An alternative strategy for harmonic numbers calculation and a numerical growth rate

Gerardo Miramontes-de León[1], **Diego Miramontes-de León**[2], and **Arturo Moreno-Báez**[1]

[1] *Autonomous University of Zacatecas (UAZ), Faculty of Electrical Engineering,*
Av. López Velarde 801, Col. Centro, Zacatecas , Zac., México, 98000.
gmiram@ieee.org, morenob20@uaz.edu.mx
[2] *Autonomous University of Zacatecas (UAZ), Faculty of Engineering,*
Av. López Velarde 801, Col. Centro, Zacatecas , Zac., México, 98000.
diego.miramontes@gmail.com,

Abstract

Some computational limitations when it is intended to calculate harmonic numbers for very large n values are analyzed. A reformulation of Euler's theorem is proposed, with which the range of its numerical calculation is extended. Two interesting results are reported, in the first one, an approximate growth rate $\Delta H = 2.3026$/decade is defined, which follows immediately from Euler's theorem. In the second, for $n = 10^p$, where p can be as large as $p = 10^{307}$, it is proposed H_n to be $H_n \approx M\,p + \gamma$, i.e., $p = \log(n)$ times a constant M (plus γ), which is also given, and $\log$ is the base 10 logarithm. The proposed approach was also compared with other well known specialized software libraries and computation environments to emphasize the important savings in computation time and numerical range.

Keywords— Harmonic numbers, Euler's approximation, divergence rate

I Introduction

In the 18th century, Leonhard Euler [8] proposed that the sum of the inverse of the first n natural numbers, given by

$$H_n = \sum_{k=1}^{n} \frac{1}{k} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \tag{1}$$

could be approximated as $\ln(n)$ plus a constant γ, that is,

$$H_n \approx \ln(n) + \gamma = H_{E_n} \tag{2}$$

where the approximation is denoted, in this work, as H_{E_n}, that is, Euler's approximation, and γ is known as the Euler-Mascheroni constant, calculated as $\gamma = 0.5772156649015328606065120900824024310421593359399\,2$ [1]. H_n given by (1) are called harmonic numbers.

On the other hand, for centuries it is known this problem has fascinated the mathematicians. It is also known the origin is related to the vibration of strings. It is so named because the wavelength of the harmonics of a vibrating string is inversely proportional to the length of that string according to the series of unit fractions: $1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, \ldots$

An ancient application is due to the famous philosopher Pythagoras who found the numerical proportion is responsible for musical harmonies. An interesting problem, discovered by [25] is to determine how far an overhang we can achieve by stacking dominoes over a table edge, accounting for the force of gravity, in which solution appear harmonic numbers. More recently, in financial markets, which all show harmonic and repetitive swings that are inherent in each particular market [20, 18], just to name a few applications. Going into the details of these applications is not the objective of this work.

To have an idea of the use of very big numbers, let us start with Carl Sagan who pointed out that the total number of elementary particles in the universe is around 10^{80} [4]. There are many other examples in Physics and Cosmology, and for a second example, let us refer to Max Tegmark who, in a multiverse or parallel universes theory, discusses a natural four-level hierarchy of multiverses and he proposes a universe containing about $10^{10^{115}}$ Hubble volumes at the quantum level [27].

To show that (2) is an approximation of (1), the error between both expressions for different values of n can be calculated. For example, for the first 10 values of n, the difference, in absolute value, is shown in Table 1, where each number has been calculated up to 16 decimal places.

It can be seen that when n increases, the error decreases. This comparison between H_n and H_{E_n} could, in theory, be continued for any value of n. However, this is practically not possible due to several factors. A first limitation when calculating H_n, that is, making a term by term summation, is the necessary computation time, which can be very long, when a very large n value is desired. A second limitation is the numerical representation in digital format. Due to the use of a finite number of bits, the calculation tool will deliver the fraction $1/n$, for very large n, equals to zero.

Continuing with the comparison between (1) and (2), and to confirm that the error continues to decrease as n increases, Table 2 shows the results for $n = 10, 10^2, \ldots, 10^8$. Note that H_n is calculated by adding term by term, and for the last value of n there are 100 million terms.

When n tends to infinity, then the sum given in (1) is called the harmonic series,

$$S = \sum_{n=1}^{\infty} \frac{1}{n} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots \tag{3}$$

so that H_n is simply the partial sum of S. As many textbooks show, there are well-known proofs of the divergence of the harmonic series [13], and a review of some divergent series can be found in [17].

However, the growth of the harmonic series is so slow so the first 10^{43} terms sum less than 100 [26, 2]. Based on this last observation, the question can be asked, how slowly does it diverge? In this article, this reported [15, 23] "asymptotic behavior" of (3) is reviewed and it is shown that there is a numerical growth rate, which gives a clear idea of the slow divergence of (1).

The paper is organized as follows: In Section II the statement of the problem and the main interest of this work is presented. A review of related literature follows in Sections III and IV. Section V shows an analysis about the behavior of harmonic numbers when the value of n is very large. One of the main results is given in Section V.1, i.e., a numerical growth rate (at the same time, divergence rate) is defined. In Section VI, a different and possibly overlooked approach for calculating harmonic numbers is presented. Finally, in Section VII, some concluding remarks are given.

II Problem Statement

The interest, from a mathematical point of view, of knowing the value of H_n for large n values can be found in [5, 14, 21, 30]. At present, these values have been limited, in some cases, by the available computational capacity.

A very simple way to approximate H_n is through (2). However, as it can be seen in the previous section, there is an error in the approximation of that value using term by term summation.

The following question is: up to what value of n is it computationally possible to calculate term by term summations? When trying to answer this question, two aspects show the practical limitations from a computational point of view. The first of these is the necessary computation time, if the calculation of H_n for a very large n value is desired, for example for $n \geq 10^{42}$ [10]. The second aspect refers to limitations in the numerical representation. For example, when trying to calculate (1) for $n >> 10^{42}$, fractions can trigger worst-case behavior of rational arithmetic. Although according to Euler, as n increases, the approximation between (1) and (2) is better, it is also true that the calculation of $\ln()$, instead of summation, may not be accurate due to rounding errors in the numerical representation to a finite number of bits.

In this paper, these two aspects are analyzed first, and then a reformulation of Euler's theorem, that is (2), is proposed, so that without increasing the error in the approximation, the harmonic number can be calculated for n values much higher than those previously reported. There is also a special interest in the remarkable behavior of the harmonic numbers when the number of terms are very very large. So, it is also reported a numerical growth rate value, which is not commonly seen in harmonic series and harmonic numbers literature.

III Review of some reported calculations for H_n

In order to obtain a value of H_n, two paths can be followed in general: *i*) make the summation term by term, *ii*) use some kind of approximation. With the option *i*) it is possible to calculate the sum thanks to the current calculation tools. For example, in [23] the result of the sum is shown for $n = 100$ using Mathematica software. In [31] the authors also used Mathematica to manipulate symbolic calculations and present a new sequence that converges to the Euler-Mascheroni constant. Moreover, a spreadsheet can be used, as shown in [24].

Figure 1 shows the harmonic numbers up to $n = 10^7$, where the term by term summation was made. The result of the sum is $H_n = 16.69531136585727$. The graph was generated with GNU Octave [7], a scientific calculation tool. A decreasing slope can be noted, which falsely suggests convergent behavior.

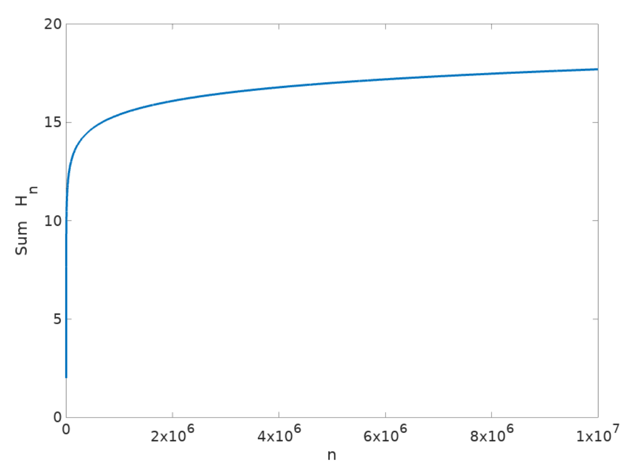

Figure 1: *Harmonic numbers up to* $n = 10^7$.

With *ii*) option, the computation time can be significantly reduced, since the term by term summation is avoided. An acceptable approximation is that given by (2). Another approach is the asymptotic standard expansion by means of the

Table 1: *H_n and H_{E_n} comparison, $n = 1, 2, \ldots, 10$.*

n	H_n	H_{E_n}	$\lvert H_n - H_{E_n}\rvert$
1	1.0000000000000000	0.5772156649015329	0.4227843350984671
2	1.5000000000000000	1.2703628454614782	0.2296371545385218
3	1.8333333333333333	1.6758279535696428	0.1575053797636905
4	2.0833333333333330	1.9635100260214235	0.1198233073119095
5	2.2833333333333332	2.1866535773356333	0.0966797559977000
6	2.4499999999999997	2.3689751341295877	0.0810248658704120
7	2.5928571428571425	2.5231258139568462	0.0697313289002963
8	2.7178571428571425	2.6566572065813685	0.0611999362757740
9	2.8289682539682537	2.7744402422377523	0.0545280117305014
10	2.9289682539682538	2.8798007578955787	0.0491674960726751

Table 2: *H_n and H_{E_n} comparison, for $n = 10, 10^2, \ldots, 10^8$.*

n	H_n	H_{E_n}	$\lvert H_n - H_{E_n}\rvert$
10	2.92896825396825	2.87980075789558	$4.91674960726751 \times 10^{-02}$
10^2	5.18737751763962	5.18238585088962	$4.99166674999607 \times 10^{-03}$
10^3	7.48547086055034	7.48497094388367	$4.99916666673705 \times 10^{-04}$
10^4	9.78760603604435	9.78755603687772	$4.99991666309541 \times 10^{-05}$
10^5	12.09014612986334	12.09014112987176	$4.99999157277387 \times 10^{-06}$
10^6	14.39272672286499	14.39272622286581	$4.99999181613475 \times 10^{-07}$
10^7	16.69531136585727	16.69531131585985	$4.99974177614604 \times 10^{-08}$
10^8	18.99789641385255	18.99789640885390	$4.99865393521759 \times 10^{-09}$

Euler-Maclaurin sum [3]:

$$\begin{aligned} H_n &\approx \ln(n) + \gamma + \frac{1}{(2n)} - \frac{1}{(12n^2)} + \frac{1}{(120n^4)} \\ &\quad - \frac{1}{(252n^6)} + \frac{1}{(240n^8)} - \frac{1}{(132n^{10})} \\ &\quad + \frac{691}{(32760n^{12})} - \frac{1}{(12n^{14})} + \ldots \\ &\approx \ln(n) + \gamma + \frac{1}{2n} - \sum_{k=1}^{\infty} \frac{B_{2k}}{n^k}\frac{1}{n^{2k}} = H_{EB_n} \end{aligned} \tag{4}$$

where B_{2k} are Bernoulli numbers. Equation (4) will be called H_{EB_n} or Euler-Bernoulli approximation.

Equation (4) is the most recommended way to calculate harmonic numbers. However, it is important to note that, for $n >> 1$, the approximation H_{EB_n} tends to be equal to the approximation H_{E_n}, since the terms to the right of $1/(2n)$, will become smaller and smaller. In fact, for $n = 10^{150}$, from the computational point of view, only the first two terms of Bernoulli remain, that is, $\frac{1}{(2n)} = 5.00\cdots \times 10^{-151}$, and $\frac{1}{(12n^2)} = 8.333\cdots \times 10^{-302}$. For $n = 10^{308}$, all terms, after γ are evaluated as zero.

IV Computational limitations when calculating H_n

Now, the problem of the time needed to perform the calculation of (1), when n is very large, is analyzed.

Using a naive approach, in a GNU Octave programming environment, the sum can be performed in a `for` loop as follows:

```
N=1e8;
S=0;
for n=1:N
  S=S+1/n;
endfor
```

where `1e8` means 1×10^8.

On a low-end personal computer (Intel Celeron(R) CPU N3050 at 1.60GHz x 2, in 64-bit mode), it was found that to obtain H_n up to $n = 10^9$ Octave requires 41.61 minutes. Using a system designed for fast computations in number theory, PARI/GP [19], a comparison is shown in Table 3. The difference in time, between both environments, is around four times for the last two numbers. Note how computation time increases according to the exponent, i.e, ten times when going from 10^8 to 10^9.

A better approach when using a calculation tool such as GNU Octave, which is a matrix calculation tool, is to build a vector of length N and with a single instruction it is possible to get the sum, that is, a vector such as `n=[1:1e8];` followed by `sum(1./n)` can be made. Then the value of $H_n \approx 18.9978964138526$ is obtained in a time of 3.2995 seconds.

Now another limitation appears in the calculation. If the vector `n=[1:1e9];` is going to be built, then Octave returns an error message, since the limit of the vector's length that can be handled is exceeded. Taking into account this limit, and taking advantage of the instruction set from Octave, there is a

Table 3: *Computation time and results for Octave and PARI/GP*

n	Time Octave	Time PARI/GP	H_n Octave	H_n PARI/GP
10^6	9.41 s	0.85 s	14.39272672286499	14.39272672865723
10^7	25.2378 s	7.50 s	16.69531136585727	16.69531136585985
10^8	4.089 min	1.12 min	18.99789641385255	18.99789641385389
10^9	41.61 min	11.50 min	21.30048150234850	21.30048150023479

function `cumsum`. In such a case, the speed of computation is further reduced, as shown in Table 4.

Table 4: *Computation time and results for Octave*

n	Time in s	H_n
10^5	0.00343012809753418	12.0901461298633
10^6	0.03386807441711426	14.3927267228650
10^7	0.60533595085144043	16.6953113658573
10^8	5.99508500099182129	18.9978964138526

For $n = 10^8$, 4.089 minutes have been reduced to 5.99 seconds.

In addition to the limitation on the length of the vector, when n begins to be very large, let's say $n > 10^{10}$, the computation time is still a problem for term by term summations.

On May 7, 2019, the Department of Energy of the United States of America announced a contract with the Cray Company, in collaboration with the processor manufacturer AMD, to deliver to the Oak Ridge National Laboratory a supercomputer which will be finished by 2021[11]. This supercomputer will have a performance greater than 1.5 exaflops, that is, 1.5×10^{18} floating point operations per second. With this in mind, and assuming that the machine can make 1.5×10^{18} summations in a second, a total of 3.17×10^{16} years will be needed to complete the sum up to a total of 1.5×10^{42} terms.

For example, Malone evaluated the sum using an AMD Athlon 64 CPU, clocked at 2.6 GHz in 64-bit mode. For $n = 2^{48}$, the calculation took a little more than 24 days [15].

One way to avoid this limitation is to use (4) to get H_n. Then, the computation time is reduced, but now a restriction appears in the representation of large numbers in digital format. According to GNU Octave, the maximum and minimum number that can be represented in double precision is $1.79769313486232 \times 10^{308}$, and $2.2507385850720 \times 10^{-308}$. If any number is exceeded above or below these values, it is obtained in Octave, `Inf` and `0` respectively.

Using (4) for $n = 10^{308}$ gives a value $H_n \approx 709.773424307068$. It is important to note that although a value for n has been used very close to the limit of the capacity of the machine, the harmonic number, or in theory the sum, is only slightly greater than 700. As a way of comparison, it is known that for $n = 10^{43}$ the sum is slightly less than 100 [2].

V Quasi asymptotic behavior of H_n

When the value of n is large enough, one can falsely observe a convergent behavior in the curve that represents the harmonic numbers. It should not be forgotten that (1) is divergent and this divergence is also reflected in the behavior of H_n when n tends to infinity. For example, Malone [15] investigated the convergence value of the harmonic series. Considering the finite precision of the computation tool, that author tried to find the value at which the sum converges or from which term the sum becomes constant. However, computational limitations are not sufficient reason to determine a convergence value. Malone found that the sum becomes constant with $n = 2^{48}$, that is, $n = 2.81474976710656 \times 10^{14}$, obtaining a value $H_{2^{48}} = 34.1220356680478715816207113675773143768310546875$.

Certainly, it is to be expected that when making term by term summation, the resolution of the machine will not be able to solve a value for $1/n$ if n is very large, giving from that value of n, a zero. For this particular case, i.e., $n = 2^{48}$, using (4), it was found $H_{2^{48}} = H_{2.81474976710656\times10^{14}} = 33.8482803317789$.

V.1 Growth rate for H_n

When investigating whether an asymptotic value can be determined for H_n with n very large, a value was determined to find how quickly the sum grows. An estimate of the speed of divergence is given in [28] as

$$H_{2^k} > \frac{k+1}{2}$$

and according to that author, *a complete response to speed of divergence of H_n in powers of $\frac{1}{n}$ is given by Euler's asymptotic standard expansion for H_n*, given in (4).

Although these approximations are well known, they do not really define a value of the speed of divergence. In this work, a different approach is taken. The first approach was to calculate the increase of H_n by taking $n = 10^p$ to 10^{p+1}, for $p < 8$. Subsequently, starting with $p = 10$, 11, 12, ... up to $p = 308$, and using (4), resulted in a constant growth rate $\Delta H = 2.3026$ / decade. A decade is the n interval given by $[10^{d-1}, \ldots, 10^d]$, with $d = 1, 2, 3, \ldots$. For example, the first decade, $d = 1$, goes from $[1, \ldots, 10]$, the second decade, $d = 2$, goes from $[10, \ldots, 100]$, and so on.

This result is remarkable, since it shows that H_{EB_n} only grows a small amount when n goes, for example, from 10 million to 100 million terms, from 100 million to 1000 million, and so on.

For example, taking $n = 10^{43}$ up to $n = 10^{44}$ it can only be expected an approximated (rounded) growth of 2.3026 in H_{EB_n}. The result was confirmed using $H_{10^{43}} = 99.5883746636455$ and $H_{10^{44}} = 101.8909597566395$ so that $\Delta H = 2.30258509299405$.

VI Proposed alternative calculation for H_n and results

Continuing with the analysis of the behavior of harmonic numbers, the following approach was chosen:

Let n be the maximum desired value in the approximation of H_n, and also be n expressed as

$$n = 10^p$$

since the main interest is for very large n values. For example, it will take $n = 15092688622113788323693563264538101449859497$ terms for H_n to exceed 100 [2]. That is $n = 1.50926886 \cdots \times 10^{43}$.

Clearly p is given by $p = \log(n)$, where log is the base 10 logarithm. Using the values of H_{EB_n}, given by (4), taking $n = 10^p$ for $p = 10$ up to $p = 308$, the question arises how does H_{EB_n} grow with the exponent p instead of the number of terms n?

Since the main interest is the calculation of H_n for $n >> 1$, and since $H_{EB_n} \approx H_{En}$, the following calculation is proposed.

Let $n = 10^p$, so $p = \log(n)$:

$$M \log(n) + \gamma = \ln(n) + \gamma,$$
$$M\, p + \gamma = \ln(10^p) + \gamma$$

where it is easy to verify that

$$M = \frac{1}{\log(e)} = 2.30258509299405. \quad (5)$$

It should be noted that M is independent of p and n.

Then, the approximation of H_n is proposed for $n >> 1$ as

$$H_n \approx M\, p + \gamma = H_{Mp} \quad (6)$$

Equation (6) will be called approximation H_{Mp}.

This is an important simplification in the calculation of H_n, when n is very large, since only a single product $M\, p$ and a sum (the term γ) are now required, provided that n is expressed as $n = 10^p$. This avoids the calculation of the logarithm for a large number n and only the p exponent is required to calculate $H_{n=10^p}$.

Also

$$M = \frac{\ln(n)}{\log(n)}, \text{and as } p = \log(n) \text{ with } n > 1$$

then

$$M\, p = \frac{\ln(n)}{\log(n)} \log(n) = \ln(n)$$

thus

$$H_{E_n} = \ln(n) + \gamma = M\, p + \gamma = H_{Mp} \quad (7)$$

Now it is shown that the absolute error between H_{En} and H_{Mp} is very small.

Let us define an acceptable absolute margin of difference (ϵ) to consider two floating-point numbers as equal. According to [16] that margin of difference is many times greater than the machine's ϵ. This is because a sum involving thousands of terms, and other calculations can have a significant number of rounding errors. An exhaustive explanation of rounding errors and a guide to choose an acceptable ϵ, can be found in [9].

In GNU Octave $\epsilon = 2.22044 \times 10^{-16}$. So, defining $\epsilon = 10^{-14}$, then

$$\text{Error} = |H_{En} - H_{Mp}| < \epsilon$$

Although theoretically the error must be zero, if the following calculation is performed in GNU Octave

$$\text{Error} = |\ln(n) - M\, p| = |\ln(n) - \frac{\ln(n)}{\log(n)} \log(n)| \quad (8)$$

for $n = 10^p$ and $p = 2, 3, \ldots, 10$, it is found that the error is different from zero, but at the same time it is observed that the condition Error $< \epsilon$ is met. It should be noted that the error is different from zero due to limitations in the numerical representation and rounding errors. The results in GNU Octave are shown in Table 5. It is important to see that with this value

Table 5: *Error calculation due to machine number representation*

$n = 10^p$	$\lvert ln(n) - \frac{ln(n)}{p} p\rvert < \epsilon$
10^2	8.88178419700125e-16
10^3	1.77635683940025e-15
10^4	1.77635683940025e-15
10^5	1.77635683940025e-15
10^6	3.55271367880050e-15
10^7	3.55271367880050e-15
10^8	3.55271367880050e-15
10^9	7.10542735760100e-15
10^{10}	7.10542735760100e-15

of M, for $n \geq 10^p$, H_n can be calculated without the need of (1), or (2), nor (4). Moreover, using p instead of $\log(n)$, it can be obtained an approximation of H_n for very large n values. Since $n = 10^p$ and p can be equal to 10^{308}, then the approximation of H_n would be calculated with $n = 10^{10^{308}}$.

Now that the advantage of using the constant M has been shown, it should be noted that another way of seeing that the growth rate per decade of harmonic numbers is precisely equal to M, is obtained by calculating the difference between $H_{M(p+1)}$ and H_{Mp}

$$\begin{aligned} \Delta H &= H_{M(p+1)} - H_{Mp} \\ &= (M(p+1) + \gamma) - (Mp + \gamma) \\ &= M = 2.30258509299405 \end{aligned} \quad (9)$$

which had already been found in a heuristic manner in Section V.1.

To verify the calculations of H_n with the constant M, that is using (6), it was compared with the result of applying (2). In Table 6, H_{Mp} corresponds to (6). It can be seen, the error is close to ϵ, and in some cases the machine returns it as zero.

Although, to our knowledge, there are no values similar to those reported here, in Table 7 some harmonic numbers are shown for n up to 10^{5000}. It should be noted that for these values of n, (2) or (4) cannot be applied anymore, since the machine limit is $1.79769313486232 \times 10^{308}$.

About the behavior of harmonic numbers, it is interesting to observe the value of H_n is relatively small, that is, for $n =$

Table 6: *Comparison between H_{Mp} and H_{E_n}*

$n = 10^p$	$H_n \approx H_{Mp}$	$H_n \approx H_{En}$	Error $= \lvert H_{Mp} - H_{En} \rvert$
10^{10}	23.6030665948420	23.6030665948420	3.55271367880050e-15
10^{43}	99.5883746636455	99.5883746636455	1.42108547152020e-14
10^{50}	115.7064703146038	115.7064703146038	0.00000000000000e+00
10^{100}	230.8357249643061	230.8357249643061	0.00000000000000e+00
10^{200}	461.0942342637107	461.0942342637107	0.00000000000000e+00
10^{300}	691.3527435631154	691.3527435631153	1.13686837721616e-13
10^{305}	702.8656690280856	702.8656690280856	0.00000000000000e+00
10^{308}	709.7734243070677	709.7734243070677	0.00000000000000e+00

Table 7: *Calculation of H_{Mp} for $n > 10^{308}$*

$n = 10^p$	$H_n \approx$M×p
10^{500}	1151.86976216192
10^{1000}	2303.16230865895
10^{2000}	4605.74740165299
10^{5000}	11513.50268063513

10^{5000}, $H_n \approx 11522.29$, which continues to reflect how slowly H_n diverges.

Thus, in this work, harmonic numbers for values of n much greater than 10^{43} are reported. In addition, if the well-known approximation given in (2) or (4) is used, again using Octave, $n = 10^{308}$ would be the highest possible value of n to make the calculation of H_n, since the limit capacity of the machine cannot be exceeded.

VI.1 Comparison results between Mp and specialized software

Specialized software libraries such as class libraries in C++, or Python, becomes common place for mathematical algorithms, so it is not unreasonable to compare the results between different approaches. One such a specilized software is mpmath, a free (BSD licensed) Python library for real and complex floating-point arithmetic with arbitrary precision [12].

In the mpmath library, a function `harmonic(n)` can be found. If n is an integer, `harmonic(n)` gives a floating-point approximation of the n-th harmonic number H_n.

According to the mpmath documentation, "the function `mpmath.harmonic` is evaluated using the digamma function rather than by summing the harmonic series term by term. It can therefore be computed quickly for arbitrarily large n, and even for nonintegral arguments."

For sake of comparison, let us obtain H_n for $n = 10^{100}$ in mpmath, the result is given as

```
>>> harmonic(10**100)
230.835724964306
```

This result from mpmath can be compared with the value shown in Table 6, for $n = 10^{100}$, i.e., forth line. It can be seen, both results are practically the same, but instead of digamma function, a single product plus γ was used. This difference in the computation of harmonic numbers is one of the contributions we are reporting.

If the reader is interested, other available platform can be revised, like [22]. `dCode` is also a tool for calculating the values of the harmonic numbers [6], among many others. The reader can try to obtain the value of harmonic numbers for $n \geq 10^{1000}$ on those platforms.

Another comparison can be made with a widely used computer algebra system designed for fast computations in number theory that is called PARI/GP [19]. For this case, the interesting issue is the size of the number it can be introduced to perfom the computation of the harmonic number. It was found it is not possible to introduce numbers such as $10^{10^{10}}$. For example, with the default configuration, the largest accepted number, without issuing an error, was 10^{10^6}. Trying 10^{10^7}, PARI/GP delivers the message `the PARI stack overflows`.

A last comparison was done using a very interesting platform, WolframAlpha [29]. In this case, the online version gives the opportunity to try really big numbers. Without any problem it is possible to obtain H_n for $n = 10^{5000}$, and $n = 10^{10^{10}}$. To better compare the results between the proposed approach and the WolframAlpha platform, Table 8 shows some comparing results between the proposed approach and the funtion `HarmonicNumber[n]` of the WolframAlpha platform.

The out from WolframAlpha in the last case is shown as:

```
Try the following:
    Use different phrasing or notations
    Enter whole words instead of abbreviations
    Avoid mixing mathematical and other notations
    Check your spelling
    Give your input in English
```

It can be seen for $n \geq 10^{10^{16}}$ WolframAlpha is not capable to deliver a result. With the proposed approach we could use $n = 10^{10^{307}}$, since in this case $p = 10^{307}$, and as it was shown along the paper we can write $H_n \approx M\,p + \gamma$, where M is already known.

VII Conclusions and future work

In this paper, an alternative strategy was proposed to find harmonic numbers for very large n values, that is, the computational limit of $n = 10^{308}$ was exceeded to calculate H_n with n close to $10^{10^{308}}$.

Table 8: *Comparison of H_{Mp} and WolframAlpha for $n > 10^{308}$.*

$n = 10^p$	$H_n \approx H_{Mp}$	`HarmonicNumber[n]`
10^{500}	1151.86976216192	$1151.86976216192\ldots$
10^{1000}	2303.16230865895	$2303.16230865895\ldots$
10^{2000}	4605.74740165299	$4605.74740165299\ldots$
10^{5000}	11513.50268063513	$11513.50268063513\ldots$
$10^{10^{15}}$	$2.30258509299405 \times 10^{15}$	$2.302585092994046 \cdots \times 10^{15}$
$10^{10^{16}}$	$2.30258509299405 \times 10^{16}$	NO RESULT

The results presented are the following:

1. The calculation limits of H_n were exceeded, according to the literature review, for the n values consulted.
2. A growth rate of harmonic numbers H_n was established, approximately, at 2.3026/decade.
3. A constant M was defined and a new expression that allows to calculate H_n drastically reducing the computational load. This new expression can be compared to Euler's formula where H_n tends to be exactly $\ln(n)$ plus a constant γ, that is, $H_n \approx \ln(n) + \gamma$.

It was shown that for $n \geq 10^p$, $p \geq 1$, H_n tends to be $\log(n)$ times a constant M, which is, after rounding, $M = 2.30258509299405$, that is $H_n \approx M\,p + \gamma$.

Future work includes an evaluation of the effect of the number of bits in the error found in Table 5. However, it should be clear that the processing of floating point numbers is independent of the GNU or proprietary environment. Even when environments such as Octave or MATLAB give the impression that it works with fractional figures, internally the calculation must be processed according to the IEEE754 standard, in this sense, there is no point in comparing processors. Therefore, this work is extensible to any architecture that follows this standard.

The change from a cycle-based algorithm to one based on a simple product of two factors, that is, $M\,p$, and a sum, with an acceptable error, sets the path for future reformulation of other similar problems, which are based on infinite sums.

References

[1] W. A. Beyer and M. S. Waterman. "Error Analysis of a computation of Euler's constant". In: *Math of Comp.* 28 (1974), pp. 599–604.

[2] R. P. Boas Jr. "Partial Sums of the Harmonic Series". In: *The Amer. Math. Monthly* 78.8 (1971), pp. 864–870.

[3] T. J. l'A. Bromwich. *An Introduction to the Theory of Infinite Series*. St Martin's Street, London: 324-325: Macmillan and Co. Limited, 1926.

[4] Sagan C. *Cosmos*. Book Club Associates. 1981, pp. 220–221. ISBN: 9780354045315.

[5] C-P Chen. "Ramanujans Formula for the Harmonic Number". In: *Appl. Math. Comput.* 317 (2018), pp. 121–128.

[6] *dCode Tool*. https://www.dcode.fr/nombre-harmonique.

[7] J. W. Eaton et al. *Octave version 4.2.1 manual: a high-level interactive language for numerical computations*. `https://www.gnu.org/software/octave/doc/v4.2.1/`.

[8] L. Euler. *De progressionibus harmonicis observationes*. 1740.

[9] D. Goldberg. "What Every Computer Scientist Should Know About Floating-Point Arithmetic". In: *ACM Computing Surveys* 23.1 (Mar. 1991).

[10] J. Havil. *Gamma, Exploring Euler's Constant*. Princeton University Press, 2003, p. 23.

[11] https://www.amd.com/frontier. *Frontier supercomputer*. `https://www.amd.com/frontier`.

[12] F. Johansson et al. *mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 1.1.0)*. `http://mpmath.org/`. Dec. 2018.

[13] R. Larson and B. H. Edwards. 10th. Independence, N. Y.: Cengage Learning, 2013.

[14] B. Lubeck and V. Ponomarenko. "Subsums of the Harmonic Series". In: *The Amer. Math. Monthly* 125.4 (2018), pp. 351–355.

[15] D. Malone. "To what does the harmonic series converge?" In: *Irish Math. Soc. Bulletin* 71 (Nov. 2013), pp. 59–66. ISSN: ISSN 0791-5578.

[16] *MicroSoft*. `https://docs.microsoft.com/en-us/dotnet/api/system.single.epsilon?redirectedfrom=MSDN&view=netframework-4.8`.

[17] D Miramontes-de León and G. Miramontes-de León. "Los infinitos de algunas series divergentes". In: *Revista Digital Matemática, Educación e Internet* 20.2 (Apr. 2020). ISSN: 1659 -0643.

[18] C. Mitchell. *Harmonic Patterns in the Currency Markets*. Jan. 2020.

[19] *PARI/GP version* `2.11.2`. available from `http://pari.math.u-bordeaux.fr/`. The PARI Group. Univ. Bordeaux, 2019.

[20] L. Pesavento and L. Jouflas. *Harmonic Numbers and How to Use Them*. Larry Pesavento and Leslie Jouflas. 2012.

[21] A. Plaza. "The Harmonic Series Diverges". In: *The Amer. Math. Monthly* 125.3 (2018), p. 222.

[22] *Program to find N-th Harmonic Number*. https://www.geeksforgeeks.org/program-to-find-sum-of-harmonic-series/.

[23] A. Rivera. "Divergencia de la serie armónica". In: *Educación Matemática* 11.3 (1999), pp. 89–94.

[24] J. A. Rochowicz Jr. "Harmonic Numbers: Insights, Approximations and Applications". In: *Spreadsheets in Education (eJSiE)* 8.2 (2015).

[25] R. T. Sharp. "Problem 52: Overhanging dominoes." In: *Pi Mu Epsilon Journal* 1.10 (1954), pp. 411–412.

[26] N. J. A. Sloane. *Sequence A082912 (Sum of a(n) terms of harmonic series is $> 10^n$)*. https://oeis.org/A082912.

[27] Max Tegmark. "Parallel Universes". In: *Scientific American* 288.5 (Mar. 2003). DOI: 10.1038/scientificamerican0503-40.

[28] M. B. Villarino. *Ramanujan's Harmonic Number Expansion into Negative Powers of a Triangular Number*. https://arxiv.org/abs/0707.3950v2. 2007.

[29] *WolframAlpha computational intelligence*. https://www.wolframalpha.com/input/.

[30] A. Xu. "Ramanujan's Harmonic Number Expansion and Two Identities for Bernoulli Numbers". In: *Results Math* 72 (2017), pp. 1857–1864.

[31] X. You and DR. Chen. "A new sequence convergent to Euler-Mascheroni constant". In: *J. Inequal Appl.* 1.75 (2018).