COMPENG 3SK3 Project 1 Report

Raeed Hassan
hassam41

January 31, 2023

1 Pseudo Code

```
% initialize singles

% divide series into 4 parts for each parallel processor

% evaluate each part using a parallel processor
p1 = increment N, divide 1/N and alternate between addition and
    subtraction to sum
p2 = increment N, divide 1/N and alternate between addition and
    subtraction to sum
p3 = increment N, divide 1/N and alternate between addition and
    subtraction to sum
p4 = increment N, divide 1/N and alternate between addition and
    subtraction to sum

% sum all parts for final result
sum = p1_sum + p2_sum + p3_sum + p4_sum
```

2 Simulation

The simulation implementation is slightly different as it puts each operation on a parallel processor instead of splitting up terms equally to do on each processor. The method described in the pseudo code is preferred as the parallel processors will not be dependent on each other (essentially becoming serial), but this simulation is easier to understand.

```
%% Initialize variables
n = single(2);
alt = single(-1);
x = single(1);
sum(1) = single(0);

% maximum number of terms calculated
N_max = 2^10;
num_err = single(zeros(1,N_max));

for N = 1:N_max
    %% processor 1
    % alternates between addition and subtraction
    alt_prev = alt;
    alt = alt * -1;

    %% processor 2
    % increments N
    n_prev = n;
```

```
    n = n + 1;

    %% processor 3
    % divide 1 / N
    x_prev = x;
    x = alt_prev / n_prev;

    %% processor 4
    % adds x to sum
    sum(N+1) = x_prev + sum(N);
end
```

3 Design Decision

The pseudo code design splits groups of terms and executes these terms on serially on parallel processors. This allows each processor to function independently from each other. The simulation instead puts each individual operation onto a parallel processor and performs operations for any term N on all parallel processors. This was done to make coding the problem simpler, as performance was not a concern for such a simple program.

4 Questions

4.a High Precision

It is not possible to achieve any high precision you desire by summing up the first N terms of the series as the IEEE 32-bit floating number system inherently has precision limitations. The sum of the series, which would also be represented in the IEEE 32-bit floating number system, would inherit the same precision limitations of the system and would have an upper-bound limit on its precision.

4.b Minimum N

The minimum N value such that 1/N becomes too small to be representable is N = 2^{128}.

The minimum N value such that 1/(N-1)-1/N becomes too small to be representable is N = $2^{24} + 4$.

The minimum N value such that 1/N is smaller than machine precision is N = $2^{23} + 1$.

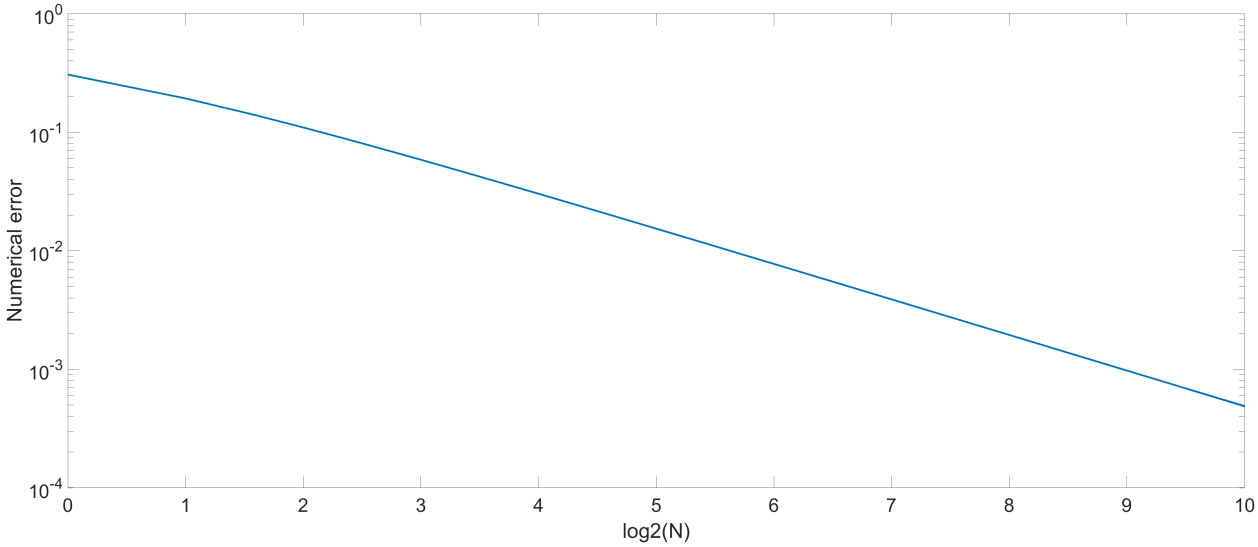
The minimum N value such that 1/(N-1)-1/N is smaller than machine precision is 2898.

4.c Numerical Error Against N

The numerical error against N is plotted in Figure 1. The error curve plots the numerical error against N from N = 1 to N = 2^{10}. We can observe the error curve decreases as the

value of N increases. This means the value of the sum of the alternating harmonic series gets closer to $\ln 2$ as the number of terms in the series increases.

Figure 1: Numerical Error against N



4.d Highest Possible Precision

This algorithm does not achieve the highest possible precision. For example, precision is lost during the accumulation process as the result after each summation is truncated. The algorithm could be improved by implementing a method to reduce numerical error by using the Kahan summation algorithm, also called compensated summation.

This method works by keeping maintaining a separate variable which accumulates the error of the summation to compensate the sum at the end.