$ Supplement Note 4: Calculation of π via Archimedes' method

The value of $\pi = 3.141592653589793\ldots$ is defined as the ratio of a circle's circumference S to its *diameter* $D = 2r$, where r is the *radius* of the circle:

$$\pi = \frac{S}{D} = \frac{S}{2r} \tag{1}$$

In mathematics, the angle is usually measured in terms of *radian*. Although frequently omitted it is denoted by *rad*. For a circle with radius r, the magnitude in radians for an angle subtended from the center intercepting an arc with a length equal to s is equal to $\theta = \frac{s}{r}$. The circumference of a circle is equal to $S = 2\pi r$. Hence the magnitude in radians of a complete revolution (360°) is equal to 2π. This yields:

$$1 \text{ rad} = \frac{180^\circ}{\pi} \approx 57.3^\circ$$

In Archimedes's method, the circumference S is approximated by the perimeter $S_N = N\, a_N$ of an inscribed N-sided *regular polygon*, i.e. a polygon with equal angles and the length of each side is equal to a_N. As N increases, the perimeter of the polygon approaches to the circle and the value S_N tends to the limit S:

$$S = \lim_{N\to\infty} S_N$$

We assume that the radius is given by $r = 1$. Let us start with $N = 6$:

$$a_6 = 1 \quad \Rightarrow \quad S_6 = 6 \quad \Rightarrow \quad \pi_{\text{approx}} = \frac{S_6}{2r} = 3$$

The circumference for the inscribed regular polygon with doubling number of sides $2N$ may be obtained by using the Pythagorean theorem, (c.f Figure 1):

$$h = \sqrt{1 - \left(\frac{a_N}{2}\right)^2} \quad ; \quad b = 1 - h$$
$$a_{2N} = \sqrt{b^2 + \left(\frac{a_N}{2}\right)^2}$$

This is coded in *Archimedes-pi.py* and some of the results are listed in Table 1. A similar calculation starting with an inscribed square with $N = 4$ and $a_4 = \sqrt{2}r$ is also listed in the code.

n	π_n	error
6	3.0000000000000000	-1.42e-01
12	3.1058285412302489	-3.58e-02
24	3.1326286132812378	-8.96e-03
48	3.1393502030468667	-2.24e-03
96	3.1410319508905098	-5.61e-04
192	3.1414524722854624	-1.40e-04
6144	3.1415925166921577	-1.37e-07
6291456	3.1415926535896630	-1.30e-13
25165824	3.1415926535897856	-7.55e-15

Table 1: calculated value of π

$ **Extrapolation by using polynomials**:

Much more accurate results may be obtain by fitting the set of $k+1$ calculated values of S_N to a polynomial of degrees k for the variable $x \equiv \frac{1}{N}$:

$$S_N = \sum_{i=0}^{k} c_i \left(\frac{1}{N}\right)^i = c_0 + c_1 \frac{1}{N} + c_2 \left(\frac{1}{N}\right)^2 + \ldots + c_k \left(\frac{1}{N}\right)^k$$

The set of $k+1$ *simultaneous linear equations* for the $k+1$ unknown coefficients $coef \equiv \{c_i\}_{i=k,k-1,\ldots,0}$ may be easily solved. This is done in the procedure *numpy.polyfit*. The polynomial function constructed from these coefficients is represented by $p_n(x) = numpy.poly1d(coef)$. The extrapolated value is given by the value of the polynomial at $x = \lim_{N\to\infty} \frac{1}{N} = 0$. Some of the extrapolated results are listed In Table 2. By this simple extrapolation, results with accuracy in the order of 10^{-10} are obtained based on values whose errors are of the order 10^{-4}. Results with higher degrees of accuracy can be obtained either by starting with values of higher degrees of accuracy or by using polynomials with higher degrees. But

one should be careful about numerical cancellations due to the finite degrees of accuracy for numbers represented in the computer.

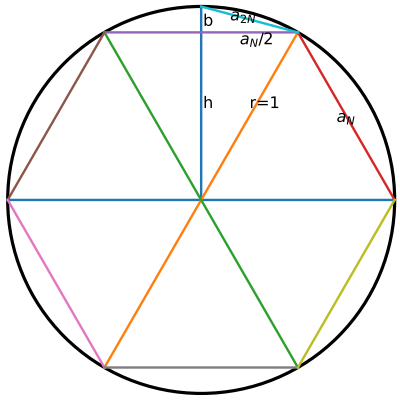

Figure 1: An inscribed regular polygon with N sides

degree of polynomial used	S_N used	extrapolated value of π	error
k = 1	[96, 192]	3.141872993680414	-2.80e-04
k = 2	[48, 96, 192]	3.141592758662502	-1.05e-07
k = 3	[24, 48, 96, 192]	3.1415925336083763	1.20e-07
k = 4	[12, 24, 48, 96, 192]	3.1415926532106053	3.79e-10
k = 5	[6, 12, 24, 48, 96, 192]	3.1415926539778987	-3.88e-10

Table 2: Values of π by extrapolation with polynomials of degree k

```
# Archimedes-pi.py
# Calculation of pi via Archimedes method
import numpy as np
#initialize
niter= [0, 1, 2, 3, 4, 5, 10, 20, 22] # number of iterations whose results
Pi= np.pi                              # were printed out
a, nn= 1 , 6      # initial length and number of sides
```

```
print('pi =', Pi)
print('Calculation starts with n = 6')
n6, pi_6= [],[]  # number of iterations and the calculated results collected
with open('pi.out','w') as fo:  # the file 'pi.out' contains the output
    for iter in range(23):
        p = nn * a /2.0  # calculated result for pi
        pe = p-Pi        # error of the calculated pi
        ss = "n= {0:8d}, pi_n = {1:21.16f}, error= {2:.2e}".format(nn,p,pe)
        if iter in niter:
            n6.append(nn)
            pi_6.append(p)
            print(ss)
            fo.write(ss+'\n')
        nn *=2          # doubling the number of sides
        ah2 = (a/2)**2
        h = np.sqrt(1 - ah2)
        a = np.sqrt((1-h)**2 + ah2) # length of the side for the new polygon
###
print('Calculation starts with n = 4')
a, nn= np.sqrt(2) , 4    # initial length and number of sides
print('pi =', Pi)
for iter in range(23):
    p = nn * a /2.0
    pe = p-Pi
    ss = "n= {0:8d}, pi_n = {1:21.16f}, error= {2:.2e}".format(nn,p,pe)
    if iter in niter: print(ss)
    nn *=2
    ah2 = (a/2)**2
    h = np.sqrt(1 - ah2)
    a = np.sqrt((1-h)**2 + ah2)
## extrapolation by fitting polynomials
n6 = n6[:6]        # datas in the first 6 inputs are used for extrapolation
x6 = np.array(n6)
```

```
x6 = 1/x6          # contains [1/n for the first 6 inputs]
pi6 =pi_6[:6]      # datas in the first 6 inputs are used for extrapolation
print('x6=', x6)
print('pi6=', pi6)
for k in range(1,6):
    print('k=',k)
    coef=np.polyfit(x6[-(k+1):], pi6[-(k+1):],k)
    s='the coefficients of the fitted polynomials $[a_n]_{n=k, k-1,...,0}$\n'
    print(n6[-(k+1):])
    print(s, coef)
    y = np.poly1d(coef)
    print('the extrapolated results', y(0))
    err = Pi-y(0)
    print('error={:.2e}'.format(err))
```