Compeng 3SK3

Project 1

Aaron Pinto
pintoa9
400190637

The pseudocode and simulation program are included in the MATLAB file.

4)

a) No, you cannot because the IEEE-754 32-bit floating point number has an inherent max precision of 6-9 significant decimal digits. Therefore, at some point, 1/N, will become too small to be representable by a 32-bit floating point number.

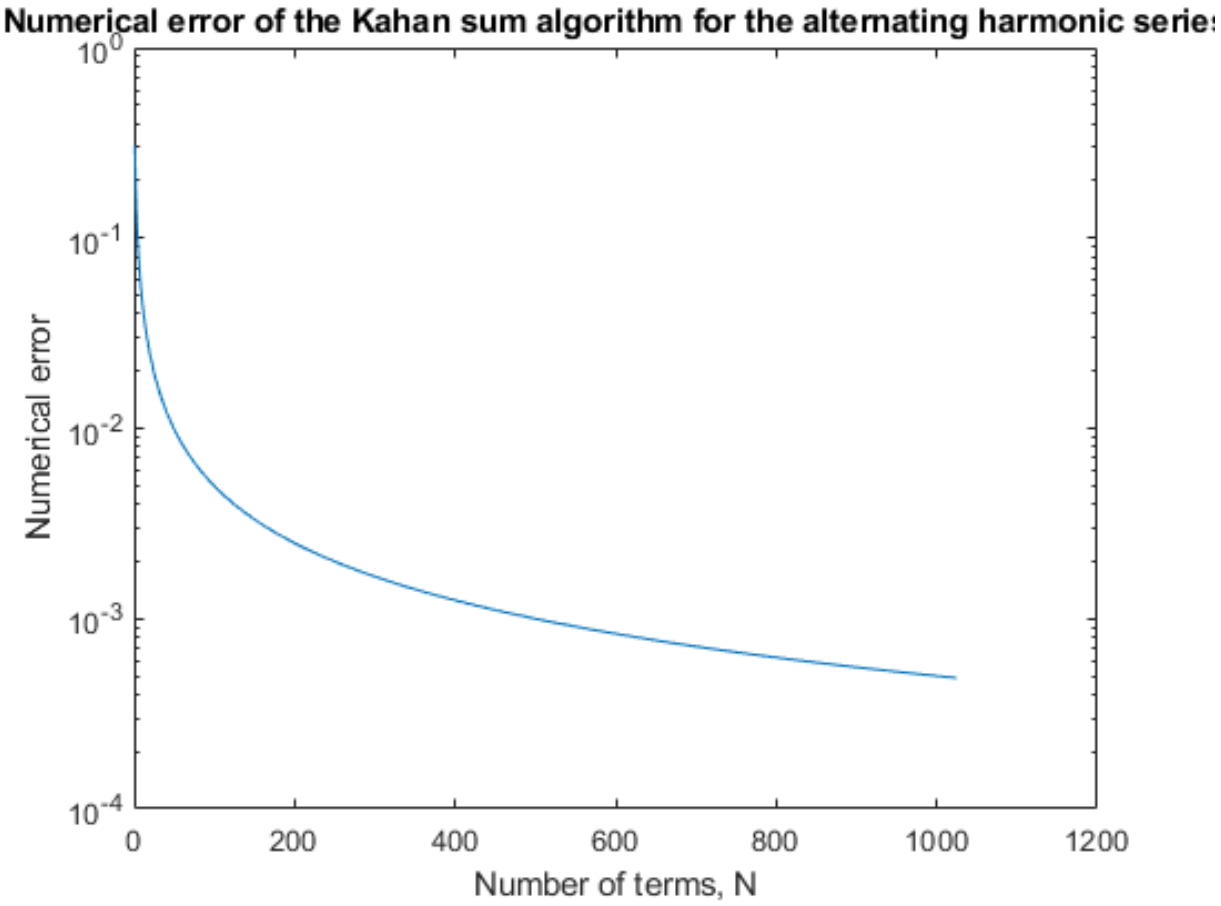
b) Using an IEEE 32-bit floating point number, the minimum N value such that 1/N becomes too small to be representable is N = 2^128, as any integer greater than or equal to 2^128 is rounded to infinity. The minimum N value such that 1/(N - 1) - 1/N becomes too small to be representable is N = 2^24 + 4. When I plugged this value - 1 into the equation represented in MATLAB, I got 0.000000000000000710542735760100185871124267578l2, so 2^24 + 3 is still representable. This is the code I used:

```
N = 2^24 + 3;
test = vpa(single(single(1.0) / single(N - 1) - single(1.0) / single(N)))
```

The vpa() does not change the actual value of the answer, without vpa() I get 7.1054e-15.

The minimum N value such that 1/N is smaller than machine precision is N = 2^23 + 1 because there are only 23 bits of mantissa in a 32-bit float. The minimum value of N such that 1/(N - 1) - 1/N is smaller than machine precision is N = 2898, which gives a result of 0.000000119121978059411048889160156l25, and machine precision is 0.00000011920928955078125. I got this value by doing a linear search in MATLAB for values of N between 2^11 and 2^12.

c)




d) It does not. I only achieved a precision of 1.9047e-09 with an N value of 2^30. One way to improve the precision of the algorithm would be to use arbitrary-precision arithmetic or use a second-order "iterative Kahan–Babuška algorithm" to increase the accuracy some more.