Insights into the Computation for π

By
Neal C. Gallagher III, Maura F. Gallagher, Joshua C. Fair, Nicolas M. Fair, Jason Cannon-Silber, Brandon W. Mayle, Samuel J. Konkol
Math Club
Socrates Preparatory School
www.socprep.org
josh.fair@socprep.org
Sponsor: Neal C. Gallagher, Ph.D, Socrates Preparatory School, 3955 Red Bug Lake Rd., Casselberry, FL 32707, neal.gallagher@socprep.com

Abstract

Searching for new ways to compute the number π leads us to an analysis of the geometric approach of Archimedes and the recursive Calculus approach of Newton-Raphson.

Archimedes estimates π by inscribing a circle with a regular polygon and calculates the circumference of the circle using the perimeter of the polygon. Unlike Archimedes, we propose a method using irregular polygons. When the computations are performed with an equal number of sides, the irregular polygon method produces a more accurate value for π than does the regular polygon method.

The Newton-Raphson analysis leads to several interesting results. First, we find a rapidly converging recursive formula that computes π to eight decimal places after only three iterations. Second, the analysis leads to a Fourier series expansion for *g(x) = arcsin(sin(x))*. This Fourier series results in several numerical series that can be used to compute π.

1. Introduction

A number of paramount importance in mathematics, science, statistics and engineering, π is commonly defined in plane geometry as the ratio between the circumference and diameter of any circle:

$$\pi = \frac{C}{d}.$$

Often π appears in expressions that have no obvious relation to circles. For example, the most important distribution in statistics, the Normal distribution, contains π:

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}.$$

Being both an irrational and transcendental number (that is, not the root of a polynomial with rational coefficients), π has been the subject of much study over the millenia, and was the first such number to be studied.[1] Some try to find mystical meaning in π [2], while others search for new ways to compute π as a novelty and, in the past, as a practicality.

One of the earliest efforts to compute π was made by Archimedes of Syracuse, who applied known principles of Euclidean geometry and the Pythagorean theorem. In his book *The Measurement of a Circle,* he stated three theorems about the circle, the third of which is as follows [3]:

The ratio of the circumference of any circle to its diameter is less than 22/7, but greater than 223/71.

The history of mathematics is filled with interesting work on the computation of π, much of it by influential mathematicians. For example, Leibniz, Gauss and Euler (who may have been the most prolific) have all worked on the computation of π. This work and that of others is found in the comprehensive book *Pi Unleashed*.[4] Even with this vast history of work, we have been fortunate to develop new and interesting results about the computation of π. In this paper we discuss several original methods for computing π. One is a variation on Archimedes' approach and another is an iteration developed after first investigating the Newton-Raphson method.[5] Section 2 covers the method of Archimedes, while Section 3 is our variation on this method. In Section 4, beginning with Newton's method, we find a new iterative approach and a corresponding proof of convergence. In Section 5 methods using Fourier series are explored that yield both previously known results and new insights.[6]

2. Archimedes' Method

Archimedes used a method of inscribed polygons. As seen in Heath's translation [3], few details remain of this. As such, we cannot be sure of his exact method. However, like Archimedes, we use inscribed polygons, the perimeter of which approximates a circle's circumference. As seen in Figure 1*,* we begin with the simple fact that a regular hexagon of side length one can be inscribed within a circle of radius one. This can be visualized by realizing that this hexagon can be constructed by piecing together six equilateral triangles of side length one that all share a common vertex. The common vertex serves as a circle center in which to inscribe the hexagon. From there, the perimeter of a twelve-sided inscribed polygon (dodecagon) is computed by doubling the number of sides of the hexagon. The length of the dodecagon's side is determined using Figure 1. Right triangle OAB has hypotenuse OB of length 1 and side BA of length ½. By use of the Pythagorean Theorem, the length of side OA, c_1**,** is

$$c_1 = \sqrt{r^2 - (\frac{s_1}{2})^2} = \sqrt{\frac{3}{4}}\ . \qquad (1)$$

This means that $d_1 = 1 - c_1 = 1 - \sqrt{3}/2$. Using the Pythagorean Theorem again for triangle BAC, one finds the side length s_2 for the dodecagon is given as

$$s_2 = \sqrt{(\frac{s_1}{2})^2 + (d_1)^2} = \sqrt{2 - \sqrt{3}}\ . \qquad (2)$$

For an n-sided polygon, the value of π is estimated by *(n/2)*(length of one side)*. For a dodecagon the estimate is

$$\pi \cong 6\sqrt{2-\sqrt{3}} = 3.106.$$

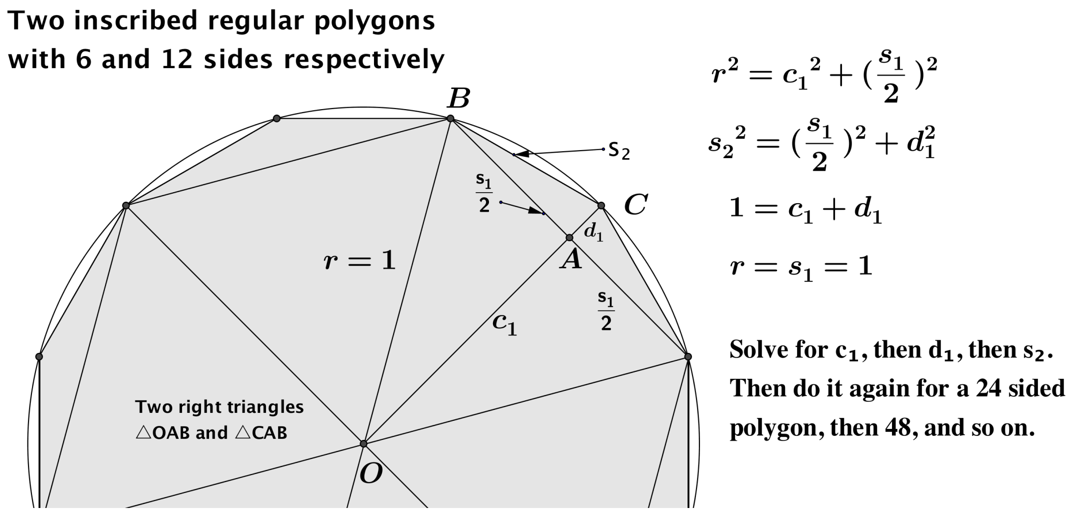
Two inscribed regular polygons
with 6 and 12 sides respectively
B
s_2
$\frac{s_1}{2}$
C
d_1
$r = 1$
A
$\frac{s_1}{2}$
c_1
Two right triangles
$\triangle OAB$ and $\triangle CAB$
O
$r^2 = {c_1}^2 + (\frac{s_1}{2})^2$
${s_2}^2 = (\frac{s_1}{2})^2 + d_1^2$
$1 = c_1 + d_1$
$r = s_1 = 1$
Solve for c_1, then d_1, then s_2. Then do it again for a 24 sided polygon, then 48, and so on.

Figure 1: Archimedes' polygon method. (Only the top portion of circle is shown.)

The method may now be expressed as a recursion over *n*, where the length of the side for a regular polygon is computed by repeatedly doubling the number of sides:

$$c_n = \sqrt{1 - (\frac{s_n}{2})^2},$$

$$d_n = 1 - c_n,$$

$$s_{n+1} = \sqrt{(\frac{s_n}{2})^2 + {d_n}^2}.$$

In Table 1, we present estimates for π computed in a Java program by using regular polygons, doubling the number of sides, beginning with a hexagon. The Java code is contained in Listing 1 found in the Appendix. The source of the error in the computation for π arises primarily from using a chord of a circle to estimate a segment of arc. For example the chord BC in Figure 1 is used as an estimate for the arclength connecting the same two points B and C.

Number of sides	6	12	96	1,572,864
Estimate for π	3.0000	3.1058	3.1410	3.141592653588
Error	0.141592654	0.035764112	5.607026E-4	2.08855155E-12

Table 1: Archimedes' estimate for *π* based on the number of polygon sides. For comparison purposes, an accurate estimate to 17 decimal places for *π* is *3.14159265358979385.*

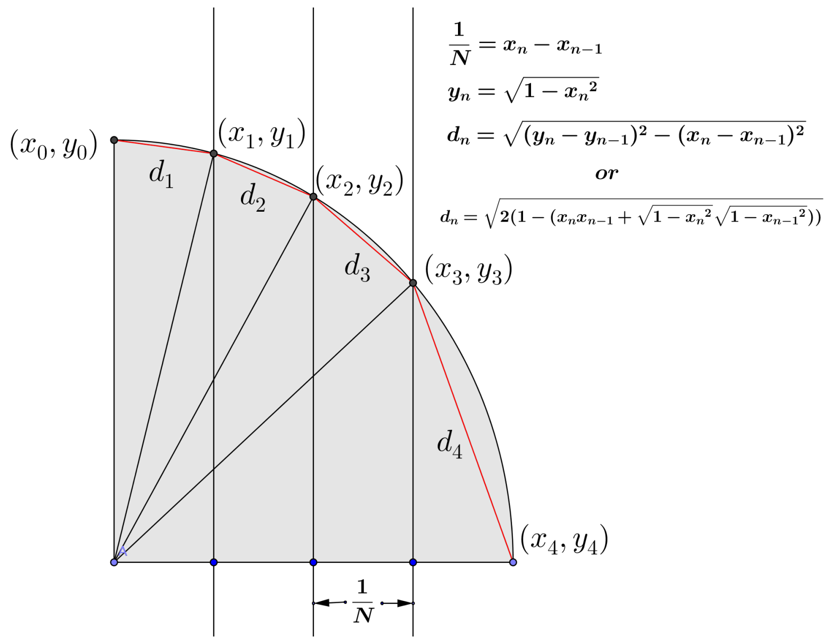
$\frac{1}{N} = x_n - x_{n-1}$
$y_n = \sqrt{1 - x_n{}^2}$
$d_n = \sqrt{(y_n - y_{n-1})^2 - (x_n - x_{n-1})^2}$
or
$d_n = \sqrt{2(1 - (x_n x_{n-1} + \sqrt{1 - x_n{}^2}\sqrt{1 - x_{n-1}{}^2}))}$
(x_0, y_0)
(x_1, y_1)
(x_2, y_2)
(x_3, y_3)
(x_4, y_4)
d_1
d_2
d_3
d_4
$\frac{1}{N}$

Figure 2: Irregular polygon approximation of circumference.

3. Modified Archimedes' Method

Archimedes' approximation uses a regular polygon. A modification of this approach is to inscribe the circle using a polygon where the vertices have equally spaced x-coordinates as illustrated in Figure 2, but the polygon sides form chords of unequal length. The length of the circle's arc over one quadrant is π/2. This arc is approximated by summing the chords connecting the vertices of the irregular polygon. If two adjacent vertices have coordinates (x_{n-1}, y_{n-1}) and (x_n, y_n), then Pythagoras tells us the length of the chord connecting these points is

$$d_n = \sqrt{(y_n - y_{n-1})^2 + (x_n - x_{n-1})^2}\ . \tag{3}$$

The estimate of the quadrant's arc length is found by summing the length of all the chords. Along the x-axis, the values x_n are computed as:

$$x_n = n/N, n = 0, \ldots N,$$

and the values y_n are computed also using the Pythagorean Theorem as:

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

Alternately, using the law of cosines and the following formula:

$$\cos(a-b) = \cos(a)\cos(b) + \sin(a)\sin(b),$$

the value d_n can also be computed as:

$$d_n = \sqrt{2(1-(x_n x_{n-1} + \sqrt{1-x_n^2}\sqrt{1-x_{n-1}^2}))}\,.$$

An examination of Figure 2 reveals the entire circumference is approximated using an irregular polygon with *4N* sides. In order to make a fair comparison between our method and Archimedes', we should compare polygons having the same numbers of sides. For example, using Archimedes' approximation with a regular 96-sided polygon should be compared to our method using a value of N = 24.

Number of Sides	12 (N=3)	24 (N=6)	48 (N=12)	96 (N=24)
Estimate for π	3.0843	3.1215	3.1345	3.1391
Error	0.057340	0.020122	0.007092	0.002504

Table 2***:*** Estimated π values for irregular polygon method. Java code is found in Listing 2 in the Appendix and computed as the variable 'sum'.

Comparing Archimedes' 96 sided regular polygon method in Table 1 with our irregular polygon method in Table 2, Archimedes consistantly computes a slightly more accurate value for π. It might be asked if any advantage can be gained by the use of irregular polygons. Answering this question requires a closer examination of our geometry as seen in Figure 3*,* where the x-axis is divided into two equal segments of length ½. It is further divided into the same number of intervals for each of the two halves. For example, in Figure 2 each half is itself divided into two equal segments. This means that in Figure 3, arc length AB and arc length BC are each spanned by the same number of polygon sides. However, arc BC is twice as long as arc AB as revealed by some simple geometry: Right triangle ODB has a side of length ½ and a hypotenous of 1, making angle BOD equal to π/3 radians. Consequently, angle AOB is π/6 radians, leading to the result that arc BC is twice as long as arc AB. Because each arc is approximated by the same number of chords (see Figure 3), the approximation is more accurate for the shorter chords of arc AB.

This can be illustrated using an example. If the entire irregular polygon has 96 sides, each of arcs AB and BC of Figure 3 has 12 sides. Therefore a 12-chord approximation for arcs BC and AB estimates angles of size π/3 and π/6 respectively. These yield the respective estimates for π to be 3.1380 and 3.1413, as computed using the Java code in Appendix Listing 2. These compare to Archimedes' result of 3.1410, showing that our estimate using the arc for π/6 is better than Archimedes'.

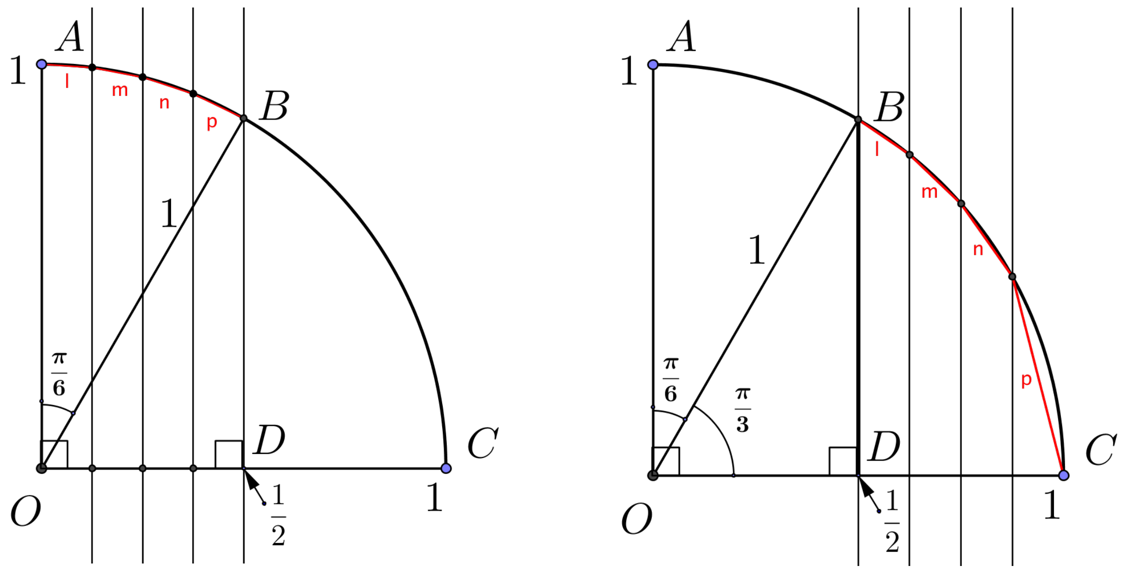

Figure 3: Relationships between *x*-segments *OD* and *DC* and arc lengths *AB* and *BC*, respectively. Both arcs have chords labeled *l, m ,n, p*.

4. Calculus and Newton's method

Now we discuss the problem of numerically finding the zero of a function, satisfying *f(π) = 0*, using Newton's method. Consider the plot of *f(x) = sin(x)* in Figure 4, noting *sin(π) = 0.* Suppose there are two points on the curve, *A* and *B*, the x-coordinates of which are *(π-δ)* and *(π+δ),* respectively, and the respective y-coordinates are *sin(π-δ)* and *sin(π+δ).* Note that, without already knowing the value of π, it is impossible to numerically identify two such points. However, using simple algebra to find the equation of the line connecting two points, and the unknown value δ, the equation for the line connecting points A and B can be determined. Finding the slope and the x-intercept, as shown in Figure 4, the dashed line connecting points *A* and *B* therefore has the equation:

$$y = -\frac{\sin(\delta)}{\delta}(x - \pi) \quad . \tag{4}$$

For this line, the x-intercept is at *x = π,* and the slope is -sin*(δ)/(δ)*. Were the value of *δ* known, the value for *π* would be trivial to compute as being midway between *A* and *B.* Significantly, as *δ* becomes smaller, the slope more and more closely estimates -1. Our purpose is to estimate the value for *π* by finding a zero of *f(x) = sin(x)* using iterations via Newton's method. This proceeds as follows[5]:

1. Select an initial value for *x*, called x_0.
2. Compute successive approximations by using

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \tag{5}$$

where *f'(x)* = the derivative of *f(x)*.

3. Continue the iteration until an acceptably accurate solution is found.

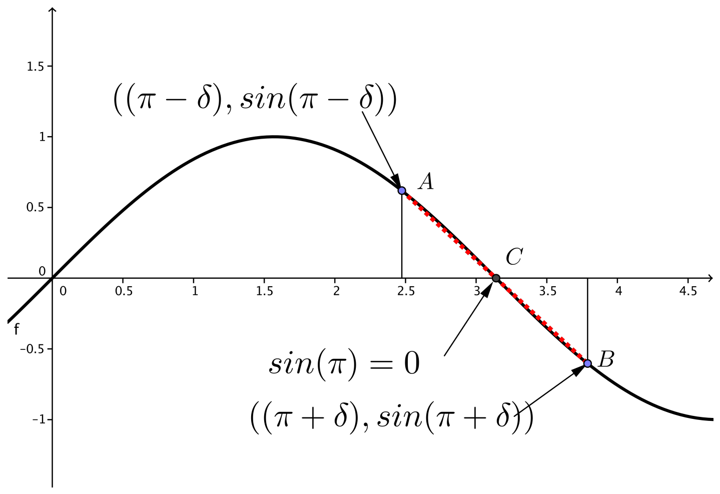

Figure 4: Estimate the value for π by finding a zero of *f(x) = sin(x)*.

Apply Newton's method to *f(x) = sin(x)* as illustrated in Figure 5**.** The initial guess for *π* is $x_0 = \pi - \delta = 2$. The value *δ* is the difference between x_0 and the true value of *π*. Draw the tangent line to *f(x)* at point *A*. Point *A* represents $f(x_0)$. As seen in Figure 5, this tangent line has a slope of $f'(x_0) = cos(x_0) = -cos(\delta)$. It is at the point where this tangent intersects the *x*-axis, that provides Newton's next estimate for *π* at about *4.185*.

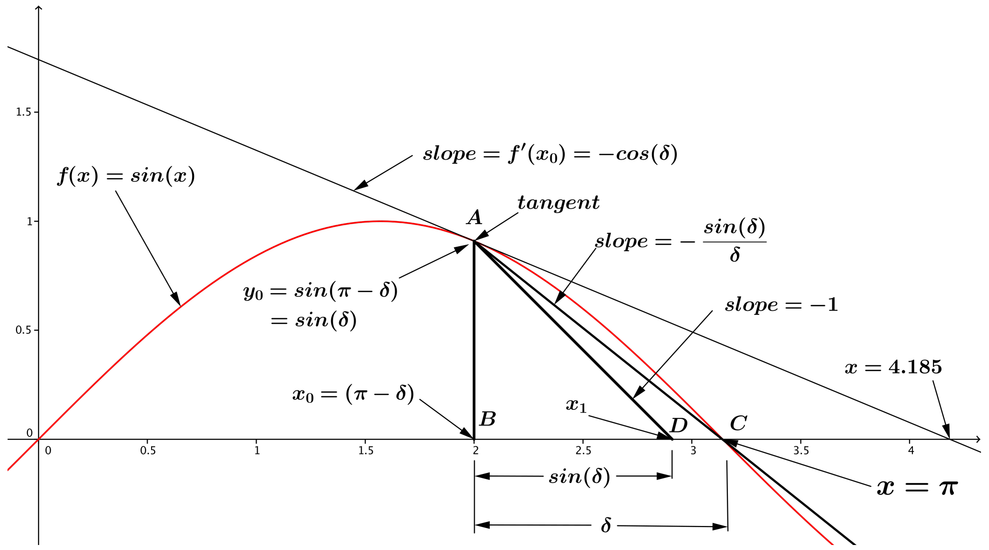

Figure 5**:** Geometry for Newton's method and modified Newton's method. Initial value $x_0 = 2$.

A modification of Newton's method improves the process. First, make an initial guess for the value of *π* such that $x_0 = \pi - \delta$. The *y*-coordinate is $f(x_0) = sin(\pi - \delta) = sin(\delta)$, specifying the point *A* on the sinusoid with coordinates $((\pi - \delta), sin(\delta))$. This point is graphed in both Figure 4 and Figure 5**.** Note the line segment *AC* is also in both figures.

Point *C* has coordinates *(π, 0)* and line segment *AC* has slope *–sin(δ)/δ* as specified in Eq. (4). Notice, with an initial guess of $x_0 = 2$, *δ* becomes *1.1416,* and the slope is *–sin(δ)/δ = -0.8.*

If we could draw line segment *AC* from point *A* to the *x* – axis, it would intersect the axis exactly at *x = π,* thus solving the problem. To do this we would need to know the value *δ*. As the value of *δ* becomes small, the slope for *AC* quickly approaches the value -1, which is the slope of line segment *AD*. This means triangle *ABD* is isosceles. If the length of line segment *AB* is *sin(δ),* then the length of line segment *BD* must also be *sin(δ).* Looking at Figure 5, if the first guess for the value *π* is x_0, the next guess is the value at point D, which is $x_1 = x_0 + \sin(\delta)$. Even though we don't yet know the value for *δ,* we know the value of $\sin(\delta) = \sin(x_0)$, making the modified Newton recursion:

$$x_{n+1} = x_n + \sin(x_n) \quad . \tag{6}$$

$x_0 = 2.0$	$x_1 = 2.90929743$	$x_2 = 3.13950913$	$x_3 = 3.14159265$

Table 3: Rapid convergence for modified Newton's method values. See Java Listing 4 in the Appendix .

Another interesting feature of this recursion is that, no matter the starting value x_0, it almost always converges to a some odd multiple of *π*. This expression is found to be an unusually effective way to compute *π*.

5. Fourier Series

If the value is known for $\sin(\delta) = \sin(x_0)$ by use of Figure 5, it is possible to compute the value for the error *δ* by using $\delta = \arcsin(\sin(x_0))$ giving the actual value of *π* as:

$$\pi = x_0 + \delta = x_0 + \arcsin(\sin(x_0)), \tag{7}$$

provided $\frac{\pi}{2} \le x_0 \le \frac{3\pi}{2}$. Here are a few examples using a calculator:

choose x = 2 sin(x) = 0.909297 arcsin(0.909297) = 1.141593 2 + 1.141593 = *π*	choose x = 3 sin(x) = 0.14112 arcsin(0.14112) = 0.141593 3 + 0.141593 = *π*	choose x = 1.765 sin(x) = 0.981202 arcsin(0.981202) = 1.376593 1.765 + 1.376593 = *π*

Table 4: Example calculations using $\pi = x_0 + \arcsin(\sin(x_0))$.

Figure 6 compares the graphs of *g(x) = arcsin(sin(x))* and *f(x) = sin(x).* A difficulty with using Eq. (7) as a method to compute *π* is that the power series computation for *arcsin(sin(x))* is cumbersome, but using the power series for *arcsin(z)* we can write [7]:

$$\arcsin(z) = z + (\frac{1}{2})\frac{z^3}{3} + (\frac{1\bullet 3}{2\bullet 4})\frac{z^5}{5} + (\frac{1\bullet 3\bullet 5}{2\bullet 4\bullet 6})\frac{z^7}{7}\ldots = \sum_{n=0}^{\infty}\frac{\binom{2n}{n} z^{2n+1}}{4^n(2n+1)}; \quad |z| \le 1$$

where $z = sin(x_0)$. Notice that if *arcsin(z)* is replaced by the first term in its power series above, then term $sin(x_0)$ replaces term $arcsin(sin(x_0))$ in Eq. (7) and it resembles Eq. (6). Also in the power series, by setting $x_0 = \pi/2$, *z* becomes equal to 1, and the series simplifies to a slowly convergent series for $\pi/2$, first computed by Ramanujan.[9]

$$\arcsin(1) = \frac{\pi}{2} = 1 + (\frac{1}{2})\frac{1}{3} + (\frac{1 \bullet 3}{2 \bullet 4})\frac{1}{5} + (\frac{1 \bullet 3 \bullet 5}{2 \bullet 4 \bullet 6})\frac{1}{7} \ldots = \sum_{n=0}^{\infty} \frac{\binom{2n}{n}}{4^n(2n+1)}. \tag{8}$$

As seen in Eq (9) the Fourier series offers a different solution [10].

$$g(x) = \frac{4}{\pi}(\sin(x) - \frac{\sin(3x)}{9} + \frac{\sin(5x)}{25} - \ldots \frac{\sin((2n+1)x)}{(2n+1)^2} + \ldots . \tag{9}$$

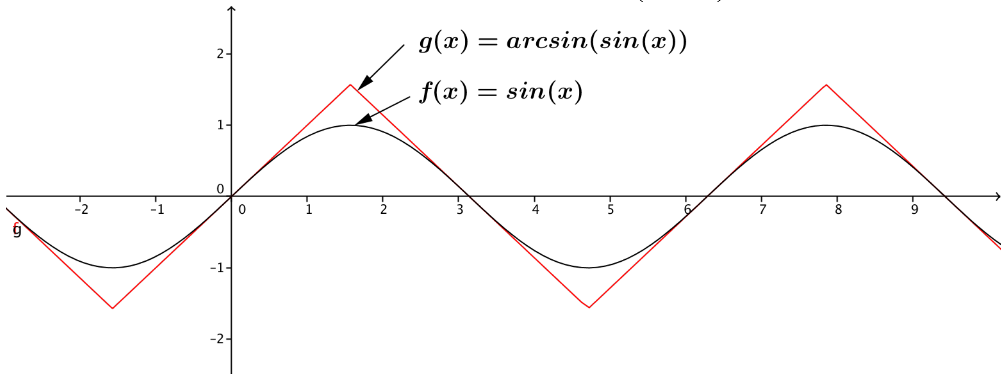

Figure 6: Comparison of functions *f(x) = sin(x*), and *g(x) = arcsin(sin(x)).*

As illustrated in Figure 6, because $g(\pi/2)=\pi/2$, $\sin(\pi/2) = 1$, $\sin(3\pi/2) = -1$, and so on. Equation (9) may be rewritten as:

$$\frac{\pi}{2} = \frac{4}{\pi}(1 + \frac{1}{9} + \frac{1}{25} + \ldots + \frac{1}{(2n+1)^2} + \ldots) , \tag{10}$$

which can be used to compute $\pi^2/8$, and thereby the value for π. This series was first computed by Euler in 1748.[11] Using the series in Eq. (9), yet another series for π can be developed. Because we know the derivative $g'(x) = -1$ when $\pi/2 < x_0 < 3\pi/2$, Eq. (9) can be used to compute the value of π as follows:

$$-\frac{\pi}{4} = \cos(x) - \frac{\cos(3x)}{3} + \frac{\cos(5x)}{5} - \ldots + (-1)^{n+1}\frac{\cos((2n-1)x)}{2n-1} + \ldots$$

Picking x = π, gives the following simple series for π:

$$\pi = 4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots + \frac{(-1)^{n+1}}{2n-1} - \ldots) \tag{11}$$

This series, sometimes called the Leibniz series, converges very slowly. [12]

Suppose we integrate both sides of Eq. (9):

$$\int_0^{\pi/2} x dx = \frac{\pi^2}{8} = \frac{4}{\pi}\int_0^{\pi/2}[\sin(x) - \frac{\sin(3x)}{9} + ... \frac{\sin((2n+1)x)}{(2n+1)^2} + ...]dx.$$

Resulting in a rapidly converging series also computed by Euler: [13]

$$\frac{\pi^3}{32} = 1 - \frac{1}{27} + \frac{1}{125} - ... + \frac{(-1)^n}{(2n+1)^3} + ... \tag{12}$$

Additional convergent series may be revealed by further manipulation of the function *g(x)* in Eq. (9).

Summary

By our observation, methods for computing π fall into three categories: geometric, recursive, and series summation. These categories are not mutually exclusive in that it is possible to have a method be of more than one type. For example, the method proposed by Archimedes is both geometric and recursive.

Our effort has been to identify new variations on methods for computing π in the hope of expanding our insight into both the methods and the number π itself. Archimedes' method approximates a circle using a process that recursively doubles the number of sides for a regular polygon. In contrast, our method uses an irregular polygon the vertices of which have equally spaced x-coordinates. The analysis indicates that under certain conditions, irregular polygons provide a better approximation of the value of π.

Using Calculus directly, Newton-Raphson uses iterations to search for the x-intercept of the function *f(x) = sin(x).* Discovering that a modified version of this method seems to work better than a simple application of Newton's formula, we propose the substitute iteration $x_{n+1} = x_n + \sin(x_n)$. Instead of trying to make a recursive set of polygons converge to a whole circle, the modified Newton's method uses a sequence of numbers that converge to the estimate of the single number π based on the fact that *sin(δ)* converges to the value δ as δ becomes small.

The analysis of Newton-Raphson leads to another geometric approach that provides a Fourier series approximation for the sawtooth function *g(x) = arcsin(sin(x)),* the slope of which is almost always ±1. This special feature, when combined with a Fourier series expansion for the sawtooth, leads to several classical series that when summed produce estimates for π. It may be possible that other similar converging series may be generated using this function *g(x).*

We point to four results that we have not seen mentioned in other literature on π. These are:

1. The irregular polygon method for estimating π,
2. *Explanation* of the convergence of $x_{n+1} = x_n + \sin(x_n)$,
3. Use of the power series for arcsin() to produce a convergent series,

4. The use of the Fourier series for the sawtooth function *g(x) = arcsin(sin(x))*, to produce multiple convergent series for π.

The final conclusion is that the modified Newton's method of Eq. (6) converges with breath-taking speed and is the clear winner in our studies for quickly and easily computing an estimate for π .

Acknowledgement
The reviewers offered suggestions that substantually improved the readability of this work. We are sincerely grateful.

References

[1] D. Wells, Penguin dictionary of curious and interesting numbers, revised edition. Penguin Group, p 31, 1997.

[2] J. Taylor, Great pyramid: why was it built? And who built it, Cambridge Univ. Press 1859, reprinted in 2014.

[3] Heath, T.L., The works of Archimedes (Dover edition), pp 93-98, 1953.

[4] J. Arndt and H. Christoph, Pi Unleashed, Springer-Verlag, Berlin Heidelberg, 2001.

[5] G.B. Thomas, Calculus and Analytic Geometry, Addison-Wesley, p 451, Reading, 1960.

[6] A. Papoulis, The Fourier Integral and It's Applications, McGraw-Hill, New York, 1962.

[7] M. R. Spiegel, Mathematical Handbook of Formulas and Tables, Schaum's Outline Series, Schaum Publishing Co 1st.ed., pp 111, New York, 1968.

[9] J. Arndt and H. Christoph, Pi Unleashed, p 226 eq.16.47, Springer-Verlag, Berlin Heidelberg, 2001.

[10] M. R. Spiegel, Mathematical Handbook of Formulas and Tables, Schaum's Outline Series, Schaum Publishing Co 1st.ed., pp 132, New York, 1968.

[11] J. Arndt and H. Christoph, Pi Unleashed, p 225 eq.16.29, Springer-Verlag, Berlin Heidelberg, 2001.

[12] Ibid, p 70.

[13] Ibid, p 225 eq 16.31.

Appendix
Java code used in computations

Listing 1:
This program listing computes values for π using the Archimedes polygon approximation approach.

```
class PiArchimedes{
  public static void main(String args[]){
  // computes the value for pi using inscribed regular
  // polygons like Archimedes
    double s = 1;// hexagon of side lenght 1 inscribed in a circle of
               // radius 1
    int N = 96;// N limits the size of the largest polygon to 2*N
    int n = 6; // n = 6 begins with a hexagon
    double E;
      while(n <= N){
      double c = c(s);
      double d = d(c);
      s = new_s(s,d);// computes s for a 2n sided polygon
      // pi for dodecagon is 6*s - in general pi = n*s
      double pi = n*s;// pi estimate is computed as
                //(number of sides)/2 * (length of each side)
      E = Math.PI - pi;//E is the computation error

      System.out.println("number of sides is "+2*n);
      System.out.println("c = "+c+" d = "+d+" s2 = "+s+" Pi = "+pi);
      System.out.println("Error in calculation = "+E);
      n *= 2;
      }
  }
  // the parameter s2 is for the next polygon with 2*n sides
  static double new_s(double s1,double d){
    double s2=Math.sqrt((s1/2)*(s1/2)+(d*d));
    return s2;
  }
  //Compute the parameter c1 from s1
  static double c(double s1){
   double c1 = Math.sqrt(1-(s1/2)*(s1/2));
   return c1;
  }
  //Compute the parameter d using c.
  static double d(double c){
   double d = 1.0 - c;
   return d;
  }
}
```

Listing 2:
This contains code for our alternate method for the computation of π by use of irregular polygons. The variables sum and sum1 compute our full quadrant irregular polygon estimate. sum and sum1 are the same estimate computed using slightly different methods as a check on one another. sum3 is the estimate computed using the π/3 included angle, while sum4 is the estimate computed using the π/6 included angle.

```
// three sets of computations for irregular polygons - sum and sum1
// are computations for the entire quadrant - sum2 is for the included
// angle of pi/3 - sum3 is for the included angle pi/6
  class Pi2_4{
  public static void main(String args[]){
  int N = 24;// equivalent to a 4N sided Archimedes polygon
          // estimate depending on method
    double x, y, sum = 0.0, sum1 = 0.0;
// estimates the sum for a irregular polygon about the first quandrant
// pi = 2*sum = 2*sum1
    double x0 = 0.0,y0 = 1.0;
    for(int i = 1; i < N+1; i++){
      x = i*1.0/N;
      y = Math.sqrt(1 - (x*x));
// estimates pi using irregular polygon about first quadrant
// total number of polygon sides circumscribing the circle is 4N
// sum computes the length of the polygon sides using the
// formula that computes the distance between points (x,y) and (x0,y0)
      sum += Math.sqrt((x-x0)*(x-x0)+(y-y0)*(y-y0));
// sum1 computes the same distance using the law of cosines formula
// both sum and sum1 estimate the value for pi/2
      sum1 += Math.sqrt(2*(1-(x*x0 + Math.sqrt(1-x*x)*Math.sqrt(1-x0*x0))));
      x0 = x;
      y0 = y;
  }
// estimates pi over the angle of size pi/3 over the x-range
// circumscribed polygon has 8N sides
// 0.5 <= x <= 1.0
       N = N/2;// N is divided by because we are only using
       x0 = 0.5;// half of the region 0<x<1. Equivalent now to a
       y0 = Math.sqrt(1.0 - x0*x0);// 8*N sided Archimedes polygon
       double sum2 = 0.0;
       for(int i = 1; i < N+1; i++){
         x = x0 + 0.5/N;
         if(x>1.0) x = 1.0;
         y = Math.sqrt(1 - (x*x));
 // sum2 estimates pi/3
         sum2 += Math.sqrt((x-x0)*(x-x0)+(y-y0)*(y-y0));
```

```
          x0 = x;
          y0 = y;
        }
// estimates pi over the angle of size pi/6 over the x-range
// 0 <= x <= 0.5
// circumscribed polygon has 8N sides
        x0 = 0.0;
        y0 = Math.sqrt(1.0 - x0*x0);
        double sum3 = 0.0;
        for(int i = 1; i < N+1; i++){
          x = x0 + 0.5/N;
          if(x>1.0) x = 1.0;
          y = Math.sqrt(1 - (x*x));
// sum3 estimates pi/6
          sum3 += Math.sqrt((x-x0)*(x-x0)+(y-y0)*(y-y0));
          x0 = x;
          y0 = y;
        }
      System.out.println("Irregular "+(4*N)+" sided polygon about first quandrant
estimates Pi as       " + 2*sum+"\n error is "+ (Math.PI-2*sum));
      System.out.println("Same irregular polygon using law of cosines distance
computation Pi as "+2*sum1+"\n error is "+ (Math.PI-2*sum1));
      System.out.println("Pi/3 estimate for Pi using "+(8*N)+" sides is
"+ 3*sum2+"\n error is "+ (Math.PI-3*sum2));
      System.out.println("Pi/6 estimate for Pi using "+(8*N)+" sides is
" + 6*sum3+"\n error is "+ (Math.PI-6*sum3));
      System.out.println("True Pi =                                        "+ Math.PI);

  }
}
```

```
        System.out.println("Pi/3 estimate for Pi using "+(8*N)+" sides is
"+ 3*sum2+"\n error is "+ (Math.PI-3*sum2));
      System.out.println("Pi/6 estimate for Pi using "+(8*N)+" sides is
" + 6*sum3+"\n error is "+ (Math.PI-6*sum3));
      System.out.println("True Pi =                                        "+ Math.PI);

  }
}
```

Listing 3:
Our article contains several series that sum to π. This listing contains the code that sums several of these series as noted in the comments. We mention that sum2 is a very slowly converging series and the program can take quite a while to perform this calculation.

```
 class PIsimplesum{
 public static void main(String Args[]){
    // outputs the number of terms it takes for each series
```

```
  // to compute pi to within the value of error.
  //
  double error = .00003;
 simpleSum(error);
 simpleSum2(error);
 simpleSum3(error);
 simpleSum4(error);
}
static void simpleSum(double error){
  // This computes the series for equation(11).
  double term = 1;
  double sum = 0.0;
  double pi = 0.0;
  long n = 1;
  int sign = -1;
     while (Math.abs((Math.PI-pi)) > error){
     sign = - sign;
     term = sign*1.0/(2*n-1);
     sum = sum + term;
     pi = 4.0*sum;
     n++;
}
   System.out.println("pi1 is  "+pi+" n is "+n);
}
static void simpleSum2(double error){
  // This computes the series for equation(8).
  double a = 1.0;
  double b = 1.0;
  double sum = 1.0;
  double pi = 0.0;
  long n = 1;
     while (Math.abs(Math.PI-pi) > error){
     a = a*(2*n-1.0)/(2*n);
     b = a/(2*n+1);
     sum = sum + b;
     pi = 2.0*sum;
     n++;
}
     System.out.println(" pi2 is "+pi+" n is "+n);
}
static void simpleSum3(double error){
//This computes the series in equation(10).
double sum = 1.0;
double term = 1.0;
int n = 1;
double pi = 0.0;
```

```
  while (Math.abs(Math.PI-pi) > error){
    term = 1.0/((2*n+1.0)*(2*n+1.0));
    sum = sum + term;
    pi = Math.sqrt(8.0*sum);
    n++;
  }
    System.out.println("  pi3 is "+pi+" n is "+n);
  }
  static void simpleSum4(double error){
    //This computes the series in equation(12).
  double sum = 1.0;
  double term = 1.0;
  int n = 1;
  int sign = -1;
  double pi = 0.0;
  while (Math.abs(Math.PI-pi) > error){
    term = sign/((2.*n+1)*(2.*n+1)*(2.*n+1));
    sum = sum + term;
    sign = -sign;
    pi = Math.pow(32*sum,1.0/3.0);
    n++;
  }
  System.out.println("    pi4 is "+pi+" n is "+n);
  }
}
```

Listing 4:
This Java listing implements the rapidly converging iteration of Equation (6).

```
// implements the modified Newton's method of Eq.(6)
import java.util.*;
class PiNewton{
  public static void main(String Args[]){
      Scanner in = new Scanner(System.in);
      System.out.println("input minimum error in Pi computation - for example .0001");
      double error = in.nextDouble();
  double x0 = 2.0;// intial guess is x0 = 2.0
  double x1 = x0 + sinusoid(x0);
  //Modified Newton iteration is here in the main method
      while (Math.abs((x1-x0)) > error){
        x0 = x1;
        x1 = x0 + sinusoid(x0);
        System.out.println("xo is "+x0+", x1 is "+x1+" Error is "+Math.abs((x1-Math.PI)));
      }
  }

  static double sinusoid(double x1){// this method computes the function sin(x)
    double sin = x1;
    double term = x1;
      for(int n = 3; n < 50; n += 2){
      term *= (-1) * (x1/n)*(x1/(n-1));
      sin += term;
      if(Math.abs(term) < 0.00000000000001) break;//compute the value of sin accurately
as may be needed
    }
    return sin;
  }
}
```