# LECTURE 1: NUMBER APPROXIMATION

## 1. Numbers

### 1.1. Number sets

Most scientific disciplines introduce an idea of the amount of some entity or property of interest. Furthermore, the amount is usually combined with the concept of a *number*, an abstraction of the observation that the two sets $A=\{\text{Mary, Jane, Tom}\}$ and $B=\{\text{apple, plum, cherry}\}$ seem quite different, but we can match one distinct person to one distinct fruit as in $\{\text{Mary}\leftrightarrow\text{plum, Jane}\leftrightarrow\text{apple, Tom}\leftrightarrow\text{cherry}\}$. In contrast, we cannot do the same matching of distinct persons to a distinct color from the set $\{\text{red, green}\}$, and one of the colors must be shared between two persons. Formal definition of the concept of a number from the above observations is surprisingly difficult since it would be self-referential due to the apperance of the numbers "one" and "two". Leaving this aside, the key concept is that of *quantity* of some property of interest that is expressed through a number.

Several types of numbers have been introduced in mathematics to express different types of quantities, and the following will be used throughout this text:

$\mathbb{N}$. The set of natural numbers, $\mathbb{N}=\{0,1,2,3,\dots\}$, infinite and countable, $\mathbb{N}_+=\{1,2,3,\dots\}$;

$\mathbb{Z}$. The set of integers, $\mathbb{Z}=\{0,\pm1,\pm2,\pm3,\dots\}$, infinite and countable;

$\mathbb{Q}$. The set of rational numbers $\mathbb{Q}=\{p/q, p\in\mathbb{Z}, q\in\mathbb{N}_+\}$, infinite and countable;

$\mathbb{R}$. The set of real numbers, infinite, not countable, can be ordered;

$\mathbb{C}$. The set of complex numbers, $\mathbb{C}=\{x+iy, x, y\in\mathbb{R}\}$, infinite, not countable, cannot be ordered.

These sets of numbers form a hierarchy, with $\mathbb{N}\subset\mathbb{Z}\subset\mathbb{Q}\subset\mathbb{R}\subset\mathbb{C}$. The size of a set of numbers is an important aspect of its utility in describing natural phenomena. The set $S=\{\text{Mary, Jane, Tom}\}$ has three elements, and its size is defined by the *cardinal number*, $|S|=3$. The sets $\mathbb{N},\mathbb{Z},\mathbb{Q},\mathbb{R},\mathbb{C}$ have an infinite number of elements, but the relation

$$z=\begin{cases}-n/2 & \text{for}\, n\,\text{even}\\ (n+1)/2 & \text{for}\, n\,\text{odd}\end{cases}$$

defines a one-to-one correspondence between $n\in\mathbb{N}$ and $z\in\mathbb{Z}$, so these sets are of the same size denoted by the *transfinite number* $\aleph_0$ (aleph-zero). The rationals can also be placed into a one-to-one correspondence with $\mathbb{N}$, hence

$$|\mathbb{N}|=|\mathbb{Z}|=|\mathbb{Q}|=\aleph_0.$$

In contrast there is no one-to-one mapping of the reals to the naturals, and the cardinality of the reals is $|\mathbb{R}|=\mathfrak{c}$ (Fraktur-script c). Georg Cantor established set theory and introduced a proof technique known as the diagonal argument to show that $\mathfrak{c}=2^{\aleph_0}$. Intuitively, there are exponentially more reals than naturals.

### 1.2. Quantification

One of the foundations of the scientific method is *quantification*, ascribing numbers to phenomena of interest. To exemplify the utility of different types of number to describe natural phenomena, consider common salt (sodium chloride, Fig. 1) which has the chemical formula NaCl with the sodium ions ($Na^+$) and chloride ions ($Cl^-$) spatially organized in a cubic lattice, with an edge length $a=5.6402$ Å (1 Å $=10^{-10}$ m) between atoms of the same type. Setting the origin of a Cartesian coordinate system $Oxyz$ at a sodium atom, the position of some atom within the lattice is

$$(x,y,z)=\left(i\frac{a}{2},j\frac{a}{2},k\frac{a}{2}\right).$$

Sodium atoms are found positions where $i+j+k$ is even, while chloride atoms are found at positions where $i+j+k$ is odd. The Cartesian coordinates $(x,y,z)$ describe some arbitrary position in space, which is conceptualized as a continuum and placed into one-to-one correspondence with $\mathbb{R}^3$. A particular lattice position can be specified simply through a label consisting of three integers $(i,j,k)\in\mathbb{Z}^3$. The position can be recovered through a *scaling operation*

$$(x,y,z)=\frac{a}{2}(i,j,k),$$

and the number $a/2 \in \mathbb{R}$ that modifies the length scale from 1 to $a/2$, it is called a *scalar*.

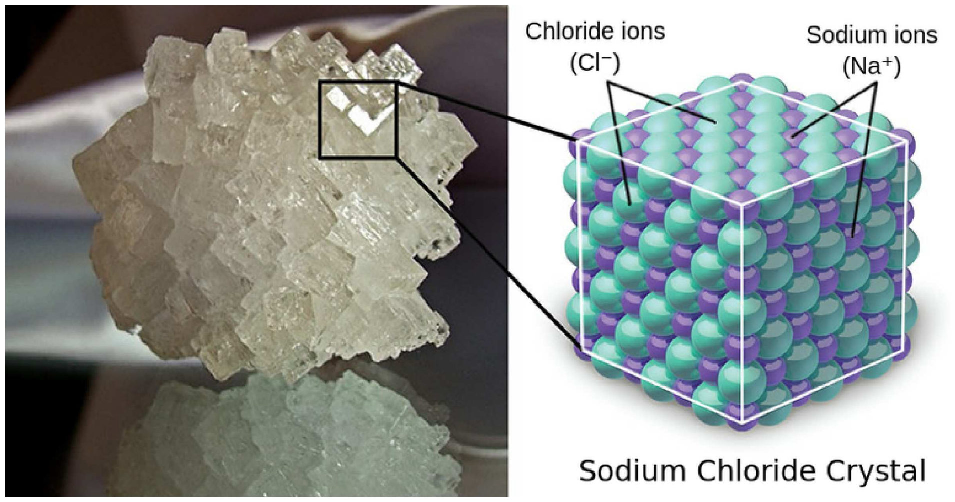

**Figure 1.** Left: Polycrystalline sodium chloride. Right: Cubic lattice structure of a single sodium chloride crystal

## 1.3. Computer number sets

A computer has a finite amount of memory, hence cannot represent all numbers, but rather subsets of the above number sets. Current digital computers internally use numbers represented through binary digits, or bits. Many computer number types are defined for specific purposes, and are often encountered in applications such as image representation or digital data acquisition. Here are the main types.

**Subsets of $\mathbb{N}$.** The number types `uint8`, `uint16`, `uint32`, `uint64` represent subsets of the natural numbers (unsigned integers) using 8, 16, 32, 64 bits respectively. An unsigned integer with $b$ bits can store a natural number in the range from 0 to $2^b - 1$. Two arbitrary natural numbers, written as $\forall i, j \in \mathbb{N}$ can be added and will give another natural number, $k = i + j \in \mathbb{N}$. In contrast, addition of computer unsigned integers is only defined within the specific range 0 to $2^b - 1$. If $k > 2^b - 1$, the result might be displayed as the maximum possible value or as $k \bmod 2^b$.

```
∴ i=UInt8(15); j=UInt8(10); k=i+j
```

25

```
∴ i=UInt8(150); j=UInt8(200); k=i+j; [k i+j mod(350,256)]
```

$$[\ 94\ \ 94\ \ 94\ ] \tag{1}$$

```
∴ i=UInt8(150); j=UInt8(200); k=i-j; [k i-j mod(-50,256)]
```

$$[\ 206\ \ 206\ \ 206\ ] \tag{2}$$

```
∴ typeof(i-j)
```

```
UInt8
```

```
∴
```

**Subsets of $\mathbb{Z}$.** The number types `int8`, `int16`, `int32`, `int64` represent subsets of the integers. One bit is used to store the sign of the number, so the subset of $\mathbb{Z}$ that can be represented is from $1 - 2^{b-1}$ to $2^{b-1} - 1$.

```
∴ i=Int8(15); j=Int8(21); k=i+j
```

36

```
∴ i=Int8(100); j=Int8(101); k=i+j; [k i+j mod(201,128)-128]
```

$$[\,-55\;\;-55\;\;-55\,] \tag{3}$$

```
∴ typeof(k)
```

```
Int8
```

```
∴ [typemin(Int8) typemax(Int8)]
```

$$[\,-128\;\;127\,] \tag{4}$$

```
∴
```

**Subsets of $\mathbb{Q},\mathbb{R},\mathbb{C}$.** Computers approximate the real numbers through the set $\mathbb{F}$ of *floating point numbers*. Floating point numbers that use $b=32$ bits are known as *single precision*, while those that use $b=64$ are *double precision*. A floating point number $x\in\mathbb{F}$ is stored internally as $x=\pm .B_1B_2\dots B_m\times 2^{\pm b_1b_2\dots b_e}$ where $B_i$, $i=1,\dots,$ $m$ are bits within the *mantissa* of length $m$, and $b_j$, $j=1,\dots,e$ are bits within the *exponent*, along with signs $\pm$ for each. The default number type is usually double precision, more concisely referred to `Float64`. Common irrational constants such as $e$, $\pi$ are predefined as irrationals and casting to `Float64` or `Float32` gives floating point approximation. Unicode notation is recognized. Specification of a decimal point indicates a floating point number; its absence indicates an integer.

```
∴ pi
```

$\pi$

```
∴ typeof(pi)
```

```
Irrational{:π}
```

```
∴ [Float32(pi) Float64(pi) Float64(π)]
```

$$[\,3.1415927410125732\;\;3.141592653589793\;\;3.141592653589793\,] \tag{5}$$

```
∴ a=2.3; b=2; c=3.; [typeof(a) typeof(b) typeof(c)]
```

```
DataType[Float64 Int64 Float64]
```

```
∴
```

The approximation of the reals $\mathbb{R}$ by the floats $\mathbb{F}$ is characterized by: `floatmax()`, the largest float, `floatmin` the smallest positive float, and `eps()` known as *machine epsilon*. Machine epsilon highlights the differences between floating point and real numbers since it may be informally defined as the smallest number of form $\epsilon=2^k\in\mathbb{F}$ that satisfies $1+\epsilon\neq 1$. If $\varepsilon\in\mathbb{R}$ of course $1+\varepsilon=1$ implies $\varepsilon=0$, but floating points exhibit "granularity", in the sense that over a unit interval there are small steps that are indistinguishable from zero due to the finite number of bits available for a float leading to $1+\epsilon/2$ being indistiguishable from 1, and the apparently endless loop shown below actually terminates.

```
∴ eps=1.0;
```

```
∴ while (1.0+0.5*eps != 1.0)
    global eps;
    eps=0.5*eps;
  end
```

```
∴ eps
```

$2.220446049250313\,e-16$

The granularity of double precision expressed by machine epsilon is sufficient to represent natural phenomena, and floating point errors can usually be kept under control,

```
∴ [floatmin(Float32) floatmax(Float32) eps(Float32)]
```

$$[\ 1.1754944e-38\ \ 3.4028235e38\ \ 1.1920929e-7\ ] \tag{6}$$

```
∴ [floatmin(Float64) floatmax(Float64) eps(Float64)]
```

$$[\ 2.2250738585072014e-308\ \ 1.7976931348623157e308\ \ 2.220446049250313e-16\ ] \tag{7}$$

```
∴
```

Keep in mind that perfect accuracy is a mathematical abstraction, not encountered in nature. In fields such as sociology or psychology 3 digits of accuracy are excellent, in mechanical engineering this might increase to 6 digits, or in electronic engineering to 8 digits. The most precisely known physical constant is the Rydberg constant known to 12 digits, hence a mathematical statement such as

$$x = 2.6309283450461248350319486319845$$

is unlikely to have any real significance, while

$$x = 2.631 \pm 0.0005$$

is much more informative.

Within the reals certain operations are undefined such as $1/0$. Special float constants are defined to handle such situations: `Inf` is a float meant to represent infinity, and `NaN` ("not a number") is meant to represent an undefinable result of an arithmetic operation.

```
∴ [1/0 -1.0/0.0 1/Inf -1/Inf Inf/Inf]
```

$$[\ \infty\ \ -\infty\ \ 0.0\ \ -0.0\ \ NaN\ ] \tag{8}$$

```
∴
```

Complex numbers $z \in \mathbb{C}$ are specified by two reals, in Cartesian form as $z = x + iy$, $x, y \in \mathbb{R}$ or in polar form as $z = \rho e^{i\theta}$, $\rho, \theta \in \mathbb{R}$, $\rho \geqslant 0$. The computer type complex is similarly defined from two floats and the additional constant I is defined to represent $\sqrt{-1} = i = e^{i\pi/2}$. Functions are available to obtain the real and imaginary parts within the Cartesian form, or the absolute value and argument of the polar form.

```
∴ z1=1+1im; z2=1-im; [z1+z2 z1/z2]
```

$$[\ 2.0 + 0.0i\ \ -0.0 + 1.0i\ ] \tag{9}$$

```
∴ [real(z1) real(z2) real(z1+z2) real(z1/z2)]
```

$$[\ 1.0\ \ 1.0\ \ 2.0\ \ -0.0\ ] \tag{10}$$

```
∴ [imag(z1) imag(z2) imag(z1+z2) imag(z1/z2)]
```

$$[\ 1.0\ \ -1.0\ \ 0.0\ \ 1.0\ ] \tag{11}$$

```
∴ [abs(z1) abs(z2) abs(z1+z2) abs(z1/z2)]
```

$$[\ 1.4142135623730951\ \ 1.4142135623730951\ \ 2.0\ \ 1.0\ ] \tag{12}$$

```
∴ [angle(z1) angle(z2) angle(z1+z2) angle(z1/z2)]
```

$$[\ 0.7853981633974483\ \ -0.7853981633974483\ \ 0.0\ \ 1.5707963267948966\ ] \tag{13}$$

```
∴
```

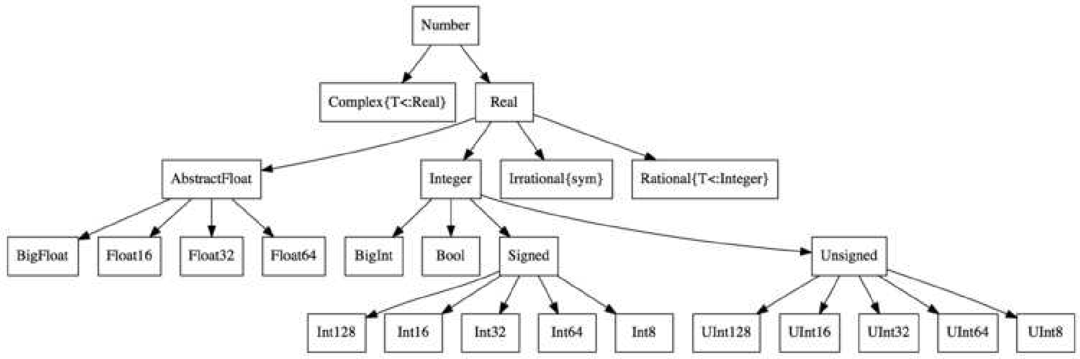

**Figure 2.** Hierarchy of number types in the `Julia` language.

## 2. Approximation

### 2.1. Axiom of floating point arithmetic

The reals $\mathbb{R}$ form an algebraic structure known as a field $(\mathbb{R}, +, \cdot)$. The set of floats together with floating point addition and multiplication are denoted as $(\mathbb{F}, \oplus, \odot)$. Operations with floats do not have the same properties as the reals, but are assumed to have a relative error bounded by machine epsilon

$$\forall x, y \in \mathbb{R}, \left| \frac{\mathrm{fl}(x) \circledast \mathrm{fl}(y) - x * y}{x * y} \right| \leqslant \epsilon, (\circledast, *) \in \{(\oplus, +), (\odot, \cdot)\},$$

where $\mathrm{fl}(x) \in \mathbb{F}$ is the floating point representation of $x \in \mathbb{R}$. The above is restated

$$\mathrm{fl}(x) \circledast \mathrm{fl}(y) = (x * y)\ (1 + \varepsilon), |\varepsilon| \leqslant \epsilon,$$

and accepted as an axiom for use in error analysis involving floating point arithmetic. Computer number sets are a first example of *approximation*: replacing some complicated object with a simpler one. It is one of the key mathematical ideas studied throughout this text.

### 2.2. Cummulative floating point operations

Care should be exercised about the cummulative effect of many floating point operations. An informative example is offered by Zeno's paradox of motion, that purports that fleet-footed Achilles could never overcome an initial head start of $D = 2$ given to the lethargic Tortoise since, as stated by Aristotle:

> In a race, the quickest runner can never over-take the slowest, since the pursuer must first reach the point whence the pursued started, so that the slower must always hold a lead.

The above is often formulated by considering that the first half of the initial head start must be overcome, then another half and so on. The distance traversed after $N$ such steps is

$$D_N = 1 + \frac{1}{2} + \cdots + \frac{1}{2^N} = \frac{1 - (1/2)^{N+1}}{1 - (1/2)} = 2\left[1 - (1/2)^{N+1}\right] < 2.$$

Calculus resolves the paradox by rigorous definition of the limit $D = \lim_{N \to \infty} D_N = 2$ and definition of velocity as $v(t) = \lim_{\delta t \to 0} (D(t + \delta t) - D(t)) / \delta t$, $\delta t = 1/N$, $D(t) = 2[1 - (1/2)^{t/\delta t}]$.

Undertake a numerical invesigation and consider two scenarios, with increasing or decreasing step sizes

$$D_N = 1 + \frac{1}{2} + \cdots + \frac{1}{2^N}, C_N = \frac{1}{2^N} + \frac{1}{2^{N-1}} + \cdots + 1.$$

In $(\mathbb{R}, +, \cdot)$ associativity ensures $D_N = C_N$.

```
∴ N=10; D=2.0 .^ (0:-1:-N); C=2.0 .^ (-N:1:0); sum(D)==sum(C)
```

*true*

```
∴ N=20; D=2.0 .^ (0:-1:-N); C=2.0 .^ (-N:1:0); sum(D)==sum(C)
```

*true*

```
∴
```

Irrespective of the value for $N$, $D_N = C_N$ in floating point arithmetic. Recall however that computers use binary representations internally, so division by powers of two might have unique features (indeed, it corresponds to a bit shift operation). Try subdividing the head start by a different number, perhaps $\pi$ to get an "irrational" numerical investigation of Zeno's paradox of motion. Define now the distance $S_N$ traversed by step sizes that are scaled by $1/\pi$ starting from one to $T_N$, traversed by step sizes scaled by $\pi$ starting from $\pi^{-N}$

$$S_N = 1 + \frac{1}{\pi} + \frac{1}{\pi^2} + \cdots + \frac{1}{\pi^N}, T_N = \frac{1}{\pi^N} + \frac{1}{\pi^{N-1}} + \cdots + 1.$$

Again, in the reals the above two expressions are equal, $S_N = T_N$, but this is no longer verified computationally for all $N$, not even within a tolerance of machine epsilon.

```
∴ fpi=Float64(pi);
```

```
∴ N=10; S=fpi .^ (0:-1:-N); T=fpi .^ (-N:1:0); sum(S)==sum(T)
```

*true*

```
∴ N=20; S=fpi .^ (0:-1:-N); T=fpi .^ (-N:1:0); sum(S)==sum(T)
```

*false*

```
∴ sum(S)-sum(T)<eps(Float64)
```

*false*

```
∴
```

This example gives a first glimpse of the steps that need to be carried out in addition to mathematical analysis to fully characterize an algorithm. Since $S_N \neq T_N$, a natural question is whether one is more accurate than the other. For some arbitrary ratio $a$, the exact value is known

$$E_N = \frac{1-(1/a)^{N+1}}{1-(1/a)},$$

and can be used to evaluate the errors $|S_N - E_N|$, $|T_N - E_N|$.

```
∴ function E(N,a)
      (1-(1/a)^(N+1))/(1-(1/a))
  end;
```

```
∴ function εs(N,a)
     S=a .^ (0:-1:-N)
     abs(sum(S)-E(N,a))
  end;
```

```
∴ function εt(N,a)
     T=a .^ (-N:1:0)
     abs(sum(T)-E(N,a))
  end;
```

```
∴
```

Carrying out the computations leads to results in Fig. 3.

```
∴ n=30; errs=zeros(Float64,n); errt=zeros(Float64,n);
∴ for i=1:n
    errs[i]=εs(N,fpi); errt[i]=εt(N,fpi);
  end
```

```
∴ clf(); plot(1:n,errs,1:n,errt,marker="o"); title("Summation␣error");
  grid("on"); xlabel("n"); ylabel("εs,εt"); legend(["εs"; "εt"]);
```

```
∴
```

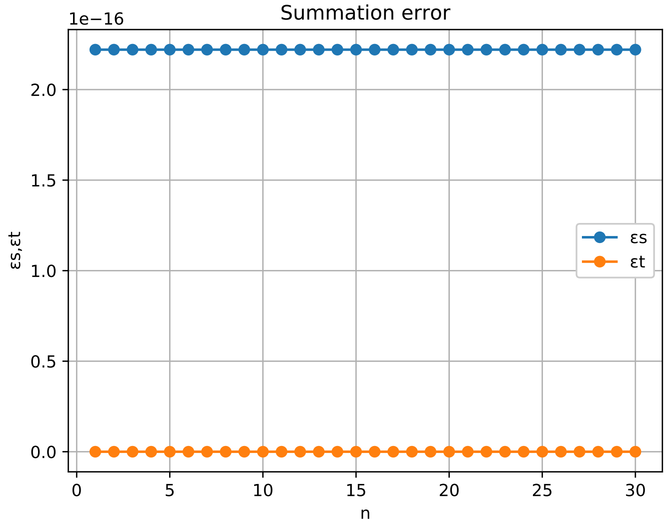

**Figure 3.** Summation order errors.

Note that errors are about the size of machine epsilon for $S_N$, but are zero for $T_N$, it seems that the summation ordering $T_N = a^{-N} + a^{-N+1} + \cdots + 1$ gives the exact value. A bit of reflection reinforces this interpretation: first adding small quantities allows for carry over digits to be accounted for.

This example is instructive beyond the immediate adage of "add small quantities first". It highlights the blend of empirical and analytical approaches that is prevalent in scientific computing.

## 3. Successive approximations

### 3.1. Sequences in $\mathbb{R}$

Single values given by some algorithm are of little value in the practice of scientific computing. The main goal is the construction of a sequence of approximations $\{x_n\}_{n\in\mathbb{N}}$ that enables assessment of the quality of an approximation. Recall from calculus that $\{x_n\}_{n\in\mathbb{N}}$ converges to $x$ if $|x_n - x|$ can be made as small as desired for all $n$ beyond some threshold. In precise mathematical language this is stated through:

DEFINITION. *$\{x_n\}_{n\in\mathbb{N}}$ converges to $x$ if $\forall \varepsilon > 0$, $\exists N(\varepsilon)$ such that $|x_n - x| < \varepsilon$ for $n > N(\varepsilon)$.*

Though it might seem natural to require a sequence of approximations to converge to an exact solution $x$

$$\lim_{n\to\infty} x_n = x,$$

such a condition is problematic on multiple counts:

1. the exact solution is rarely known;
2. the best approximation might be achieved for some finite range $n_1 \leqslant n \leqslant n_2$, rather than in the $n \to \infty$ limit.

Both situations arise when approximating numbers and serve as useful reference points when considering approximation other types of mathematical objects such as functions. For example, the number $\pi$ is readily defined in geometric terms as the ratio of circle circumference to diameter, but can only be approximately expressed by a rational number, e.g., $\pi \cong 22/7$. The exact value of $\pi$ is only obtained as the limit of an infinite number of operations with rationals. There are many such infinite representations, one of which is the Leibniz series

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots\ldots$$

No finite term

$$L_n = \sum_{k=0}^{n} \frac{(-1)^k}{2k+1}$$

of the above Leibniz series equals $\pi/4$, i.e.,

$$\nexists n \in \mathbb{N} \text{ such that } L_n = \pi/4.$$

Rather, the Leibniz series should be understood as an algorithm, i.e., a sequence of elementary operations that leads to succesively more accurate approximations of $\pi/4$

$$\lim_{n\to\infty} L_n = \pi/4.$$

Complex analysis provides a convergence proof starting from properties of the arctan function

$$\frac{\mathrm{d}}{\mathrm{d}z}\arctan(z) = \frac{1}{1+z^2} \Rightarrow \frac{\pi}{4} = \arctan(1) - \arctan(0) = \int_0^1 \frac{\mathrm{d}z}{1+z^2}.$$

For $|z| < 1$ the sequence $S_n = \sum_{k=0}^{n} (-z^2)^k$ of partial sums of a geometric series converges uniformly

$$\sum_{k=0}^{\infty} (-z^2)^k = \lim_{n\to\infty} \frac{1-(-z^2)^n}{1-(-z^2)} = \frac{1}{1+z^2},$$

and can be integrated term by term to give

$$\frac{\pi}{4} = \int_0^1 \sum_{k=0}^{\infty} (-z^2)^k \,\mathrm{d}z = \sum_{k=0}^{\infty} \int_0^1 (-z^2)^k \,\mathrm{d}z = \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}.$$

This elegant result does not address however the points raised above: if $\pi$ were not known, how could the convergence of the sequence $\{L_n\}_{n\in\mathbb{N}}$ be assessed? A simple numerical experiment indicates that the familiar value of $\pi$ is only recovered for large $n$, with 10000 terms insufficient to ensure five significant digits.

```
∴ function L(n)
    L=1.0; s=-1.0
    for k=1:n
      L += s/(2*k+1); s = -s
    end
    return 4*L
  end
```

```
L
```

```
∴ [L(100) L(1000) L(10000) Float64(π)]
```

```
[ 3.1514934010709914  3.1425916543395442  3.1416926435905346  3.141592653589793 ]        (14)
```

```
∴
```

### 3.2. Cauchy sequences

Instead of evaluating distance to an unknown limit, as in $|L_n - \pi| < \varepsilon$, one could evaluate if terms get closer to one another as in $|L_n - L_m| < \varepsilon$, a condition that can readily be checked in an algorithm.

DEFINITION. *$\{x_n\}_{n\in\mathbb{N}}$ is a Cauchy sequence if $\forall \varepsilon > 0$, $\exists N(\varepsilon)$ such that $|x_n - x_m| < \varepsilon$ for all $m, n > N(\varepsilon)$.*

Note that the distance between *any* two terms after the threshold $N(\varepsilon)$ must be smaller than an arbitrary tolerance $\varepsilon$. For example the sequence $a_n = \sqrt{n}$ is not a Cauchy sequence even though the distance between successive terms can be made arbitrarily small

$$a_{n+1} - a_n = \sqrt{n+1} - \sqrt{n} = \frac{\left(\sqrt{n+1} - \sqrt{n}\right)\left(\sqrt{n+1} + \sqrt{n}\right)}{\sqrt{n+1} + \sqrt{n}} = \frac{1}{\sqrt{n+1} + \sqrt{n}} < \frac{1}{2\sqrt{n}}.$$

Verification of decreasing successive distance is therefore a necessary but not sufficient condition to assess whether a sequence is a Cauchy sequence. Furthermore, the distance between successive iterates is not necessarily an indication of the distance to the limit. Reprising the Leibniz example, successive terms can be further apart than the distance to the limit, though terms separated by 2 are closer than the distance to the limit (a consequence of the alternating Leibniz series signs)

```
∴ n=1000; [log10(abs(L(n)-L(n-1))) log10(abs(L(n)-π))]
```

$$[\ -2.6991870973082537 \quad -3.000434185835426\ ] \tag{15}$$

```
∴  [log10(abs(L(n)-L(n-2))) log10(abs(L(n)-π))]
```

$$[\ -5.698969895788488 \quad -3.000434185835426\ ] \tag{16}$$

```
∴
```

Another question is whether a Cauchy sequence is itself convergent. For sequences of reals this is true, but the Leibniz sequence furnishes a counterexample since it contains rationals and converges to an irrational. Such aspects that arise in number approximation sequences become even more important when considering approximation sequences composed of vectors or functions.

### 3.3. Sequences in $\mathbb{F}$

Consideration of floating point arithmetic indicates adaptation of the mathematical concept of convergence is required in scientific computation. Recall that machine epsilon $\epsilon$ is that largest number such that $1+\epsilon = 1$ is true, and characterizes the granularity of the floating point system. A reasonable adaptation of the notion of convergence might be:

DEFINITION. $\{x_n\}_{n\in\mathbb{N}}$, $x_n \in \mathbb{F}$ *converges to* $x \in \mathbb{F}$ *if* $\forall \varepsilon > \epsilon$, $\exists N(\varepsilon)$ *such that* $|x_n - x| < \varepsilon$ *for* $n > N(\varepsilon)$.

What emerges is the need to consider a degree of uncertainty in an approximating sequence. If the uncertainty can be bounded to the intrinsic granularity of the number system, a good approximation is obtained.

**Summary.** The problem of approximating numbers uncovers generic aspects of scientific computing:

- different models of some phenomenon are possible and it is necessary to establish correspondence between models and of a model to theory;
- scientific computation seeks to establish viable approximation techniques for the mathematical objects that arise in models;
- correspondence of a model to theory is established through properties of approximation sequences, not single results of a particular approximation technique;
- physical limitations of computer memory require revisiting of mathematical concepts to characterize approximation sequence behavior, and impart a stochastic aspect to approximation techniques;
- computational experiments are a key aspect, giving an empirical aspect to scientific computing that is not found in deductive or analytical mathematics.