Proving Archimedes' Method for Calculating Pi (π) using GCSE Level Mathematics

by;

Mr V. Kangellaris

I have derived the following iteration formulae by adapting Archimedes' method for calculating Pi (π). I have begun with a square inside a circle for the lower bound and a square outside the circle for the upper bound, then continued doubling the number of sides. By the end of my work I was calculating pi to a specified accuracy by using the difference between my lower bound value and upper bound value. My method assumes no prior knowledge of pi. I believe this work follows on from the author of; https://www.craig-wood.com/nick/articles/pi-archimedes/, who has only derived the lower bound.

As the author of https://www.craig-wood.com/nick/articles/pi-archimedes/ has only derived the lower bound part of the formula they are unable to get their own error on their calculating and have had to rely on comparing their initial results to the accepted value of pi.

I believe my derivations test Archimedes method for calculating pi and give the learners a chance to think about rounding to an appropriate accuracy, which was how Archimedes worked. It is also interesting to compare the final codes that myself and the author of https://www.craig-wood.com/nick/articles/pi-archimedes/ used. Who is the better mathematician? I will let you decide.

Here are the formulae that I will be deriving;

$$\lim_{n\to\infty} 2^n x_n \to \pi$$

and

$$2^n y_n \to \pi$$

Where;

$$2^n x_n < \pi < 2^n y_n$$

$$x_1 = \sqrt{2}$$

$$x_{n+1} = \sqrt{2 - 2\sqrt{1 - \frac{{x_n}^2}{4}}}$$

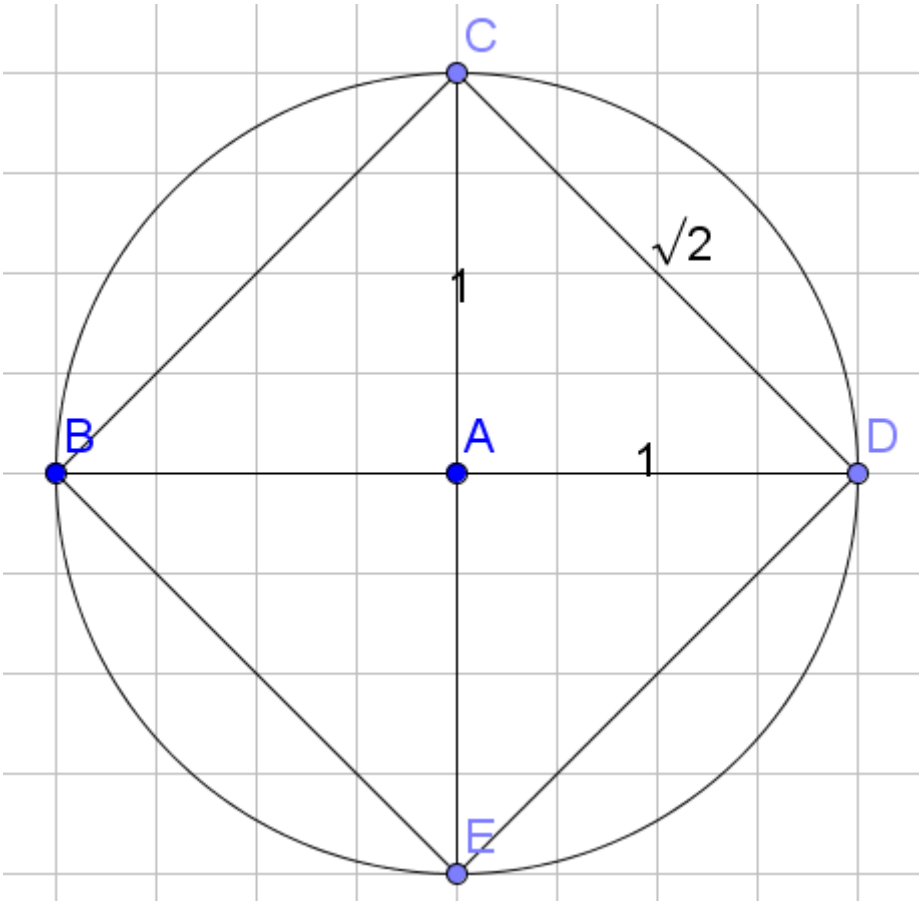
And

$$y_n = \frac{x_n}{\sqrt{1 - \frac{{x_n}^2}{4}}}$$

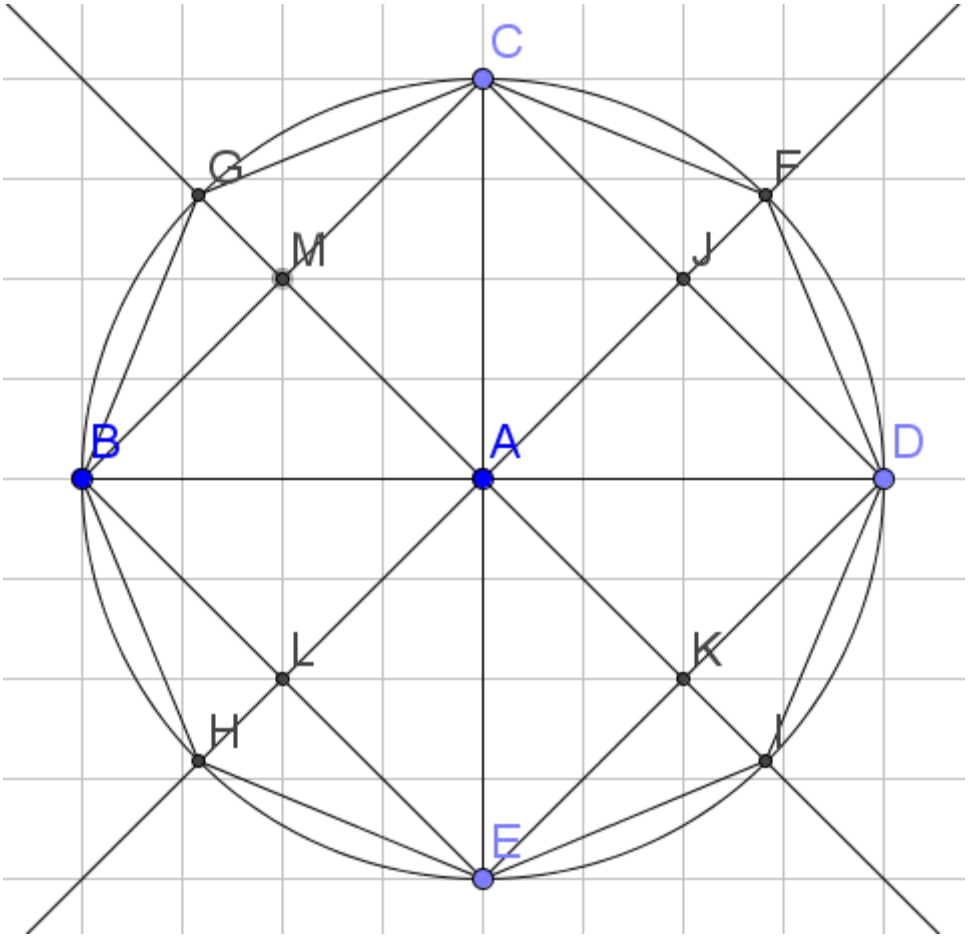
Calculating a Lower Bound Value for Pi (π)

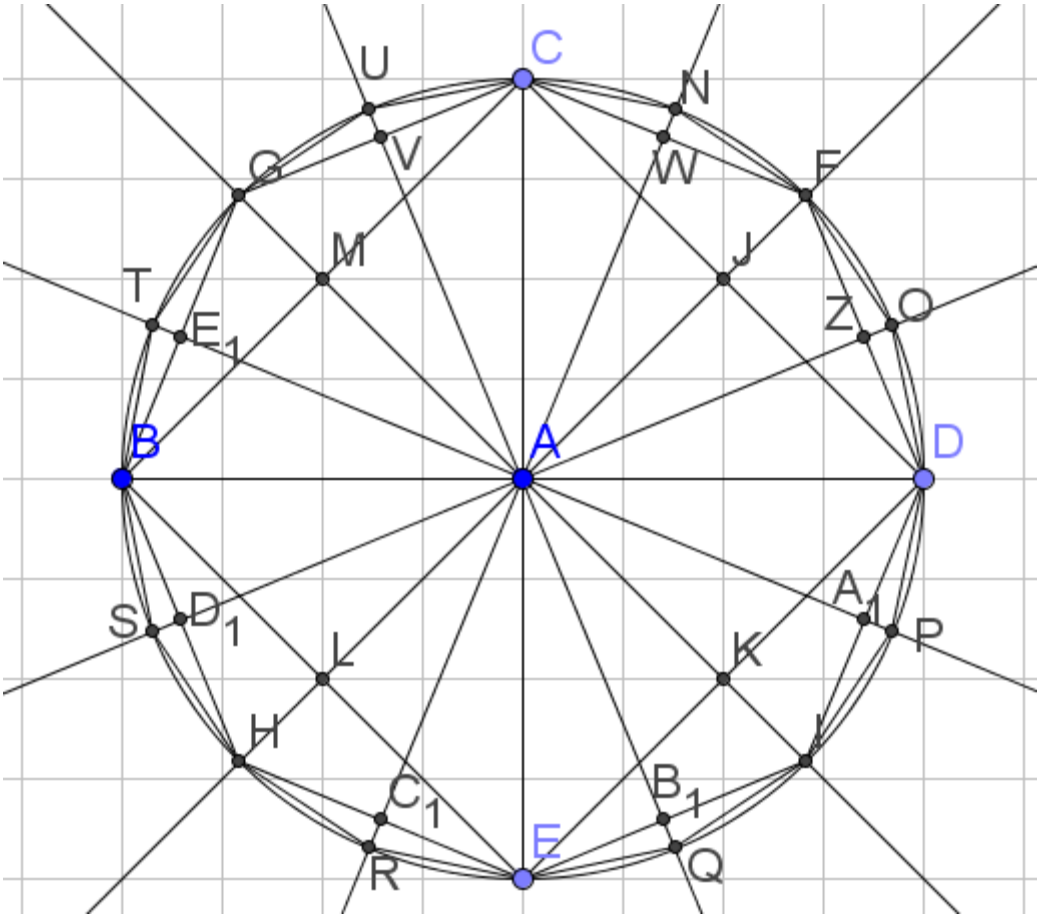
The diameter of our circle is fixed at 2. All the polygons used when calculating a lower bound value for pi are regular, with all corners found on the circle's circumference.

The first regular polygon considered is a square where x_1 is the length of one side. If the above is true then $x_1 = \sqrt{2}$. See below



The polygon's sides are doubled by putting new corners equidistant between adjacent corners on the circumference. This means any line segment formed using a new corner and the centre of the circle will bisect the side of the previous polygon.


C
G
F
M
J
B
A
D
L
K
H
I
E



For an Octogon the length of a side is x_2:

$$x_1 = \sqrt{2}$$

Line segment CJ is equal to $\frac{x_1}{2}$ as point J is the midpoint of the line segment CD.

As line segment AF is the perpendicular bisector of line segment CD and of length 1.

Angles ∠CJF and ∠CJA are 90 degrees. Using Pythagoras' theorem;

$$\text{AJ} = \sqrt{1 - \frac{{x_1}^2}{4}}$$

$$\text{FJ} = 1 - \left(\sqrt{1 - \frac{{x_1}^2}{4}}\right)$$

$$\text{CF} = \sqrt{\frac{{x_1}^2}{4} + \left[1 - \left(\sqrt{1 - \frac{{x_1}^2}{4}}\right)\right]^2}$$

Since our octagon is regular than:

$$x_2 = \sqrt{\frac{{x_1}^2}{4} + \left[1 - \left(\sqrt{1 - \frac{{x_1}^2}{4}}\right)\right]^2}$$

For the Hexadecagon the length of a side is x_3:

Line segment CW is equal to $\frac{x_2}{2}$ as point W is the midpoint of the line segment CF.

As line segment AN is the perpendicular bisector of line segment CF and of length 1.

Angles ∠CWA and ∠CWN are 90 degrees. Using Pythagoras

$$\text{AW} = \sqrt{1 - \frac{{x_2}^2}{4}}$$

$$\text{WN} = 1 - \left(\sqrt{1 - \frac{{x_2}^2}{4}}\right)$$

$$\text{CN} = \sqrt{\frac{{x_2}^2}{4} + \left[1 - \left(\sqrt{1 - \frac{{x_2}^2}{4}}\right)\right]^2}$$

Since our Hexadecagon is regular;

$$x_3 = \sqrt{\frac{{x_2}^2}{4} + \left[1 - \left(\sqrt{1 - \frac{{x_2}^2}{4}}\right)\right]^2}$$

Since all new regular polygons are going to be formed in the same way, then the length of the new sides can be found using Pythagoras' Theorem.

Therefore;

$$\boldsymbol{x_{n+1} = \sqrt{\frac{{x_n}^2}{4} + \left[1 - \left(\sqrt{1 - \frac{{x_n}^2}{4}}\right)\right]^2}}$$

$$\boldsymbol{x_{n+1} = \sqrt{\frac{{x_n}^2}{4} + 1 - 2\sqrt{1 - \frac{{x_n}^2}{4}} + 1 - \frac{{x_n}^2}{4}}}$$

$$x_{n+1} = \sqrt{2 - 2\sqrt{1 - \frac{{x_n}^2}{4}}}$$

Since the perimeter of the

Square is $2^2\sqrt{2}$

or $2^2\, x_1$,

then the perimeter of the

Octagon is $2^3\sqrt{2 - 2\sqrt{1 - \frac{(\sqrt{2})^2}{4}}}$

or $2^3\, x_2$

Which can be generalised to;

$$2^{n+1}\, x_n$$

As we increase the number of sides,

$$\lim_{n\to\infty} \; 2^{n+1}\, x_n \to C$$

where C is the circumference of a circle with a diameter of 2.

Since π is defined in the following formula;

C = $\pi\, D$,

then

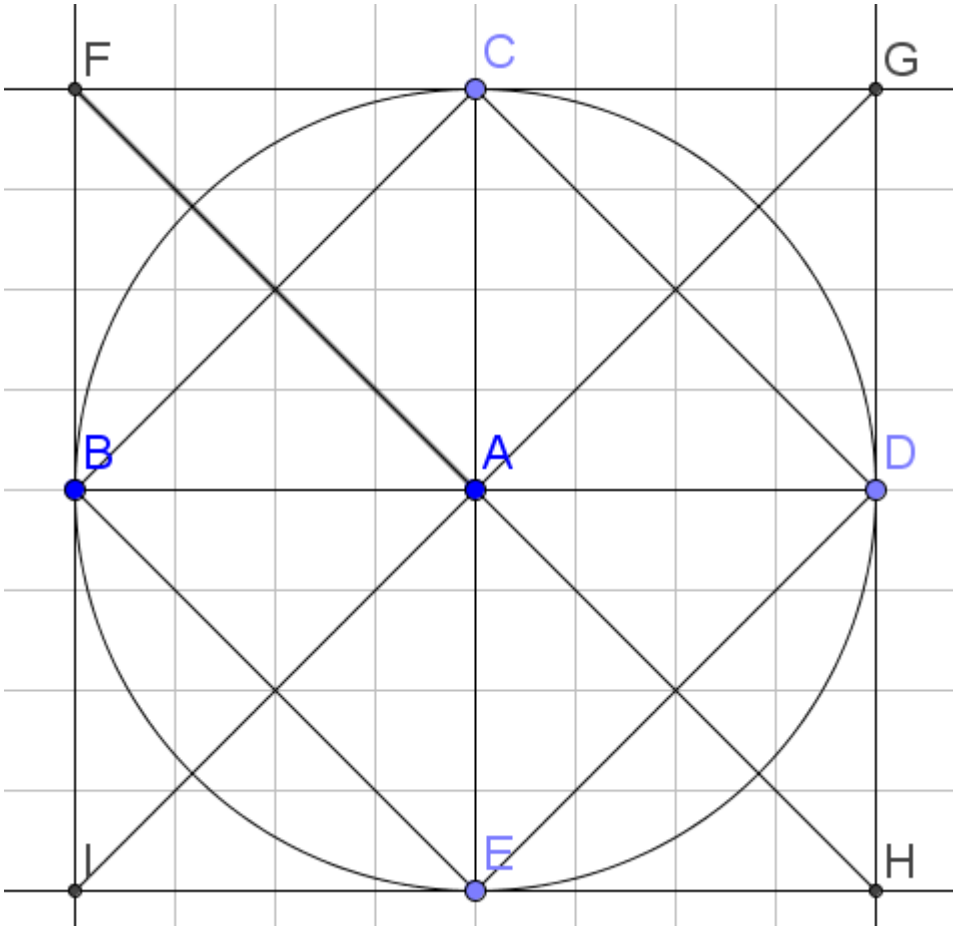
$$\lim_{n\to\infty} \; 2^n\, x_n \to \pi$$

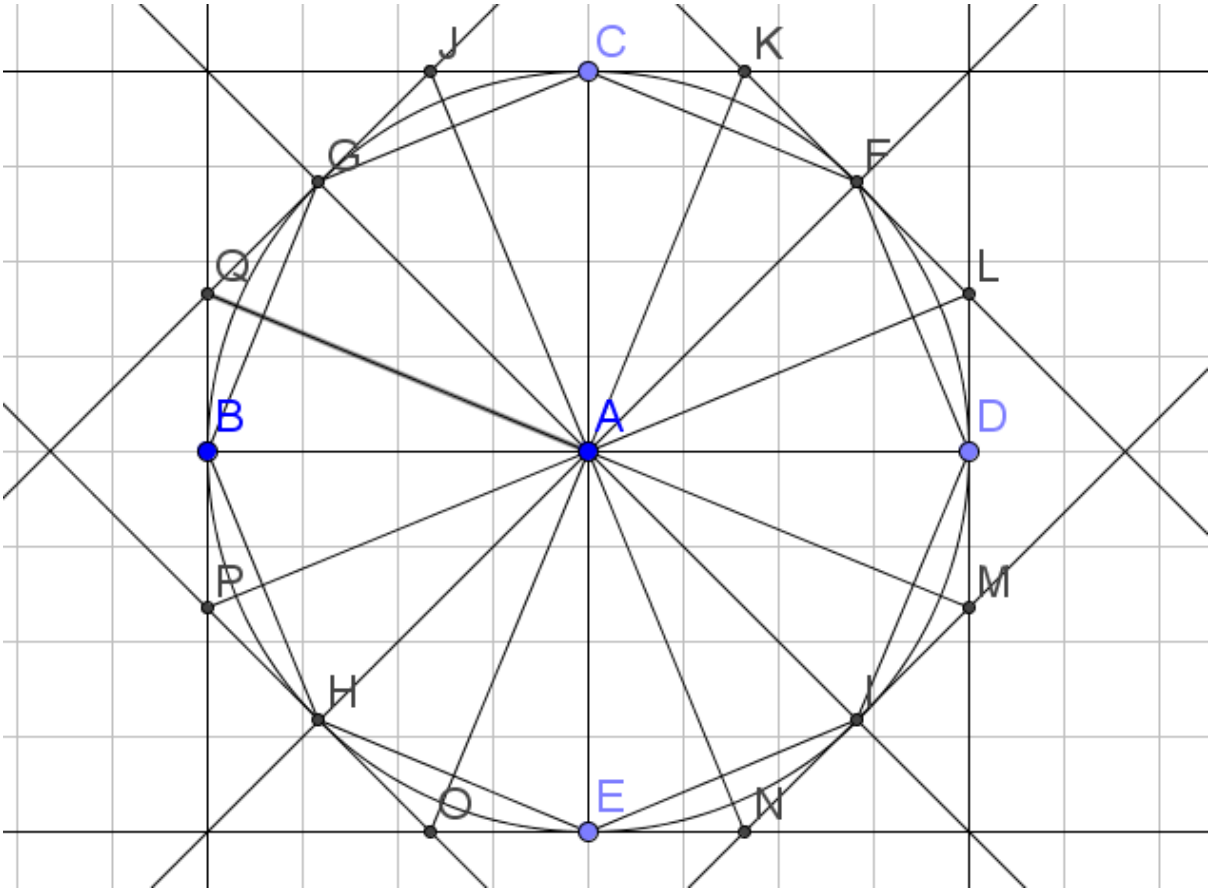
Where;

$$x_1 = \sqrt{2}$$

$$x_{n+1} = \sqrt{2 - 2\sqrt{1 - \frac{{x_n}^2}{4}}}$$

Calculating an Upper Bound Value for Pi (π):

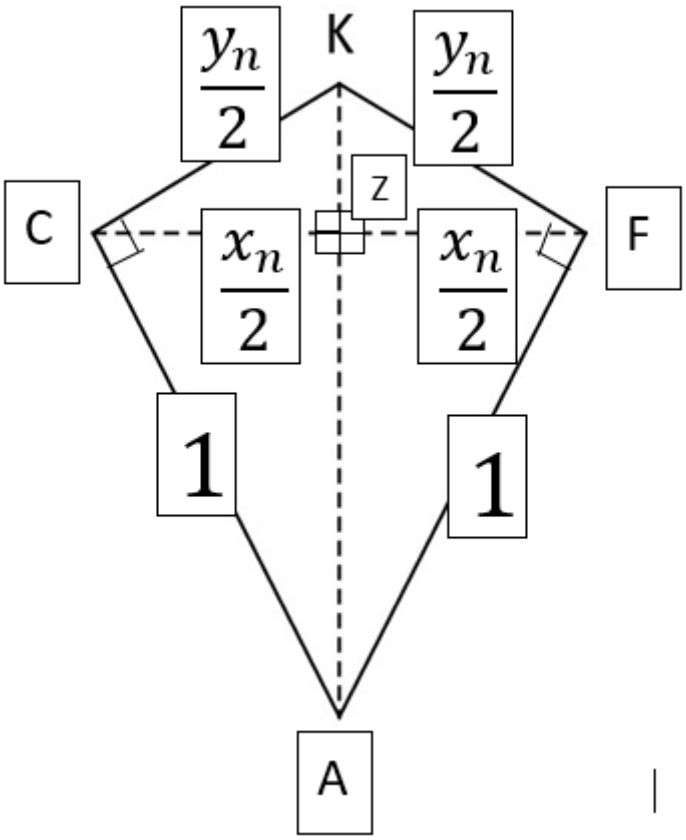




The upper bound polygons are formed by the tangents at the corners of the lower bound polygons, as the lower bound polygons are regular and their corners are on the circumference of the circle, the tangents intersect to form upper polygon sides of equal lengths where the corners of the upper bound polygon are equidistant between the adjacent corners of the lower bound polygon. Hence, the line segments from the corners of the upper bound polygon to the centre of the circle bisect the sides of the lower bound polygons at right angles.

Take $\boldsymbol{y_n}$ to be the length of a side of our regular upper bound polygon with $\boldsymbol{2^{n+1}}$ sides.

Then following is true when considering similar triangles



Triangles KFZ, KCZ, FZA, ACZ, KCA and KFA are all similar as they all have the same angles. This has come about because they are triangles which have been formed by joining the opposite corners of either a square or a kite. Also, as the angles in a triangle total 180 degrees and all the angles at the point of intersection Z are 90 degrees the following must be true;

angle ∠ZAC = a°, ∠FAZ = a° (∠FAC is a bisected angle), ∠ZCA = (90 –a)°, ∠ZFA = (90 –a)° (as line segment KA is a line of symmetry), ∠KCZ = a°(because ∠KCA =90° and ∠KCZ + ∠ZCA= ∠KCA), ∠KFZ = a° (as line segment KA is being a line of symmetry). ∠CKZ = (90 –a)° (because ∠KZC=90°, ∠CKZ + ∠KCZ= 90°). ∠FKZ = (90 –a)° (as line segment KA is being a line of symmetry)

Considering these similar triangles and taking line segment KZ to be of an arbitrary length u;

$$\frac{u}{\frac{y_n}{2}} = \frac{\frac{x_n}{2}}{1}$$

$$u = \frac{x_n\, y_n}{4}$$

So using Pythagoras;

$$\left(\frac{x_n}{2}\right)^2 + u^2 = \left(\frac{y_n}{2}\right)^2$$

$$\left(\frac{x_n}{2}\right)^2 + \left(\frac{x_n\, y_n}{4}\right)^2 = \left(\frac{y_n}{2}\right)^2$$

$$\left(\frac{y_n}{2}\right)^2 - \left(\frac{x_n\, y_n}{4}\right)^2 = \left(\frac{x_n}{2}\right)^2$$

$$\left(\frac{y_n}{2}\right)^2 \left(1 - \left(\frac{x_n}{2}\right)^2\right) = \left(\frac{x_n}{2}\right)^2$$

$$\left(\frac{y_n}{2}\right)^2 = \frac{\left(\frac{x_n}{2}\right)^2}{\left(1 - \left(\frac{x_n}{2}\right)^2\right)}$$

Simplifying to;

$$y_n = \frac{x_n}{\sqrt{1-\frac{x_n{}^2}{4}}}$$

Since the upper bound polygon is always made in the same way and y_n has been derived by considering the line segment KZ to be of an arbitrary length u, the formula for y_n is true for all upper bound polygons.

As the upper bound polygons we are considering have the same number of sides as the lower bound polygons. The perimeter of any upper bound polygon can be found by;

$$2^{n+1}\, y_n$$

As we increase the number of sides,

$$\lim_{n\to\infty} \; 2^{n+1}\, y_n \to \mathbf{C}$$

where C is the circumference of a circle with a diameter of 2.

And as π is defined by the formula

$$C = \pi\, D$$

then

$$\lim_{n\to\infty} \; 2^n \, y_n \to \pi$$

My formulas were checked using Python Programming language, here was my first attempt;

```
import math
import time
print("Caculating a value for pi by considering polygons with increasing numbers of sides")
print("Polygons have been given a fixed diagonal of 2" )
print("___")
n=1
siDe_sQuared=float(2)
for counter in range(21):
    print("The number of sides of this polygon is", 2**(n+1))
    pil=siDe_sQuared**0.5*(2**(n))
    piu=(siDe_sQuared/(1-(siDe_sQuared/4)))**0.5*(2**n)
    error=piu-pil
    print("We have a lower bound value of pi equal  to", pil)
    print("We have an upper bound value of pi equal to", piu)
    print("The error in either of our pi values is", error)
    print("___")
    siDe_sQuared=float(2-2*((1-((siDe_sQuared)/4))**0.5))
    n=n+1
time.sleep(2000)
```

Here are my results:

Python 3.3.2 Shell

File Edit Shell Debug Options Windows Help

```
We have a lower bound value of pi equal  to 3.1415926334632482
We have an upper bound value of pi equal to 3.14159269121694
The error in either of our pi values is 5.77536916068766e-08

___
The number of sides of this polygon is 32768
We have a lower bound value of pi equal  to 3.1415926548075 89
We have an upper bound value of pi equal to 3.1415926692460 12
The error in either of our pi values is 1.443842290171915e-08

___
The number of sides of this polygon is 65536
We have a lower bound value of pi equal  to 3.1415926453212153
We have an upper bound value of pi equal to 3.1415926489308212
The error in either of our pi values is 3.6096059474743925e-09

___
The number of sides of this polygon is 131072
We have a lower bound value of pi equal  to 3.1415926073757197
We have an upper bound value of pi equal to 3.1415926082781214
The error in either of our pi values is 9.02401708913203e-10

___
The number of sides of this polygon is 262144
We have a lower bound value of pi equal  to 3.1415929109396727
We have an upper bound value of pi equal to 3.14159291116527 3
The error in either of our pi values is 2.2560042722830076e-10

___
The number of sides of this polygon is 524288
We have a lower bound value of pi equal  to 3.1415941251951 91
We have an upper bound value of pi equal to 3.1415941252515913
The error in either of our pi values is 5.640021782937765e-11

___
The number of sides of this polygon is 1048576
We have a lower bound value of pi equal  to 3.1415965537048196
We have an upper bound value of pi equal to 3.14159655371892
The error in either of our pi values is 1.4100276501949338e-11

___
The number of sides of this polygon is 2097152
We have a lower bound value of pi equal  to 3.1415965537048196
We have an upper bound value of pi equal to 3.1415965537083 45
The error in either of our pi values is 3.525180147789797e-12
```

Due to way computers carry out their calculations we were having trouble with getting to a large amount of decimal places. We called on more of our Maths library to ensure we were getting the best results (As did the author of https://www.craig-wood.com/nick/articles/pi-archimedes/). We got our program to use the “decimal module” and “getcontext” to increase the precision of our calculating. The code was run until the difference between the upper bound and lower bound values was less than 10^{-103} to ensure that both the upper bound and lower bound values of pi round correctly to at least 100 decimal places.

Here is my code;

```
import math

import time

import decimal

from decimal import Decimal, getcontext

print("Calculating a value for pi by considering polygons with increasing numbers of sides")

print("Polygons have been given a fixed diagonal of 2" )

print("___")

n=1

siDe_sQuared=decimal.Decimal(2)

decimalPLaces=100

getcontext().prec=(decimalPLaces*4)

print("The number of sides of this polygon is", 2**(n+1))

pil=decimal.Decimal(siDe_sQuared.sqrt()*(2**(n)))

piu=decimal.Decimal((siDe_sQuared/(1-(siDe_sQuared/4))).sqrt()*(2**n))

error=decimal.Decimal(piu-pil)

print("We have a lower bound value of pi equal  to", pil)

print("We have an upper bound value of pi equal to", piu)

print("The error when our upper and lower bounds are subtracted", error)

print("___")

siDe_sQuared=decimal.Decimal(2-2*((1-((siDe_sQuared)/4)).sqrt()))

n=n+1
```

```
while piu - pil >= decimal.Decimal(10**-(decimalPLaces+3)):
    print("The number of sides of this polygon is", 2**(n+1))
    pil=decimal.Decimal(siDe_sQuared.sqrt()*(2**(n)))
    piu=decimal.Decimal((siDe_sQuared/(1-(siDe_sQuared/4))).sqrt()*(2**n))
    error=decimal.Decimal(piu-pil)
    print("We have a lower bound value of pi equal  to", pil)
    print("We have an upper bound value of pi equal to", piu)
    print("The error when our upper and lower bounds are subtracted", error)
    print("___")
    siDe_sQuared=decimal.Decimal(2-2*((1-((siDe_sQuared)/4)).sqrt()))
    n=n+1

else:
    print("We are all done we have both values of pi which round correctly to at least ", decimalPLaces , "decimal places. ")
    print("Our value of pi rounded to ", decimalPLaces , "decimal places is ", round(pil,decimalPLaces))


time.sleep(2000)
```

Here are my results:

```
___
The number of sides of this polygon is 23945242826029513411849172299223580994042798784118784
We have a lower bound value of pi equal  to 3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798213907371032001815202786404633919503728681566454120409449309451610826078741367371145736684557428416106973721942150143491035829395795393718017188464890876401032432207026278049173839196915675270322659353039915429007521088363136477680141435410384882150132027607278097322618313503628069293221600992285 7
We have an upper bound value of pi equal to 3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798216611211920688363722580573597335662060244689530303516336466327588929398848933151095411317364257757114695862777994351630914424975724013645583136572149142605489611176957913731704674238563004045086143653178531575274015897095070807524484598097292668402862700331347381315890122109744975892186137897470 9
The error when our upper and lower bounds are subtracted 2.70384088868654851979416896341615833156312307618310688715687597810332010756577994967463280682934100772214083584420813987859557992861992756594810725826620445717896993163568253083504164732877476348430013861614626649480873193432984434316268690778625273067272406928399327180860611690659896453690518 52E-104
___
We are all done we have both values of pi which round correctly to at least  100 decimal places.
Our value of pi rounded to  100 decimal places is  3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706 80
```

Success!!!

We have both an upper bound and lower bound which round to the same 100 decimal place value. We have calculated a value of pi which has an accuracy determined by the formulae. You will also notice that after the 103rd decimal place our upper bound and lower bound values can no longer be rounded to the same value.

When considering the limits of accuracy, the upper and lower bound will also round to a correct 101 decimal place value. However, this is as accurate as you can claim with an error of 10^{-3}. I decided to be cautious when choosing the wording for the final code.

I am sure you are going to try the code for yourself and even change the precision and decimal places.

“

decimalPLaces=100

getcontext().prec=(decimalPLaces*4)

”

You can find the accepted value of pi here;

http://www.geom.uiuc.edu/~huberty/math5337/groupe/digits.html

Here is some brilliant coding from “ https://www.craig-wood.com/nick/articles/pi-archimedes/” it just uses the lower bound formula;

```
import math
import time
import decimal
from decimal import Decimal, getcontext


def pi_archimedes(n):
    """
    Calculate n iterations of Archimedes PI recurrence relation
    """
    polygon_edge_length_squared = Decimal(2)
```

```
    polygon_sides = 2
    for i in range(n):
        polygon_edge_length_squared = 2 - 2 * (1 -
polygon_edge_length_squared / 4).sqrt()
        polygon_sides *= 2
    return polygon_sides * polygon_edge_length_squared.sqrt()


def main():
    """
    Try the series
    """
    places = 100
    old_result = None
    for n in range(10*places):
        # Do calculations with double precision
        getcontext().prec = 2*places
        result = pi_archimedes(n)
        # Print the result with single precision
        getcontext().prec = places
        result = +result          # do the rounding on result
        print("%3d: %s" % (n, result))
        if result == old_result:
            break
        old_result = result

if __name__ == "__main__":
    main()
```