Contents

List of Figures

1 Abstract

The *golden ratio* denotes the divine proportionality for a given whole value and its subdivison into two parts, the *larger* (major) and the *smaller* (minor), where the smaller is to the larger, as the larger is to the whole.

There exists a very long lasting history, (some say since the beginning of math, which they associate with Thales, with Thales having no teacher), and most likely a vast literature, and most probably an infinite number of applications in all fields of the math, arts and nature on the fascinating topic.

The given paper shall document some own experimention and considertion on the topic.

Figure 1: The Coast of Seven Leagues - Hokusai

2 Introduction

The golden ratio and the according formulae are well documented, but since there are two approaches, both expressing the same correspondence, or proportionality, I shall describe and define them here for proper understanding and reference. The two approaches can be summarized as, (a) the relation of the smaller s to the larger l, and (b) the reciprocal relation of the larger l to the smaller s, with s and l forming a whole w.

I want to call this binary of the two approaches the **golden ratio endianess**, naming the version starting at the big end - the larger to the smaller (which is usually expressed with the symbol Φ, greek uppercase letter Phi) - the golden ratio big endian, and the other version, starting at the small end - the smaller to the larger (symbol $\varphi = 1/\Phi$, greek lowercase letter Phi) - the golden ratio little endian.

Endianess	Symbol	Relation	Correspondence
BIG	Φ	l/s	w/l
LITTLE	φ	s/l	l/w

There exist search algorithms, which, for the purpose of optimization, explicitely make use of the golden ratio for asymptotic approach, for hash tables or finding extremum. Unlike those, which take the golden ratio constant as a given input, I want to introduce an iterative and recursive algorithm for the asymptotic approximation of the value of the golden ratio constant value itself. The proposed algorithm I did not find in my google search on the topic, (which does not mean it is not already out there somewhere), so I thought I write it down here.

Since the golden ratio constant is an irrational number, in reality there do not exist two specific and exact values for a larger l and a smaller s, which then, in their relationship, would be equal to the golden ratio constant Φ (or φ), nor is there a real number, exactly denoting the golden ratio constant value Φ (or φ). So there can be only an approximation, having two values l and s together building a whole w, with their relationship coming close to the idea or the concept of Φ.

Φ is a well defined concept of a special relationship between two numbers, which is a **partitioning concept** for dividing a given whole into two parts. In reality, this pair of two numbers for the size of the parts l and s of a whole w, meeting the definitions of the concept Φ, do not exist. Any two numbers existing in real can meet the definitions of the concpet Φ to some degree, but not for 100%. This degree, up to which the concept is met, is then the **tolerance**, or the **precision**.

It is almost, as if math were saying, there is an idea of a golden relationship with particular and well defined characteristics, but in reality it cannot be expressed with nominal values, yet the very existence can be constructed and demonstrated at all possible, and thinkable representations of geometry and nature. We have prove of the existence of a phenomen, but our language is not good or fine enough, to give it an exact representation.

This should not come as a surprise, the opposite, this is the very nature of any model making - using formal or natural language, symbols, terms, definition, syntax, grammer and semantics - and by that is exemplary, and reminds us, not to get carried away, if something is expressed with mathematical or scientific accuracy. Nevertheless, or even because of the irrationality, does the golden ratio express an outstanding beauty and draws our attention and admiration since, it seems, the beginning of the universe, with its numerous manifestations and its intrinsic, mathematical, particular setup.

The most famous formula for Φ

$$\frac{l}{s} \equiv \Phi \equiv \frac{1+\sqrt{5}}{2} \approx 1.6180\ldots \tag{2.0.1}$$

utilizes another irrational number $\sqrt{5}$ for the expression of the irrational number Φ, and has a very beautiful geometrical derivation - compass and straightedge construction - shown in wikipedia.

Having a square-root (of 5), implies, that something has been sqaured before, i.e. in order to solve this one-dimensional problem, someone had to enter another dimension, by squaring a value you create an area (two dimensions, 2d) from a line (one dimension, 1d), to solve the problem there (in 2d), and now comes back to the original first and one dimension (1d), providing us the solution. This is a beautiful aspect of math and problem solving, to enter a different dimension or domain, calculate there - usually more efficient -, and come back to the original dimension, or domain, with a given solution. One example for that approach is for multiplication of numbers to actually use addition of the logarithm. For now, for the sake of ease, for not getting too complicated too quickly, I propose a solution in the same dimension as the problem occurs, one-dimensional (1d) that is.

The algorithm presented in the following of this document will use the principles outlined in the concept Φ, to measure any existing, inevitable deviation or difference from the concept, and will use the magnitude and the direction (the sign of the number) of the difference to the concept as an input and magnitude for correction of the parts smaller s and larger l, for an approximation towards the concept ideal, with a predefined tolerance τ, until a desired precision is reached. Since it is clear that 100% cannot be reached, the precision must be below 100%, e.g. at 99.999%.

An error function will be used to measure and quantify the difference to the partitioning concept Φ.

3 Definitions

3.1 Golden Ratio Endianess Definition

The **Golden Ratio Endianess** shall denote the two existing approches, to express the same correspondence or principle, known as the golden ratio.

Existing literature on math does not use *endianess* to distinguish the two approaches, (endianess is used to specify a byte order or bit order for computer memory). Usually the two approaches are distinguished by using lower case greek letter Phi φ for one, and the uppercase greek letter Phi Φ for the other. The *endianess* I introduce to explicitly distinguish the two approaches by name, rather than by symbol alone, which I will need for my argumentation.

Starting at the low end, relating the smaller s to the larger l, denoted by the symbol φ, shall be called the **golden ratio little endian** notation.

$$LITTLE\ ENDIAN\ :\ \varphi = \frac{s}{l} \tag{3.1.1}$$

Starting at the big end, relating the larger l to the smaller s, denoted by the symbol Φ, shall be called the **golden ratio big endian**, or the **partitioning big endian** notation.

$$BIG\ ENDIAN\ :\ \Phi = \frac{l}{s} \tag{3.1.2}$$

Some references use the symbols the exact other way around, be it irony or is it insesibility of the mathematicians I do not know. Anyhow, the endianess (big or little) for golden ratio does not exist in literature, it is my invention for the purpose of the discussion here. Using uppercase and lowercase letters Phi for one or the other proportion of the parts s and l is common.

3.2 Golden Ratio Definition - Little Endian

Starting at the low end, with the little endian notation, relating the smaller s to the larger l, the golden ratio is an equation as such:

$$LITTLE\ ENDIAN\ :\ \frac{s}{l} \equiv \frac{l}{w} \equiv \varphi \tag{3.2.1}$$

where s is the smaller, l is the larger, and w is the whole

$$w = s + l \tag{3.2.2}$$

and

$$0 < s < l < w \tag{3.2.3}$$

Given is the *partitioning* of the whole w into a smaller s and a larger l.

Given are two *proportions* denoting the (relative) size of the part with, either a reference to the other part, or a reference to the whole. Both proportions tell us about the structure of the partitioning. One proportion implies the other, changing one, inevitable also changes the other, in reciprocal manner. Increasing the smaller makes the larger less, increasing the larger reduces the smaller, when compared to the other, or when compared to the whole, no matter if the whole stays the same, or changes with changing parts.

Changing one of the two parts, changes either the other part or the whole. Changing one of the two proportions, requires the re-calculation of the other proportion.

Any one of the two given proportions by its own already fully specifies the whole setup, the partitioning, or subdivions of a whole into two parts, a smaller and a larger, with symmetry as a special case.

With a given value (specification) for s/l, we also, implicitely understand and know l/w, and, viceversa, for a given value l/w, we can derive s/l.

We usually would not specify the given setup by both proportions, but rather specify one and then derive the other. This may seem obvious, but in the case of the golden ratio this is what happens, we get both specifications at once in one equation. The right side of the equation (w/l) tells us what s/l shall be, the left side of the equation (s/l) tells us what w/l shall be. And worse: the definition is self referential, since changing s (to comply to the specification), will also change l, (which is part of the specification).

I see how this setup is trembling, without ever coming to equilibrium or halt. Whenever you try to fix s/l to a particular proportion, and then try to apply this number to l/w, making l/w to comply to the number, you will need to adjust l or w to comply, and by that adjustment getting an new s/l, starting all over, continuing forever.

I would now rephrase my earlier statement, where I said that the golden ratio cannot be expressed by nominal values, but does exist and can be constructed: I would say now, that the golden ratio does not exist and cannot exist and the idea expressed as ϕ (or φ) points to nowhere.

The golden ratio equation is a logical equivalence of an oscillator. Maybe these are the ingredients for creating irrational numbers: over-specification and self-reference.

We are informed about a constraint of the partitioning, which sets the size of l (and by that, indirectly also for s) to a distinct value with respect to the whole w, by referring to the intrinsic structure of the partitioning. This is a **self-referential, recursive setup**, which adjusts itself by its own laws and definitions.

This very momentum of self reference, recursion and self-adjustment I want to pick up, when working out the algorithm in the following of this document.

The given golden ratio formula, or equation, explicitely creates a **new instance** of a partitioning, or creates a new whole w', based on a given, **original** whole w, and based on, and **inheriting** the particular structure or composition of w, which, the new whole w', is

$$w' = l + w \tag{3.2.4}$$

given the right side expression of the equation l/w.

The new whole w' inherits the same structure as the original whole w, i.e the original l of the original w takes the role of a new $s'(l = s')$ and the original w takes the role of the new $l'(w = l')$,

$$LITTLE\ ENDIAN\ :\ \frac{s'}{l'} \equiv \frac{l'}{w'} \equiv \varphi \tag{3.2.5}$$

which also implies, there is yet another, even bigger whole w'', composed by l' and w'.

This is a very creative moment of this unique setup, which indicates to us an endless creative process and growth where every new instance in the evolution inherits the structure from the previous, and expands towards the infinite. Whow!

3.3 Golden Ratio Definition - Big Endian

Starting at the big end, with the big endian notation, relating the larger l to the smaller s, the golden ratio is an equation as such:

$$BIG\ ENDIAN\ :\ \frac{l}{s} \equiv \frac{w}{l} \equiv \Phi \tag{3.3.1}$$

where s is the smaller, l is the larger, and w is the whole

$$w = s + l \tag{3.3.2}$$

and

$$0 < s < l < w \tag{3.3.3}$$

Having the larger on top creates different dynamics and magnitude, than we saw with the smaller related to the larger. We can see that at the two extremes, $s = l$ (symmetry, equality) and $s = 0$ or $s \to 0$ (no partition, or s becoming very small, shown in the overview table below.

	LE s/l	BE l/s
$s = l$	1	1
$s \to 0$	$\to 0$	$\to \infty$

3.4 Partitioning Examples - Overview

The following table shall provide an overview for some selective partitioning concepts with regard to structure and size characteristics.

The concept of a delta, or deviation from the golden ratio, shown in the table column on the right as absolute value (not considering signedness), shown under column title $\mid \delta_{LE} \mid$, shall be further elaborated in the following chapters.

	s/l	l/s	s/w	l/w	w	w'	$\mid \delta_{LE} \mid$
Fifth	1/4 (0.25)	4	1/5 (0.2)	4/5 (0.8)	1	1.8	0.6
Quarter	1/3 (0.33)	3	1/4 (0.25)	3/4 (0.75)	1	1.75	~ 0.41
Third	1/2 (0.5)	2	1/3 (0.33)	2/3 (0.66)	1	1.66	~ 0.16
Golden Ratio	φ (0.61)	Φ (1.61)	$1-\varphi$ (0.38)	φ	1	$1+\varphi$ (1.61)	~ 0
Symmetry	1	1	1/2 (0.5)	1/2	1	1.5	0.5

3.5 Partitioning Definition

Earlier we said, that both, s/l and l/w specify a structure for that particular partitioning setup. The resulting numbers do not distinguish between structure (proportion), size or quantity, all are expressed with (potentially same) numbers. Mathematics provide us the means to make our own definition of structure, which I shall attempt with a signature, using algebraic, formal specification of a data-type, as shown below. (For the formal definition of a data-type for partitioning, used for this particular partitioning concept for the golden ratio, I make use of a mathematical structure, comprising data and operations on that data.)

The signature for partitioning PART:

$$PART = (E, P, :, prop, scale) \tag{3.5.1}$$

where

E	is a set comprising Endianesses, with $E = \{BIG, LITTLE\}$
P	is a set for a PARTITION, with $P = \{s, l, w\}$, with smaller s, larger l, whole w
:	division
prop	using division to express proportion $prop(s,l) = \begin{cases} l/s & \text{for BIG endian} \\ s/l & \text{for LITTLE endian} \end{cases}$
scale	using division to express scaling $scale(l,w) = \begin{cases} w/l & \text{for BIG endian} \\ l/w & \text{for LITTLE endian} \end{cases}$

3.6 Error Function Definition

Within the scope of the given document and our golden ratio considerations, every point on the line AB, with $\mid AB \mid \; = \; w$, is characterized by its distance to the golden section S, with S dividing AB into a smaller s and a larger l.

If we transform the golden ratio equation

$$LITTLE\ ENDIAN\ :\ \frac{s}{l} \equiv \frac{l}{w} \equiv \Phi \tag{3.6.1}$$

to

$$LITTLE\ ENDIAN\ :\ \frac{s}{l} - \frac{l}{w} \approx 0 \tag{3.6.2}$$

and we know already, that 0 never can be reached, but a rest always remains, so we can say

$$LITTLE\ ENDIAN\ :\ \frac{s}{l} - \frac{l}{w} = \delta_{LE} \tag{3.6.3}$$

which gives us a measure of the deviation δ (greek letter delta) to the golden ratio, and provides us the formula for the error function, which is shown below as python implementation, for both, the little and the big endian approach.

```
1
2       def E_LE(self,s,l):
3         """
4         golden ratio error function (E) for little endian (LE) notation
5         relating the smaller to the larger
6         - s ... smaller
7         - l ... larger
8         - return delta aligned to granularity
9         """
10        if s>l: return self.E_LE(l,s)
11        delta = ((s/l)-(l/(s+l)))
12        return delta-delta%self._tau
```

```
1
2       def E_BE(self,s,l):
3         """
4         golden ratio error function (E) for big endian (BE) notation
5         relating the larger to the smaller
6         - s ... smaller
7         - l ... larger
8         - return delta aligned to granularity
9         """
10        if s>l: return self.E_BE(l,s)
11        if s == 0: return -100
12        if l == 0: return -100
13        delta = (((s+l)/l)-(l/s))
14        return delta-delta%self._tau
```

The following graph (figure 2) shows the deviation from the golden ratio for all possible partitions between 0 and 1, by applying the error function E_LE(s,l) for the little endian notation, as specified above.

For the symmetry case, where $s = l$, we get a deviation indication from the error function with the value $+0.5$, for the other extremes, smaller equals 0, on the far left and on the far right, i.e. no partition, we get a deviation indication with value -1.

Please notice, that the error function does not give the distance from the golden ratio section, where the graph crosses the x-axis, but rather yields the difference of two relations, as it was shown in formula (3.5.3) above, indicating the magnitude of the delta to the golden ratio proportion.

Looking at the beautiful graph (figure 2) - the mathematical equivalent for grace as in the wood print from Hokusai showing Mount Fuji (figure 1) - we also realize, that there are two partitions for the whole w divided into a smaller s and a larger l, which qualify for the golden ratio criteria: we either have the short end on the left or on the right, and vice versa for the long end. This tells us, that the golden ratio and all other asymmetric partitioning concepts, have as much symmetric as asymmetric features, if applied twice with reciprocal parameters. The graph actually shows both partitionings.

The x-axis in the graph shows the whole w with size 1, any point on the x-axis shows a valid partitioning into a smaller s and a larger l for that w, for which we then invoke the error function, which yields a measure for the difference to the golden ratio section, with the output shown on the y axis, given in the range of -1 and $+0.5$ for the little endian, for all points between 0 and 1. Two out of the many partitons, qualify for the golden ratio criteria, shown with $y = 0$ in the graph.

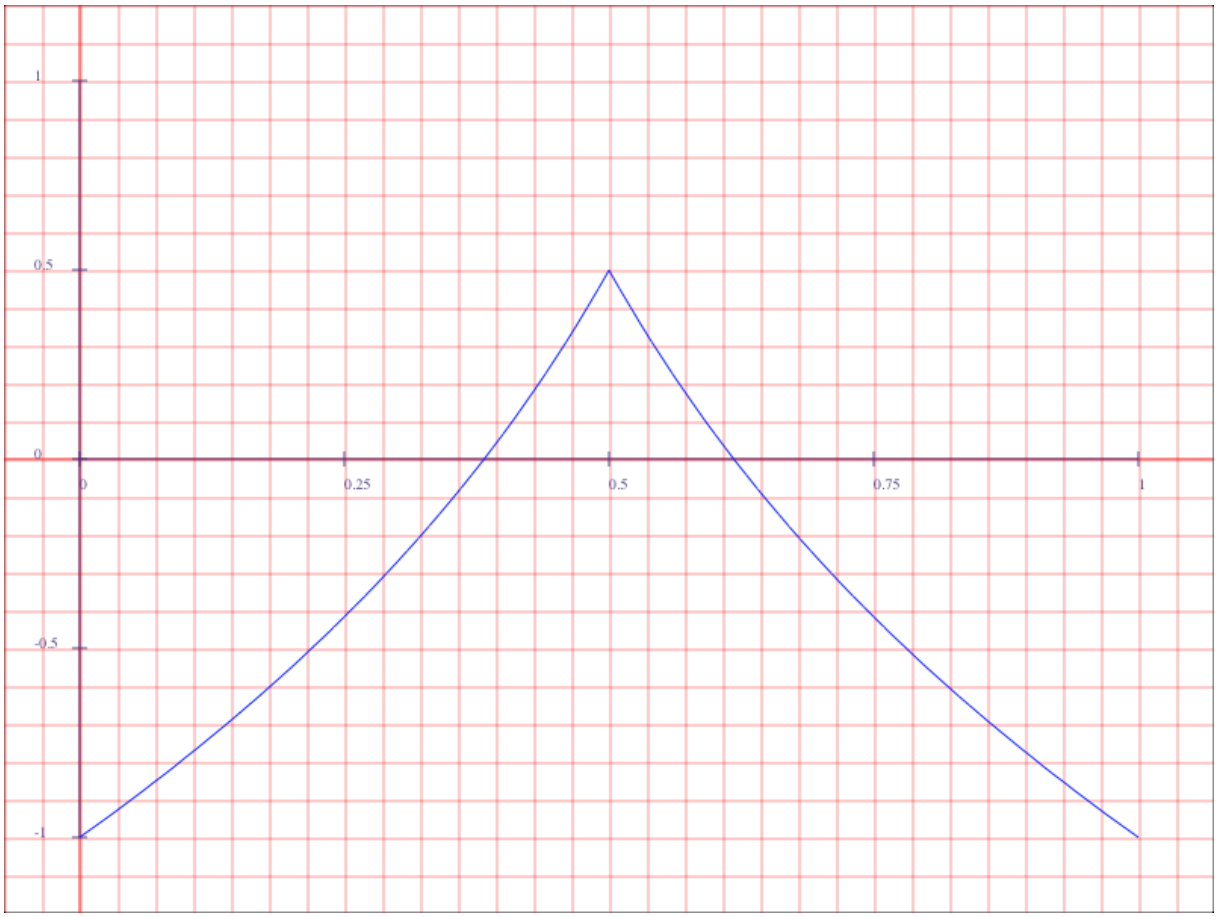

Figure 2: golden ratio error function values - little endian

The graph below (figure 3) shows the devation from the golden ratio for all possible partitions between 0 and 1, by applying the error function E_BE(s,l) for the big endian notation, previously specified. Please note the different scale for the y-axis.

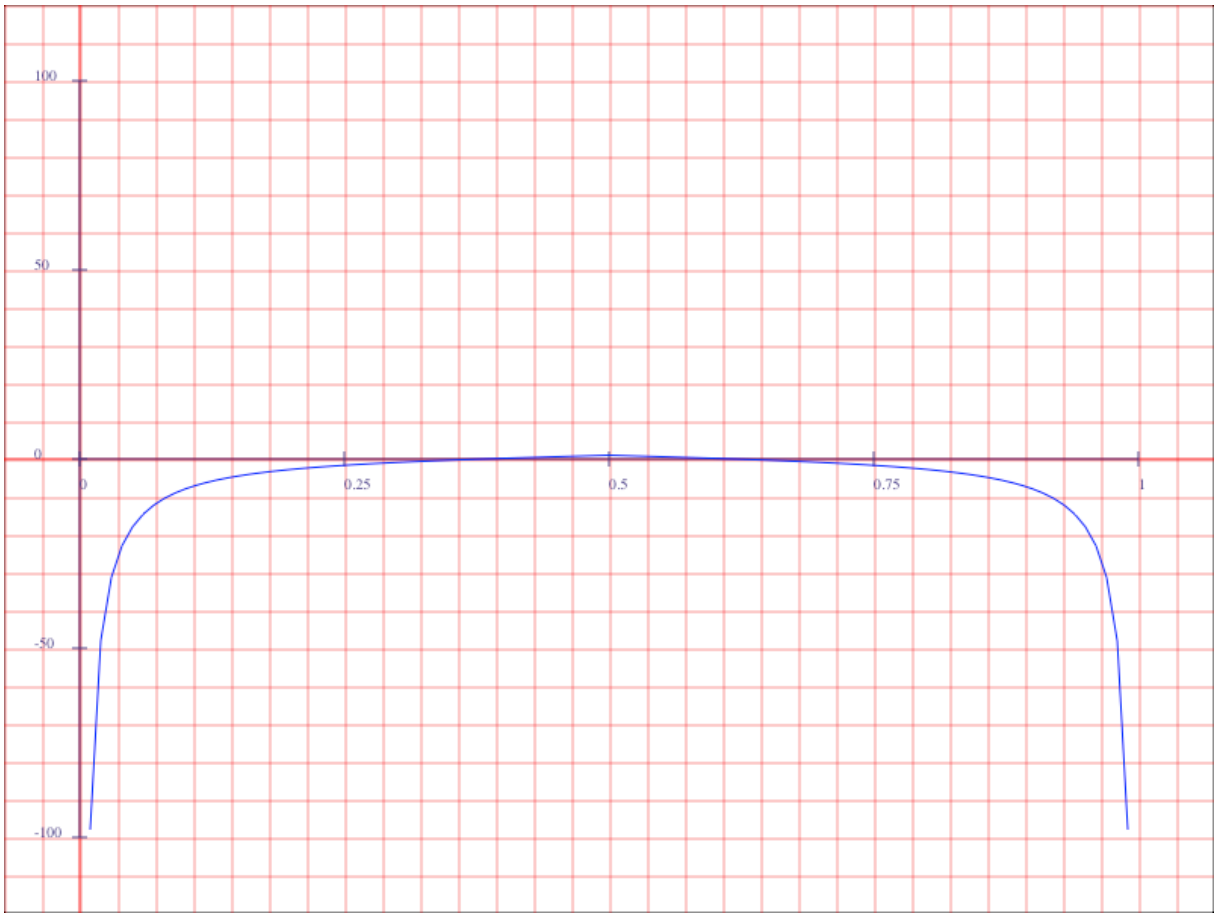

Figure 3: golden ratio error function values - big endian

4 Golden Ratio Approximation - Brute Force Algorithm

With the provided error function (using little endian notation, function E_LE()) to measure the misfit to the golden ratio proportion, we want to run through all possible partitionings between 0 and 1 with a given granularity τ, to find the points with the smallest deviation from the golden ratio (the null points in figure 2).

```
1
2       def brute_force_setup(self):
3         """
4         setup for brute force golden ratio calculation
5         """
6         self._tau = 1e-2
7         self._c = int(1/self._tau)
8         self._w = 1.0
9         self._null = np.array([[0,100.0],[0,100.0]])
10        self._min = np.array([[0,100.0],[0,100.0]])
11        self._max = np.array([0,-100.0])
12        self._null_id = 0
13        self._min_id = 0
14        self._golden_ratio = math_golden_ratio_t_golden_ratio.class_t_golden_ratio(tau=self._tau)
```

```
1
2       def brute_force_run(self):
3         """
4         calculate golden ratio with use of error function
5         by brute force run through all possible partitions between
6         0 and 1 for a given granularity tau
7         to find zero crossings (where GR applies).
8         """
9         for x in range(self._c+1):
10          s = 1.0*x/self._c
11          l = self._w-s
12          delta = self._golden_ratio.E_LE(s,l)
13          if delta < self._min[self._min_id][1]:
14            self._min[self._min_id] = (x,delta)
15          if abs(delta) < self._null[self._null_id][1]:
16            self._null[self._null_id] = (x,abs(delta))
17          if delta > self._max[1]:
18            self._max = (x,delta)
19          if x == self._c/2:
20            self._min_id += 1
21            self._null_id += 1
```

Above code produces the output shown below, providing the x values (left side from the colon) and their according error function result (right side of the colon) for the extremes min, null and max.

```
c: 100
tau: 0.01
null points: 38:0.01,62:0.01
min points: 0:-1.0,100:-1.0
max point: 50:0.49
```

With the given brute force method we already found the values for the golden ratio for the given tolerance tau=0.01 at null points $x = 0.38$ (38/100) and $x = 0.62$ (62/100).

5 Golden Ratio Approximation - Golden Ratio Algorithm

With the provided error function we can do better than the brute force method demonstrated above. To optimize the algorithm to aprroximate the golden ratio constant for a given whole, i.e. to find the pair of (s, l) which will come close to the golden ratio within the limits of a given tolerance τ,

For this we define a golden ratio function (for little endian) as shown in the following.

```
1
2       def GR_LE(self,s,l,cb=None):
3         """
4         golden ratio function (GR) for little endian (LE) notation
5         return modified s,l for a given pair of s,l
6         - w ... whole
7         - s ... smaller
8         - l ... larger
9         """
10        if s>l: return self.GR_LE(l,s)
11        delta = self.E_LE(s,l)
12        if (abs(delta)>self._tau):
13          if cb: cb(delta,s,l)
14          l = l+delta*s
15          s = s-delta*s
16          return self.GR_LE(s,l,cb=cb)
17        if cb: cb(delta,s,l)
18        return (s,l)
```

We use the golden ratio function as shown in the following.

```
1
2       def setup(self):
3         """
4         setup for recursive golden ratio calculation
5         """
6         self._tau = 1e-3
7         self._c = 0
8         self._i = 0
9         self._w = 1.0
10        self._golden_ratio = math_golden_ratio_t_golden_ratio.class_t_golden_ratio(tau=self._tau)
11        self._s = self._w/2-self._tau
12        self._l = self._w - self._s
13        (self._s,self._l) = self._golden_ratio.GR_LE(self._s,self._l,cb=self.count)
14        self._series_sle = np.zeros((self._c,3))


1
2       def run(self):
3         """
4         calculate golden ratio with use of error function,
5         recursive call with adoption of s,l based on deviation (error function result)
6         until a given precision, granularity tau is reached.
7         we start with 'random', initial s and l
8         """
9         self._s = self._w/2-self._tau
10        self._l = self._w - self._s
11        (self._s,self._l) = self._golden_ratio.GR_LE(self._s,self._l,cb=self.callback)
```

Above code produces the output shown below, showing the log of the golden ratio function iterations, recursions with smaller s, larger l and the delta for each iteration and the final result.

```
golden ratio calculation - recursive
    whole w : 1.0
  smaller s : 00.382
   larger l : 00.618
        tau : 00.001 ... precision
 iterations :
   [0] E_LE: 0.495 s: 0.4990 l: 0.5010
   [1] E_LE:-0.412 s: 0.2520 l: 0.7480
```

```
[2] E_LE:-0.092 s: 0.3558 l: 0.6442
[3] E_LE: 0.024 s: 0.3886 l: 0.6114
[4] E_LE:-0.010 s: 0.3792 l: 0.6208
[5] E_LE: 0.003 s: 0.3830 l: 0.6170
[6] E_LE:-0.001 s: 0.3819 l: 0.6181
```

The following graph (figure 4) shows the value smaller s starting at ca. 0.5 for 7 iterations for the approximation of the golden ratio constant. The recursive iterations stop once a given precision is reached.

Figure 4: golden ratio approximation - values for s and l over seven iterations

With the given golden ratio method we can approximate the golden ratio constant φ for any given whole w.

Index