure 10.

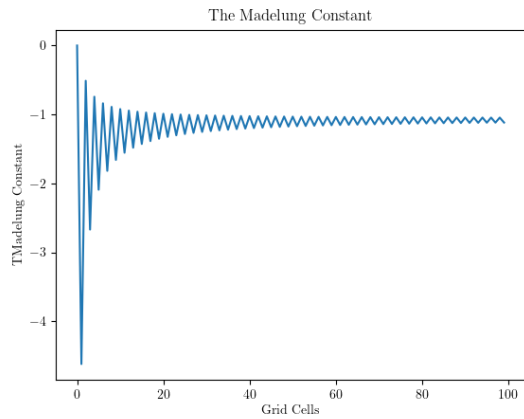


FIG. 10: [The plot of the Madelung function, described in Equation 6.]

The expression

$$M = 12\pi \sum_{m,n \geq 1, \text{odd}} \text{sech}^2\left(\frac{\pi}{2}\sqrt{m^2 + n^2}\right) \quad (7)$$

gives a possible solution for the Madelung constant. Using the process described in Figure 9, this yields a result $M \approx 1.7475$, which lies on the curve in Figure 10.

-
- [1] K. Stroethoff, The Mathematics Enthusiast (2014).
 [2] L. Glasser, Inorganic Chemistry (2012).