Infinity and Finite Arithmetic

Walter Gander[1]

Department of Computer Science, ETH Zurich, Switzerland
gander@inf.ethz.ch

Abstract. Juraj Hromkovič discusses in his book *Algorithmic Adventures* [4] the notion of infinity. The title of Chapter 3 is *Infinity Is Not Equal to Infinity, or Why Infinity Is Infinitely Important in Computer Science.* Infinity is indeed important in mathematics and computer science. However, a computer is a finite machine. It can represent only a finite set of numbers and an algorithm cannot run forever. Nevertheless we are able to deal with infinity and compute limits on the computer as is shown in this article.

1 Infinity Is Man Made

The popular use of infinity describes something that never ends. An ocean may be said to be infinitely large. The sky is so vast that one might regard it as infinite. The number of stars we see in the sky seems to be infinite. And the time passes by, it never stops, it runs infinitely forever. Christians end the Lord's Prayer mentioning eternity, never stopping time:

> For thine is the kingdom, the power, and the glory, for ever and ever.

However, we know today that all what we experience is *finite*. One liter of water contains 3.343×10^{25} molecules H_2O or 1.003×10^{26} hydrogen and oxygen atoms (according to Wolfram Alpha[1]). Every human is composed of a finite number of molecules, though it may be difficult to actually count them. In principle one could also count the finite number of atoms which form our planet.

The following quotation is attributed to Albert Einstein (though there is no proof for this):

> "Two things are infinite: the universe and human stupidity; and I'm not sure about the universe."

Today scientists believe what Einstein refers to. Wolfram Alpha estimates that the universe contains 10^{80} atoms.

Thus we have to conclude:

Infinity does not exists in nature – it is man made.

[1] https://www.wolframalpha.com/

2 Infinity in Mathematics

The Enzyclopaedia Britannica[2] defines

> **Infinity**, the concept of something that is unlimited, endless, without bound. The common symbol for infinity, ∞, was invented by the English mathematician John Wallis in 1657. Three main types of infinity may be distinguished: the mathematical, the physical, and the metaphysical. Mathematical infinities occur, for instance, as the number of points on a continuous line or as the size of the endless sequence of counting numbers: 1, 2, 3, ... Spatial and temporal concepts of infinity occur in physics when one asks if there are infinitely many stars or if the universe will last forever. In a metaphysical discussion of God or the Absolute, there are questions of whether an ultimate entity must be infinite and whether lesser things could be infinite as well.

The notion of infinity in mathematics is not only very useful but also necessary. Calculus would not exist without the concept of limit computations. And this has consequences in physics. Without defining and computing limits we would e.g. not be able to define velocity and acceleration.

The model for a set with countable infinite elements are the natural numbers $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$. Juraj Hromkovič discusses the well known fact in [4] that the set of all rational numbers

$$\mathbb{Q} = \left\{ \frac{p}{q}, \quad p, q \in \mathbb{N}, q \neq 0 \right\}$$

has the same cardinality as $\mathbb{N}$, thus $|\mathbb{N}| = |\mathbb{Q}|$. On the other hand there are more real numbers even in the interval $[0, 1]$, thus $|\mathbb{N}| < |\mathbb{R}|$. Juraj Hromkovič points in [4] also to the known fact that the real numbers are uncountable and that *there are at least two infinite sets of different sizes.*

We shall not discuss the difference in size of these infinite sets of numbers, rather we will concentrate in the following on computing limits.

3 Infinite Series

Infinite series $\sum_{k=1}^{\infty} a_k$ occur frequently in mathematics and one is interested if the partial sums

$$s_n = \sum_{k=1}^{n} a_k, \quad \lim_{n\to\infty} s_n = ?$$

converge to a limit or not.

It is well known that the *harmonic sum*

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots \tag{1}$$

[2] `https://www.britannica.com/topic/infinity-mathematics#toc252429`

diverges. We can make this plausible by the following argument. We start with the geometric series and integrate:

$$\frac{1}{1-t} = 1 + t + t^2 + t^3 + \cdots, \quad |t| < 1$$
$$\int_0^z \frac{dt}{1-t} = -\log(1-z) = z + \frac{z^2}{2} + \frac{z^3}{3} + \frac{z^4}{4} + \cdots, \quad z < 1.$$

If we let $z \to 1$ then the right hand side tends to the harmonic series, but the left hand side $-\log(1-z) \to \infty$ thus suggests that the harmonic series diverges.

By dividing the last equation by z we obtain

$$\frac{-\log(1-z)}{z} = 1 + \frac{z}{2} + \frac{z^2}{3} + \frac{z^3}{4} + \cdots$$

and by integrating we get the dilogarithm function

$$Li2(z) = \int_0^z \frac{-\log(1-u)}{u}\, du = z + \frac{z^2}{4} + \frac{z^3}{9} + \frac{z^4}{16} + \cdots.$$

For $z = 1$ we get the well known series of the reciprocal squares

$$\int_0^1 \frac{-\log(1-z)}{z}\, dz = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \cdots = \frac{\pi^2}{6}.$$

This series is also the value of the ζ-function

$$\zeta(z) = 1 + \frac{1}{2^z} + \frac{1}{3^z} + \frac{1}{4^z} + \cdots$$

for $z = 2$. By dividing the dilogarithm function by z and integrating we get

$$\int_0^1 \frac{Li2(z)}{z}\, dz = 1 + \frac{1}{2^3} + \frac{1}{3^3} + \frac{1}{4^3} + \cdots = \zeta(3).$$

We can compute the numerical value for $\zeta(3)$ but a nice result as for $\zeta(2)$ is still not known.

4 Infinity and Numerical Computations

4.1 Straightforward Computation Fails

Consider again the harmonic series. The terms converge to zero. Summing up the series using floating point arithmetic such a series converges! The following program sums up the terms of the series until the next term is so small that it does not changes the partial sum `s` anymore.

```
s=1; term=1; k=1;
while s+term ~= s
  k=k+1; term=1/k; s=s+term;
end
```

If we would let this program run on a laptop, we would need to wait a long time till it terminates. In fact it is known (see [5], page 556) that

$$s_n = \sum_{k=1}^{n} \frac{1}{k} = \log(n) + \gamma + \frac{1}{2n} + O\left(\frac{1}{n^2}\right)$$

where $\gamma = 0.57721566490153286\ldots$ is the *Euler-Mascheroni Constant*.

For $n = 10^{15}$ we get $s_n \approx \log(n) + \gamma = 35.116$ and numerically in IEEE arithmetic we have $s_n + 1/n = s_n$. So the harmonic series converges on the computer to $s \approx 35$.

My laptop needs 4 seconds to compute the partial sum s_n for $n = 10^9$. To go to $n = 10^{15}$ the computing time would be $T = 4 \cdot 10^6$ seconds or about 46 days! Furthermore the result would be affected by rounding errors.

Stammbach makes in [7] similar estimates and concludes:

> Das Beispiel ist instruktiv: Bereits im sehr einfachen Fall der harmonischen Reihe ist der Computer nicht in der Lage, den Resultaten der Mathematik wirklich gerecht zu werden. Und er wird es nie können, denn selbst der Einsatz eines Computers, der eine Million Mal schneller ist, bringt in dieser Richtung wenig.

If one uses the straightforward approach, he is indeed right with his critical opinion towards computers. However, it is well-known that the usual textbook formulas often cannot be used on the computer without a careful analysis as they may be unstable and/or the results may suffer from rounding errors. Already George Forsythe [2] noticed that even the popular formula for the solutions of a quadratic equation has to be modified for the use in finite arithmetic. It is the task of numerical analysts to develop robust algorithms which work well on computers.

Let's consider also the series of the inverse squares:

$$\zeta(2) = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \cdots .$$

Here the straightforward summation works in reasonable time, since the terms converge rapidly to zero. We can sum up until the terms become negligible compared with the partial sum:

```
s=1; term=1; n=1;
while s+term ~= s
  n=n+1; term=1/n^2; s=s+term;
end
```

The program terminates on my laptop in 0.4 seconds with $n = 94'906'266$ and $s = 1.644934057834575$. It is well known than numerically it is preferable to sum backwards starting with the smaller terms first. Doing so, with

```
n=94906266; s=0;
for k=n:-1:1
  s=s+1/k^2;
end
```

we get another value $s = 1.644934056311514$. Do the rounding error affect both results so much? Let's compare the results using MAPLE with more precision:

```
Digits:=30:
s:=0:
for k from 1 to 94906266 do
  s:=s+1.0/k^2:
od:
s                 1.64493405631151440597651536455
```

The result is the same as with the backwards summation. But we are far away from the limit value:

$$\frac{\pi^2}{6} - s = 1.5230 \times 10^9$$

Thus also this series cannot be summed up in a straightforward fashion. On the computer it converges too early to a wrong limit.

Numerical analysts have developed summation methods for accelerating convergence. In the following we shall discuss some of theses techniques: *Aitken acceleration*, *extrapolation* and the ε-algorithm.

4.2 Aitkens Δ^2-Acceleration

Let us first consider Aitkens Δ^2-Acceleration [3]. Let $\{x_k\}$ be a sequence which converges linearly to a limit s. This means that for the error $x_k - s$ we have

$$\lim_{k\to\infty} \frac{x_{k+1} - s}{x_k - s} = \rho \neq 0, \quad |\rho| < 1.$$

Thus asymptotically

$$x_n - s \sim \rho\,(x_{n-1} - s) \sim \rho^n\,(x_0 - s)$$

or when solved for x_n we get a model of the asymptotic behavior of the sequence:

$$x_n \sim s + C\rho^n, \quad C = x_0 - s.$$

If we replace "$\sim$" by "$=$" and write the last equation for $n-2, n-1$ and n, we obtain a system of three equations which we can solve for ρ, C and s. Using MAPLE

```
solve({x[n-2]=s+C*rho^(n-2),x[n-2+1]=s+C*rho^(n-1),x[n]=s+C*rho^n},{rho,C,s}):
assign(%): s:=simplify(s);
```

we get the solution

$$s = \frac{x_n x_{n-2} - x_{n-1}^2}{x_n - 2\,x_{n-1} + x_{n-2}}.$$

Forming the new sequence $x'_n = s$ and rearranging, we get

$$x'_n = x_{n-2} - \frac{(x_{n-1} - x_{n-2})^2}{x_n - 2\,x_{n-1} + x_{n-2}} = x_{n-2} - \frac{(\Delta x_{n-2})^2}{\Delta^2 x_{n-2}}$$

which is called *Aitken's Δ^2-Acceleration*. The hope is that the new sequence $\{x'_n\}$ converges faster to the limit.

Assuming that $\{x'_n\}$ converges also linearly we can iterate this transformation and end up with a triangular scheme.

$$\begin{array}{cccc} x_k & x'_k & x''_k & \cdots \\ \hline x_1 & & & \\ x_2 & & & \\ x_3 & x'_1 & & \\ x_4 & x'_2 & & \\ x_5 & x'_3 & x''_1 & \\ \vdots & \vdots & \vdots & \ddots \end{array}$$

Let's apply this acceleration to compute the Euler-Mascheroni Constant γ. First we generate a sequence of $K+1$ partial sums

$$x_k = \sum_{j=1}^{2^k} \frac{1}{j} - \log(2^k), \quad k = 0, 1, \ldots, K.$$

```
function x=HarmonicPartial(K);
% HARMONICPARTIAL computes K+1 partial sums of the harmonic series
% and subtracts the logaritm.
y=[1:2^K]; y=1./y;
s=1; x=s;
for k=1:K
   s=s+sum(y(2^(k-1)+1:2^k)); x=[x s-log(2^k)];
end
x=x';
```

Then we program the function for the iterated Aitken-Scheme:

```
function T=AitkenAcc(x)
% AITKENACC tries to accelerate the convergent sequence x
%           with repeated Aitken-Delta-Square transformations
n=length(x); m=floor((n+1)/2);
T=zeros(n,m);
T(:,1)=x;
for j=2:m
  for k=2*j-1:n
    Delta2=T(k,j-1)-2*T(k-1,j-1)+T(k-2,j-1);
    if Delta2==0, break, end
    T(k,j)=T(k-2,j-1)-(T(k-1,j-1)-T(k-2,j-1))^2/Delta2;
  end
end
```

Now we can call the main program

```
K=8
x=HarmonicPartial(K);
A=AitkenAcc(x)
```

We get

```
1.000000000000000
0.806852819440055
0.697038972213442 0.552329999700925
0.638415601177307 0.571280073489448
0.608140270989212 0.575806621446670 0.577227192023427
0.592759292636793 0.576875788763135 0.577206420206234
0.585007820346097 0.577132432059184 0.577213495239782 0.577211697690028
0.581116828669555 0.577195084788389 0.577215319609806 0.577215953496545
0.579167518337717 0.577210549043978 0.577215616874797 0.577215674740153 0.577215691876342
```

The Aitken extrapolation gives 7 correct decimal digits of γ using only the first 256 terms of the series.

The convergence of the sequence $\{x_k\}$ is linear with the factor $\rho \approx 0.5$ as we can see by computing the quotients $(x_{k+1} - \gamma)/(x_k - \gamma)$

```
(x(2:K)-0.57721566490153)./(x(1:K-1)-0.57721566490153)
```

```
   0.543154359030451
   0.521794077934416
   0.510751519455785
   0.505304547186624
   0.502629772912587
   0.501308676280856
   0.500652713588389
```

4.3 Extrapolation

We follow here the theory given in [3]. Extrapolation is used to *compute limits*. Let h be a discretization parameter and $T(h)$ an approximation of an unknown quantity a_0 with the following property:

$$\lim_{h \to 0} T(h) = a_0. \tag{2}$$

The usual assumption is that $T(0)$ is *difficult to compute* – maybe numerically unstable or requiring infinitely many operations. If we compute some function values $T(h_i)$ for $h_i > 0$, $i = 0, 1, \ldots, n$ and construct the interpolation polynomial $P_n(x)$ then $P_n(0)$ will be an approximation for a_0.

If a limit $a_0 = \lim_{m \to \infty} s_m$ is to be computed then, using the transformation $h = 1/m$ and $T(h) = s_m$, the problem is reduced to $\lim_{h=0} T(h) = a_0$.

To compute the sequence $\{P_n(0)\}$ for $n = 0, 1, 2, \ldots$ it is best to use Aitken-Neville interpolation (see [3]). The hope is that the diagonal of the Aitken-Neville scheme will converge to a_0. This is indeed the case if there exists an asymptotic expansion for $T(h)$ of the form

$$T(h) = a_0 + a_1 h + \cdots + a_k h^k + R_k(h) \quad \text{with} \quad |R_k(h)| < C_k h^{k+1}, \tag{3}$$

and if the sequence $\{h_i\}$ is chosen such that

$$h_{i+1} < c h_i \quad \text{with some} \quad 0 < c < 1,$$

i.e. if the sequence $\{h_i\}$ converges sufficiently rapidly to zero. In this case, the diagonals of the Aitken-Neville scheme converge faster to a_0 than the columns, see [1].

Since we extrapolate for $x = 0$ the recurrence for computing the Aitken-Neville scheme simplifies to

$$T_{ij} = \frac{h_i T_{i-1,j-1} - h_{i-j} T_{i,j-1}}{h_i - h_{i-j}}. \tag{4}$$

Furthermore if we choose the special sequence

$$h_i = h_0 2^{-i}, \tag{5}$$

then the recurrence becomes

$$T_{ij} = \frac{2^j T_{i,j-1} - T_{i-1,j-1}}{2^j - 1}. \tag{6}$$

Note that this scheme can also be interpreted as an algorithm for *elimination of lower order error terms* by taking the appropriate linear combinations. This process is called *Richardson Extrapolation* and is the same as Aitken–Neville extrapolation.

Consider the expansion

$$\begin{aligned} T(h) &= a_0 + a_1 h^2 + a_2 h^4 + a_3 h^6 + \cdots, \\ T\left(\tfrac{h}{2}\right) &= a_0 + a_1 \left(\tfrac{h}{2}\right)^2 + a_2 \left(\tfrac{h}{2}\right)^4 + a_3 \left(\tfrac{h}{2}\right)^6 + \cdots, \\ T\left(\tfrac{h}{4}\right) &= a_0 + a_1 \left(\tfrac{h}{4}\right)^2 + a_2 \left(\tfrac{h}{4}\right)^4 + a_3 \left(\tfrac{h}{4}\right)^6 + \cdots. \end{aligned} \tag{7}$$

Forming the quantities

$$T_{11} = \frac{4T\left(\frac{h}{2}\right) - T(h)}{3} \text{ and } T_{21} = \frac{4T\left(\frac{h}{4}\right) - T\left(\frac{h}{2}\right)}{3},$$

we obtain

$$\begin{aligned} T_{11} &= a_0 - \tfrac{1}{4} a_2 h^4 - \tfrac{5}{16} a_3 h^6 + \cdots, \\ T_{21} &= a_0 - \tfrac{1}{64} a_2 h^4 - \tfrac{5}{1024} a_3 h^6 + \cdots. \end{aligned}$$

Thus we have eliminated the term with h^2. Continuing with the linear combination

$$T_{22} = \frac{16 T_{21} - T_{11}}{15} = a_0 + \frac{1}{64} a_3 h^6 + \cdots$$

we eliminate the next term with h^4.

Often in the asymptotic expansion (3) the odd powers of h are missing, and

$$T(h) = a_0 + a_2 h^2 + a_4 h^4 + \cdots \tag{8}$$

holds. In this case it is advantageous to extrapolate with a polynomial in the variable $x = h^2$. This way we obtain faster approximations of (8) of higher order. Instead of (4) we then use

$$T_{ij} = \frac{h_i^2 T_{i-1,j-1} - h_{i-j}^2 T_{i,j-1}}{h_i^2 - h_{i-j}^2}. \tag{9}$$

Moreover, if we use the sequence (5) for h_i, we obtain the recurrence

$$T_{ij} = \frac{4^j T_{i,j-1} - T_{i-1,j-1}}{4^j - 1}, \tag{10}$$

which is used in the *Romberg Algorithm* for computing integrals.

For the special choice of the sequence h_i according to (5) we obtain the following extrapolation algorithm:

```
function A=ANS(x,factor);
% ANS  Aitken-Neville-Scheme for x,  factor is 2 or 4
K=length(x);
A(1,1)=x(1);
for i=2:K
  A(i,1)=x(i); vhj=1;
  for j=2:i
    vhj=vhj*factor;
    A(i,j)=(vhj*A(i,j-1)-A(i-1,j-1))/(vhj-1);
  end;
end
```

Let's turn now to the series with the inverse squares. We have mentioned before that this series is a special case (for $z = 2$) of the ζ-function

$$\zeta(z) = \sum_{k=1}^{\infty} \frac{1}{k^z}.$$

To compute $\zeta(2)$ we apply the Aitken-Neville scheme to extrapolate the limit of partial sums:

$$s_m = \sum_{k=1}^{m} \frac{1}{k^2}, \quad \zeta(2) = \lim_{n\to\infty} s_m.$$

So far we did not investigate the asymptotic behavior of s_m. Assuming that all powers of $1/m$ are present, we extrapolate with (6).

```
% compute partial sums s_m=\sum_{k=1}^(2^m), k=1,..,K
K=8;
y=[1:2^K]; y=1./y.^2;
for j=0:K-1
  s=sum(y(1:2^j)); x(j+1)=s;
end
x=x';
A=ANS(x,2)
```

We get the following results (we truncated the numbers to save space):

```
1.00000
1.25000 1.50000
1.42361 1.59722 1.629629
1.52742 1.63123 1.642569 1.644418529
1.58434 1.64127 1.644617 1.644909465 1.6449421945
1.61416 1.64398 1.644893 1.644933169 1.6449347501 1.64493451001
1.62943 1.64469 1.644928 1.644934037 1.6449340953 1.64493407423 1.644934067322
1.63715 1.64487 1.644933 1.644934065 1.6449340678 1.64493406692 1.644934066809 1.644934066805390
```

We have used $2^7 = 128$ terms of the series and obtained as extrapolated value for the limit $A_{8,8} = 1.644934066805390$. The error is $\pi^2 - A_{8,8} = 4.28 \cdot 10^{-11}$ so the extrapolation works well.

Asymptotic Expansion of ζ-function Consider the partial sum

$$s_{m-1} = \sum_{k=1}^{m-1} \frac{1}{k^z} = \zeta(z) - \sum_{k=m}^{\infty} \frac{1}{k^z}.$$

Applying the *Euler-MacLaurin Summation Formula* (for a derivation see [3]) to the tail we get the asymptotic expansion

$$\sum_{k=m}^{\infty} \frac{1}{k^z} \sim \frac{1}{z-1}\frac{1}{m^{z-1}} + \frac{1}{2}\frac{1}{m^z} + \frac{1}{z-1}\sum_{j=1} \binom{1-z}{2j} \frac{B_{2j}}{m^{z-1+2j}}.$$

The B_k are the *Bernoulli numbers*:

$$B_0 = 1, B_1 = -\frac{1}{2}, B_2 = \frac{1}{6}, B_4 = -\frac{1}{30}, B_6 = \frac{1}{42}, B_8 = -\frac{1}{30}, \dots$$

and $B_3 = B_5 = B_7 = \ldots = 0$. In general the series on the right hand side does not converge. Thus we get

$$\sum_{k=1}^{m-1} \frac{1}{k^z} + \frac{1}{2}\frac{1}{m^z} \sim \zeta(z) - \frac{1}{z-1}\sum_{j=0} \binom{1-z}{2j} \frac{B_{2j}}{m^{z-1+2j}}.$$

For $z = 3$ we obtain an asymptotic expansion with only even exponents

$$\sum_{k=1}^{m-1} \frac{1}{k^3} + \frac{1}{2}\frac{1}{m^3} \sim \zeta(3) - \frac{1}{2m^2} - \frac{1}{4m^4} + \frac{1}{12m^6} - \frac{1}{12m^8} \pm \cdots \tag{11}$$

And for $z = 2$ we obtain

$$\sum_{k=1}^{m-1} \frac{1}{k^2} + \frac{1}{2}\frac{1}{m^2} \sim \zeta(2) - \frac{B_0}{m} - \frac{B_2}{m^3} - \frac{B_4}{m^5} - \cdots \tag{12}$$

which is an expansion with only odd exponents.

Knowing these asymptotic expansions, there is no need to accelerate convergence by extrapolation. For instance we can choose $m = 1000$ and use (11) to compute

$$\sum_{k=1}^{m-1} \frac{1}{k^3} + \frac{1}{2}\frac{1}{m^3} + \frac{1}{2m^2} + \frac{1}{4m^4} - \frac{1}{12m^6} = 1.202056903159593$$

and obtain $\zeta(3)$ to machine precision.

If, however, knowing only that the expansion has even exponents, we can extrapolate

```
K=8; m=1;
for j=1:K
  s=0;
  for k=1:m-1
    s=s+1/k^3;
  end
  x(j)=s+1/2/m^3;
  m=2*m;
end
A=ANS(x',4)
```

and get $A_{8,8} = 1.202056903159594$, and thus $\zeta(3)$ also to machine precision.

Could we also take advantage if the asymptotic development as e.g. in (12) contains only odd exponents? We need to modify the extrapolation scheme following the idea of Richardson by eliminating lower order error terms. If

$$T(h) = a_0 + a_1 h + a_2 h^3 + a_3 h^5 + a_4 h^7 + \cdots$$

we form the extrapolation scheme

$$\begin{array}{lllll} T_{11} = T(h) & & & & \\ T_{12} = T(h/2) & T_{22} = 2T_{12} - T_{11} & & & \\ T_{31} = T(h/4) & T_{32} = 2T_{32} - T_{12} & T_{33} = \dfrac{2^3 T_{32} - T_{22}}{2^3 - 1} & & \\ \vdots & \vdots & \vdots & \ddots & \end{array}$$

Then T_{k2} has eliminated the term with h and T_{k3} has also eliminated the term with h^3. In general, for $h_i = h_0 2^{-i}$, we extrapolate the limit $\lim_{k\to\infty} x_k$ by initializing $T_{i1} = x_i, i = 1, 2, 3, \ldots$ and

$$T_{ij} = \frac{2^{2j-3} T_{ij-1} - T_{i-1j-1}}{2^{2j-3} - 1}, \quad i = 2, 3, \ldots,\ j = 2, 3, \ldots, i$$

This scheme is computed by the following function:

```
function A=ANSodd(x);
% ANSodd extrapolation for x with only odd exponents
K=length(x);
A(1,1)=x(1);
for i=2:K
  A(i,1)=x(i); vhj=2;
  for j=2:i
    A(i,j)=(vhj*A(i,j-1)-A(i-1,j-1))/(vhj-1);
    vhj=vhj*4;
  end;
end
```

We now extrapolate again the partial sum for the inverse squares:

```
K=8; m=1;
for j=1:8
  s=0;
  for k=1:m-1
    s=s+1/k^2;
  end
  x(j)=s+1/2/m^2;
  m=2*m;
end
A=ANSodd(x)
```

This time we converge to machine precision (we omitted again digits to save space):

```
0.500
1.125 1.750000
1.392 1.659722 1.64682539
1.519 1.646857 1.64502024 1.64496201552
1.582 1.645177 1.64493716 1.64493448049 1.64493426368
1.613 1.644964 1.64493416 1.64493407101 1.64493406778 1.644934067405063
1.629 1.644937 1.64493407 1.64493406688 1.64493406685 1.644934066849075 1.644934066848803
1.637 1.644934 1.64493406 1.64493406684 1.64493406684 1.644934066848226 1.644934066848226 1.644934066848226
```

4.4 The ε-Algorithm

In this section we again follow the theory given in [3]. Aitken's Δ^2-Acceleration uses as model for the asymptotic behavior of the error

$$x_n - s \sim C\rho^n.$$

By replacing "$\sim$" with "$=$" and by using three consecutive iterations we obtained in Section 4.2 a nonlinear system for ρ, C and s. Solving for s we obtain a new sequence $\{x'\}$. A generalization of this was proposed by Shanks [6]. Consider the asymptotic error model

$$x_n - s \sim \sum_{i=1}^{k} a_i \rho_i^n, \quad \text{for } k > 1.$$

Replacing again "$\sim$" with "$=$" and using $2k+1$ consecutive iterations we get a system of nonlinear equations

$$x_{n+j} = s_{n,k} + \sum_{i=1}^{k} a_i \rho_i^{n+j}, \quad j = 0, 1, \ldots, 2k.$$

Assuming we can solve this system, we obtain a new sequence $x'_n = s_{n,k}$. This is called a *Shanks Transformation*.

Solving this nonlinear system is not easy and becomes quickly unwieldy. In order to find a different characterization for the Shanks Transformation, let

$P_k(x) = c_0 + c_1 x + \cdots + c_k x^k$ be the polynomial with zeros $\rho_1, \ldots, \rho_k$, normalized such that $\sum c_i = 1$, and consider the equations

$$\begin{aligned} c_0(x_n - s_{n,k}) &= c_0 \sum_{i=1}^{k} a_i \rho_i^n \\ c_1(x_{n+1} - s_{n,k}) &= c_1 \sum_{i=1}^{k} a_i \rho_i^{n+1} \\ \vdots \qquad &= \qquad \vdots \\ c_k(x_{n+k} - s_{n,k}) &= c_k \sum_{i=1}^{k} a_i \rho_i^{n+k}. \end{aligned}$$

Adding all these equations, we obtain the sum

$$\sum_{j=0}^{k} c_j (x_{n+j} - s_{n,k}) = \sum_{i=1}^{k} a_i \rho_i^n \underbrace{\sum_{j=0}^{k} c_j \rho_i^j}_{P_k(\rho_i)=0},$$

and since $\sum c_i = 1$, the extrapolated value becomes

$$s_{n,k} = \sum_{j=0}^{k} c_j x_{n+j}. \tag{13}$$

Thus $s_{n,k}$ is a linear combination of successive iterates, a weighted average. If we knew the coefficients c_j of the polynomial, we could directly compute $s_{n,k}$.

Wynn established in 1956, see [8], the remarkable result that the quantities $s_{n,k}$ can be computed recursively. This procedure is called the *ε-algorithm*. Let $\varepsilon_{-1}^{(n)} = 0$ and $\varepsilon_0^{(n)} = x_n$ for $n = 0, 1, 2, \ldots$. From these values, the following table using the recurrence relation

$$\varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + \frac{1}{\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}} \tag{14}$$

is constructed:

$$\begin{array}{ccccccc} \varepsilon_{-1}^{(0)} & & & & & & \\ & \varepsilon_0^{(0)} & & & & & \\ \varepsilon_{-1}^{(1)} & & \varepsilon_1^{(0)} & & & & \\ & \varepsilon_0^{(1)} & & \varepsilon_2^{(0)} & & & \\ \varepsilon_{-1}^{(2)} & & \varepsilon_1^{(1)} & & \varepsilon_3^{(0)} & & \\ & \varepsilon_0^{(2)} & & \varepsilon_2^{(1)} & & \cdots & \\ \varepsilon_{-1}^{(3)} & & \varepsilon_1^{(2)} & & \cdots & & \\ & \varepsilon_0^{(3)} & & \cdots & & & \\ \varepsilon_{-1}^{(4)} & & \cdots & & & & \end{array} \tag{15}$$

Wynn showed that $\varepsilon_{2k}^{(n)} = s_{n,k}$ and $\varepsilon_{2k+1}^{(n)} = \frac{1}{S_k(\Delta x_n)}$, where $S_k(\Delta x_n)$ denotes the Shanks transformation of the sequence of the differences $\Delta x_n = x_{n+1} - x_n$. Thus every second column in the ε-table is in principle of interest. For the MATLAB implementation, we write the ε-table in the lower triangular part of the matrix E, and since the indices in MATLAB start at 1, we shift appropriately:

$$\begin{array}{l} 0 = \varepsilon_{-1}^{(0)} = E_{11}, \\ 0 = \varepsilon_{-1}^{(1)} = E_{21} \;\; x_1 = \varepsilon_0^{(0)} = E_{22}, \\ 0 = \varepsilon_{-1}^{(2)} = E_{31} \;\; x_2 = \varepsilon_0^{(1)} = E_{32} \;\; \varepsilon_1^{(0)} = E_{33}, \\ 0 = \varepsilon_{-1}^{(3)} = E_{41} \;\; x_3 = \varepsilon_0^{(2)} = E_{42} \;\; \varepsilon_1^{(1)} = E_{43} \;\; \varepsilon_2^{(0)} = E_{44}. \end{array} \tag{16}$$

We obtain the algorithm

```
function [s,Er]=EpsilonAlgorithm(x);
% EPSILONALGORITHM computes the eps-scheme E for sequence x.
%   Output is the reduced scheme Er (only even columns) and
%   diagonal element s.
n=length(x);
E=zeros(n+1,n+1);
for i=1:n
  E(i+1,2)=x(i);
end
for i=3:n+1
  for j=3:i
    D=E(i,j-1)-E(i-1,j-1);
    if D==0, s=E(i,j-1); return, end
    E(i,j)=E(i-1,j-2)+1/D;
  end
end
Er=E(2:end,2:2:end); s=E(end,end);
```

The performance of the ε-algorithm is shown by accelerating the partial sums of the series

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} \pm \cdots = \ln 2.$$

We first compute the partial sums and then apply the ε-algorithm

```
format short e
k=4;
v=1;
for j=1:2*k+1
  y(j)=v/j; v=-v;
end
x=cumsum(y);
[s,Er]=EpsilonAlgorithm(x);
Er
log(2)-s
```

We obtain the result

```
Er =
   1.0000e+00            0            0            0            0
   5.0000e-01            0            0            0            0
   8.3333e-01   7.0000e-01            0            0            0
   5.8333e-01   6.9048e-01            0            0            0
   7.8333e-01   6.9444e-01   6.9333e-01            0            0
   6.1667e-01   6.9242e-01   6.9309e-01            0            0
   7.5952e-01   6.9359e-01   6.9317e-01   6.9315e-01            0
   6.3452e-01   6.9286e-01   6.9314e-01   6.9315e-01            0
   7.4563e-01   6.9335e-01   6.9315e-01   6.9315e-01   6.9315e-01
>> log(2)-s
ans =  -1.5179e-07
```

It is quite remarkable that we can obtain a result with about 7 decimal digits of accuracy by extrapolation using only partial sums of the first 9 terms, especially since the last partial sum still has no correct digit!

References

1. J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*. Springer, 1991.
2. George E. Forsythe, *How Do You Solve a Quadratic Equation?*. Stanford Technical Report No. CS40, June 16, 1966
3. Walter Gander, Martin J. Gander, Felix Kwok, *Scientific Computing, an Introduction Using* Maple *and* Matlab. Springer Verlag, 2014.
4. Juraj Hromkovič, *Algorithmic Adventures*. Springer Berlin Heidelberg, 2009.
5. Konrad Knopp, *Theorie und Anwendungen der unendlichen Reihen*. Springer Verlag, 1947.
6. Daniel Shanks *Non-linear transformation of divergent and slowly convergent sequences*. Journal of Mathematics and Physics, 1955, Vol. 34, pp. 1–42.
7. Urs Stammbach, *Die harmonische Reihe: Historisches und Mathematisches*. El. Math. 54 (1999), 93-106.
8. P. Wynn, *On a device for computing the $e_m(S_n)$-transformation*. MTAC, 1956, Vol. 10, pp. 91-96.