PH3105 Problem Set 2

A major problem in numerical computation is that of finding the limit of a sequence, provided it exists. Since the "sum" of an infinite series $S = \sum_{n=0}^{\infty} t_n$ is really the limit of the sequence formed by the partial sums $S_N = \sum_{n=0}^{N} t_n$, $N = 0, 1, 2, \ldots$ summing up a series is essentially the same problem. One can not really generate all terms in an infinite sequence (or sum up all terms in an infinite series) in a numerical computation, and so one may resort to calculating a very large number of terms of the sequence and hope that (after the initial few terms) the final terms will be pretty close to the limit. However, this idea often works badly in practice.
As an example, let us consider the sum

$$S = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \ldots = \sum_{n=0}^{\infty} \frac{(-1)^n}{n+1} \tag{1}$$

This is an alternating series (the terms come with alternating + and - signs), where the terms decrease monotonically in magnitude. It is a standard result in mathematics that such a series always converges[a]. Indeed the "sum" of this series is easily seen[b] to be $\ln 2 = 0.6931471805599453$. If you look at the first few partial sums you get

$S_0 = 1.0$, $S_1 = 0.5$, $S_2 = 0.833333333333$, $S_3 = 0.583333333333$, $S_4 = 0.783333333333$

$S_5 = 0.616666666667$, $S_6 = 0.759523809524$, $S_7 = 0.634523809524$, $S_8 = 0.745634920635$, $S_9 = 0.645634920635$

etc. This shows that although the series may converge, it does so rather slowly. Indeed it is easy to see that in such monotone alternating series, the truncation error (the gap between the truncated sum S_N and the actual sum S) is bounded by the magnitude of the first term that you drop[c]. This shows that to sum up the series to three decimal places, you must sum up at least the first thousand terms. To get six figure accuracy requires a whopping million terms! Obviously this method needs to be improved if it is to become a practical one.

[a] It is easy to see by rewriting the sum as $\left(1 - \frac{1}{2}\right) + \left(\frac{1}{3} - \frac{1}{4}\right) + \ldots$ and $1 - \left(\frac{1}{2} - \frac{1}{3}\right) - \left(\frac{1}{4} - \frac{1}{5}\right) - \ldots$ that the sum has to lie between 0 and 1

[b] Recall that $\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots$

[c] The proof is very similar to that of the convergence in the first footnote.

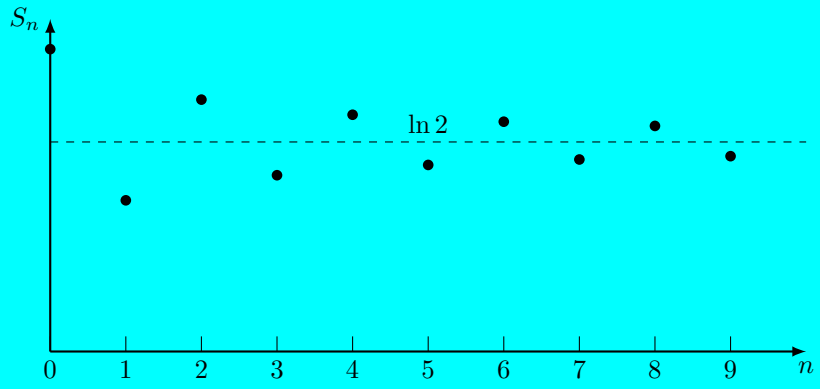
Q 1) The python command

```
ts = [(-1)**i/(i+1) for i in range(20)]
```

generates the first 20 terms in the infinite series for $\ln 2$. Write a program that will print out the first twenty partial sums S_0, $S_1, \ldots S_{19}$ of this series.

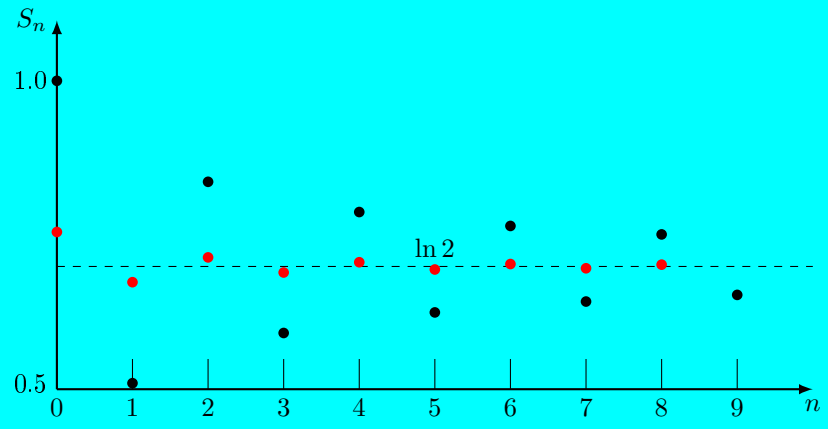
Q 2) Use the python function **sum()** which takes a list of numbers as its input and returns the sum of all the entries to write a single line of python code that you can write in the python interpreter that will sum up the series to a million terms. Check that your answer does have the desired accuracy.

While a direct sum of the series seems to be impractical, the partial sums that I have noted down before (as well as the ones that you will calculate using your program in Q1 above reveal a rather striking property of the partial sum. This is that they are alternately larger and smaller than the final limit ($S_0 = 1$ is larger than $\ln 2$, $S_1 = 0.5$ is smaller, S_2 is larger, and so on ....).



This suggests a possible method to get to the answer faster. If we replace S_0 by the average of S_0 and S_1, S_1 by the average of S_1 and S_2, and so on down the line, we will get a new sequence that converges to the same sum, only does so faster.

It is obvious from the graph below (note that the scale on the vertical axis has been expanded) that this sequence is a marked improvement over the previous one as far as convergence is concerned. In fact the 6th term in this sequence is already accurate to two decimal places (something that would have taken the original sequence a hundred terms!



So, we have found a transformed sequence - one that converges to the same limit as the earlier one - but does so faster. The process of changing the original sequence S_N to the new improved one S'_N is our first example of a **convergence accelerating transform (CAT)**.

In this case, the first six terms of the transformed sequence S'_N gives us two figure accuracy. Can we do better? It turns out that we can! The new sequence also shares the property of taking larger and smaller values than the limit alternately - which means that we can repeat the transformation - this time giving the first 8 terms of another sequence that converges to the limit even faster. All we have to do to get a much better result is keep on repeating this transform

$$S_n^{(i+1)} \equiv \frac{1}{2}\left(S_n^{(i)} + S_{n+1}^{(i)}\right) \tag{2}$$

named, after its discoverer, the Euler transform. In this equation, the superfix (i) or $(i+1)$ is the number of successive generations, the suffixes as usual standing for the sequence index. The original sequence is denoted here by $S_n^{(0)}$. Of course, if you start with a finite number, say 10, terms in the first sequence, you can only repeat this process 9 times. But at the end, the one term that you will be left with is quite an accurate estimate for $\ln 2$! In fact, it is accurate by about 1 part in 10^5 - something that would have taken 10^5 terms if the original sequence were used!

Q 3) The program **EulerCAT.py** carries out the procedure described above. It asks the user for the number of terms to use in the original series, creates a list of partial sums, and then repeatedly applies the Euler CAT as long as the sequence has more than one term left. The final term that is left at the end of the process is our estimate for the sum. The program can be shortened considerably by using the numpy array object. Rewrite the program using this.

Q 4) The Euler CAT can be used for any alternating decreasing series. Use your program to sum the two series

$$1 - \frac{1}{2} + \frac{1}{4} - \frac{1}{8} + \frac{1}{16} - ...$$

and

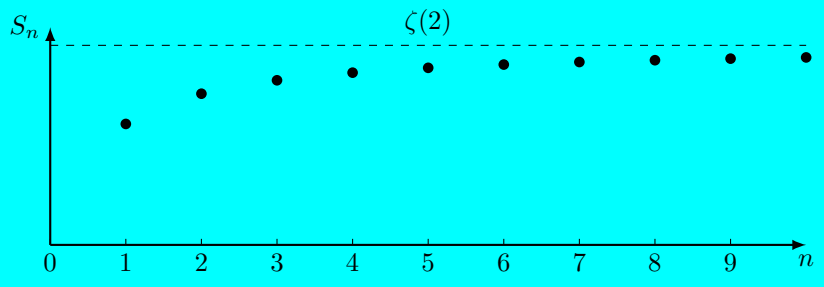
$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots$$

In each case start with ten terms of the original series and comment on how accurate your final answer is. (The first series obviously sums to $\frac{2}{3}$, while the second one's sum is $\frac{\pi}{4}$).

So far, we seem to have a wonderfully good way of summing up series. However, note that our method would completely break down for series like

$$\zeta(2) \equiv 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \ldots = \sum_{n=1}^{\infty} \frac{1}{n^2} \tag{3}$$

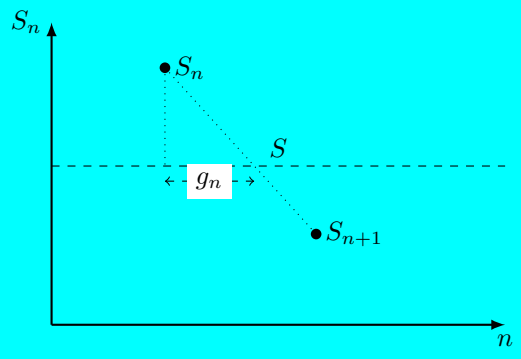
This is also a convergent series, but here the partial sums approach the limit monotonically from below. This means that averaging two successive terms takes you further away from the limit than the last term that you use, making the transformed series converge even more slowly!



In order to understand what needs to be done to accelerate this kind of series, let us first return to the example of the series for $\ln 2$. Let us look closely at any two successive terms S_n and S_{n+1}. The actual sum of the series, S, lies somewhere in between. We can write S as

$$S = S_n + g_n \Delta S_n$$

where $\Delta S_n \equiv S_{n+1} - S_n$ and g_n is a number between 0 and 1, whose interpretation you can see in the figure below.



Note that since this merely defines g_n through

$$g_n = \frac{S - S_n}{\Delta S_n} \tag{4}$$

this can always be done! In this language we can see that what we had really done in the Euler method is pretend that the sum S is exactly half way on the line joining the points (n, S_n) and $(n+1, S_{n+1})$, i.e., the assumption $g_n = \frac{1}{2}$. Of course, since g_n isn't really $\frac{1}{2}$, assuming this does not really sum the series, but the estimate $S_n + \frac{1}{2}\Delta S_n$ gives us the n-th term of a new series that coinverges better.

In order to go beyond the Euler CAT, we must give up the model $g_n = \frac{1}{2}$. The next simplest model for g_n that we can think of is that g_n is a constant, but not necessarily $\frac{1}{2}$. Using $\Delta g_n = 0$, we can reqrite (4) to the form

$$\Delta\left(\frac{S}{S_n}\right) = \Delta\left(\frac{S_n}{\Delta S_n}\right), \qquad \text{or} \qquad S = \frac{\Delta\left(S_n/\Delta S_n\right)}{\Delta\left(1/\Delta S_n\right)}$$

After a bit of algebra you can rewrite this expression (work this out using the result $\Delta\left(\frac{a_n}{b_n}\right) = \frac{(\Delta a_n)b_n - a_n(\Delta b_n)}{b_n b_{n+1}}$ which is rather easy to prove itself) as

$$S = S_n - \frac{(\Delta S_n)^2}{\Delta^2 S_n}$$

Of course, once again, the model g_n = constant is not exact, so that the value of S we get above is not the exact sum. However, we may reasonably expect this to lead to the $n-$th term of a a new (hopefully improved) sequence! This gives birth to the famous Δ^2 transform of Aitken

$$S_n^{(i+1)} = S_n^{(i)} - \frac{\left(\Delta S_n^{(i)}\right)^2}{\Delta^2 S_n^{(i)}} \tag{5}$$

Note that while the restriction that g_n must lie in between 0 and 1 is perfectly reasonable for alternating series, it is not correct for monotone series. However the assumption that g_n is a constant, while not exact, can be approximately valid even for the latter. Thus the Δ^2 transform can be used to accelerate such monotone convergent sequences. Note also that each time this transform is carried out, the number of terms in the next sequence decreases by 2. We can repeat this transform over and over again, as long as the sequence has more than two terms left. When we stop, the last term of the last sequence that we obtain should be a very good estimate for the limit.

Q 5) Write a program that will use the Aitken Δ^2 transform to sum up $\zeta(2)$. The exact value of $\zeta(2)$ is $\frac{\pi^2}{6}$. Determine how many significant digits can be obtained correctly by starting from the first ten partial sums.

Q 6) As you may guess, the transformation based on $\Delta^2 g_n = 0$ (we no longer model $g_n$as a constant, but assume that Δg_n is nearly so) is also quite powerful. This reads, obviously

$$S_n^{(i+1)} = \frac{\Delta^2\left(S_n^{(i)}/\Delta S_n^{(i)}\right)}{\Delta^2\left(1/\Delta S_n\right)}$$

Write a program to implement this algorithm and use it to estimate how accurate an answer you get for all the example series in this problem set.