CENG 371 - Scientific Computing
Spring 2022
Homework 1

Kargı, Bora
e2380566@ceng.metu.edu.tr

April 3, 2022

1 Question 1

a) The plot for the function $g(n)$ is :

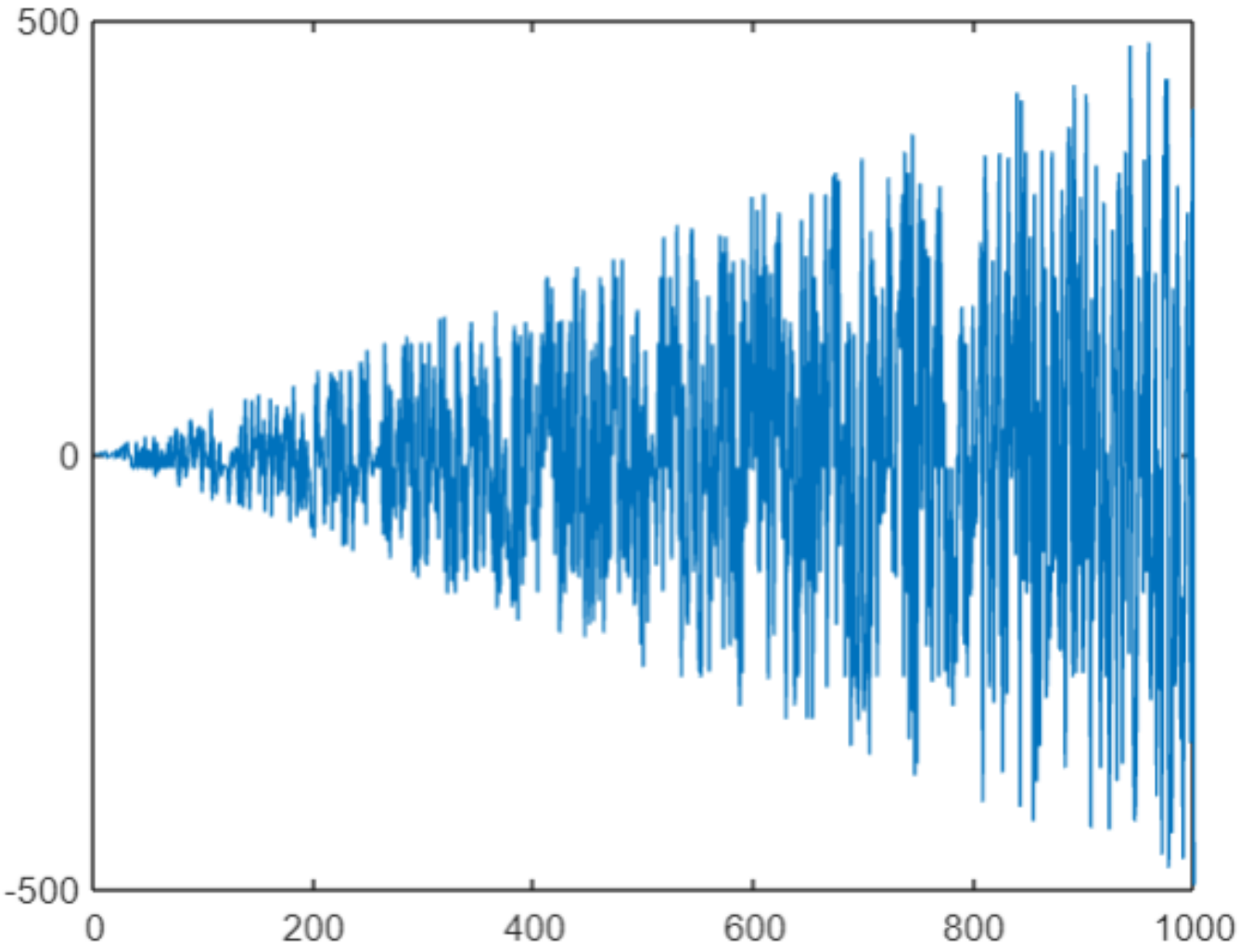

Figure 1: g(n) function.

where x-axis shows the n and y axis shows the $g(n)/\epsilon$.

b) After inspecting the results, we find that the values which makes this function 0 are actually the power of 2.

c) $g(n) = 0$ if only if the $f(n)$ function is exactly zero. Since there will be error included in the floating point operations, g(n) is zero if only if we can represent it without having an error : because computers use binary, only the operations involving power of two can be calculated perfectly,without an error as it can be seen from part b.

Figure 2: Printed result of the values where g(n) = 0.

d) As n increases, error boundary of the operation also increases. This can be shown with a theoritical analysis. Denote ϵ as a error boundary for floating point operations (depends on whether it is single or double, epsilon is 2^{-24} or 2^{-53}). So for all operation *op* between two floating point machine numbers a, b, we have $a\ op\ b = (a\ op\ b)(1+\delta)$ where $|\delta| \leq \epsilon$.

1. First step is, calculating the error in the operation $\frac{n+1}{n}$. Since $n+1$ is also a machine number we know that it has no error so we dont have to expect seperate error coming from operation $n+1$. $fl(op)$ indicates the result of the operation.
$$fl(\frac{n+1}{n}) = \frac{n+1}{n}(1+\delta_1)$$
2. Now for the next step, we have
$$fl(\frac{n+1}{n} - 1) = (\frac{n+1}{n}(1+\delta_1) - 1)(1+\delta_2)$$
$$= \frac{n+1}{n}(1+\delta_1)(1+\delta_2) - (1+\delta_2)$$
3. Moving on to the third step, we have multiplication operation.
$$fl(n(\frac{n+1}{n} - 1)) = (\frac{n+1}{n}(1+\delta_1)(1+\delta_2) - (1+\delta_2))n(1+\delta_3)$$
$$= (n+1)(1+\delta_1)(1+\delta_2)(1+\delta_3) - n(1+\delta_2)(1+\delta_3)$$
4. Lastly, we subtract 1 from the final result, so the result (with error included) of the $f(n)$ is :
$$fl(f(n)) = fl(n(\frac{n+1}{n} - 1) - 1) = ((n+1)(1+\delta_1)(1+\delta_2)(1+\delta_3) - n(1+\delta_2)(1+\delta_3) - 1)(1+\delta_4)$$
$$= (n+1)(1+\delta_1)(1+\delta_2)(1+\delta_3)(1+\delta_4) - n(1+\delta_2)(1+\delta_3)(1+\delta_4) - (1+\delta_4)$$
5. Notice that $(1+\delta_1)(1+\delta_2) = (1+\delta_1+\delta_2+\delta_1\delta_2)$. Since $\delta_1\delta_2$ is small, we can neglect it (it is smaller than ϵ^2). We can also use another parameter for $\delta_1 + \delta_2$. Using approach similar to this, we can neglect the error terms combined in multiplication and only get the errors that are added. Let :
   (a) $(1+\delta_i) = (1+\delta_1)(1+\delta_2)(1+\delta_3)(1+\delta_4)$
   (b) $(1+\delta_j) = (1+\delta_2)(1+\delta_3)(1+\delta_4)$
   (c) $(1+\delta_k) = (1+\delta_4)$

   So our equation becomes $(n+1)(1+\delta_i) - n(1+\delta_j) - (1+\delta_k)$. Since we have summed errors in while combining the error terms, boundaries of δ_i, δ_j and δ_k can be found as:
$$\delta_i = \delta_1 + \delta_2 + \delta_3 + \delta_4$$
$$\delta_j = \delta_2 + \delta_3 + \delta_4$$
$$\delta_k = \delta_4$$
   Also, using the triangle inequality, we have:
$$|\delta_i| \leq 4\epsilon$$
$$|\delta_j| \leq 3\epsilon$$
$$|\delta_k| \leq \epsilon$$

Now we found the result of the operation, lets analyze the absolute error of the function. We can not find the relative error due to the fact that actualy theoritical result of this function is 0 at all cases. Absolute error can be calculated as:
$$AbsErr(f(n)) = |f(n) - fl(f(n))| = | - fl(f(n)))|$$
$$= |(n+1)(1+\delta_i) - n(1+\delta_j) - (1+\delta_k)|$$
$$= |(n+1)(\delta_i) - n(\delta_j) - \delta_k|$$
$$= |n(\delta_i - \delta_j) + (\delta_i - \delta_k)| \leq |n(\delta_i - \delta_j)| + |\delta_i - \delta_k| = n|\delta_1| + |\delta_1 + \delta_2 + \delta_3| \leq n\epsilon + 3\epsilon$$

.

So the boundary of the absolute error grows with n. As $n \to \infty$, absolute error approachs infinity for the cases where error is not 0 (n is not multiple of two).

2 Question 2

a) For the first question, we have to compute

$$\sum_{n=1}^{10^6} 1 + 10^{-2} + 10^{-8} - n10^{-8}$$

which is equal to

$$10^6 + \frac{10^4}{2} + \frac{3 \times 10^-2}{2} = 1005000.015$$

b) Pairwise summation is a technique for summmation where we add values after number of element is less than a base case that is defined. After each iteration of adding pairs, we add the results in a similar, pair-wise way until we end up with the final result. This summation technique accumulates less error than the naive summation technique.

c) The result of the naive sum, pairwise sum and compansated sum given in the Figure 3.

```
_____________________________________________
ALGORTIHM : NAIVE SUM ALGORTIHM
Calculated sum : 1005000.0049999995389953
Error : 0.0000000004656613
Relative Error : 0.0000000000000005
Total execution time : 0.0075240000000000
_____________________________________________
_____________________________________________
ALGORTIHM : PAIRWISE SUM ALGORTIHM
Calculated sum : 1005000.0049999998882413
Error : 0.0000000001164153
Relative Error : 0.0000000000000001
Total execution time : 0.3523420000000000
_____________________________________________
_____________________________________________
ALGORTIHM : COMPANSATED SUM ALGORTIHM
Calculated sum : 1005000.0050000000046566
Error : 0.0000000000000000
Relative Error : 0.0000000000000000
Total execution time : 0.0105050000000000
_____________________________________________
```

```
_____________________________________________
ALGORTIHM : NAIVE SUM ALGORTIHM (SINGLE)
Calculated sum : 1002466.6875000000000000
Error : 2533.3173828125000000
Relative Error : 0.0025207137223333
Total execution time : 0.0084810000000000
_____________________________________________
_____________________________________________
ALGORTIHM : PAIRWISE SUM ALGORTIHM (SINGLE)
Calculated sum : 1005000.0000000000000000
Error : 0.0049999998882413
Relative Error : 0.0000000049751243
Total execution time : 0.5648220000000000
_____________________________________________
_____________________________________________
ALGORTIHM : COMPANSATED SUM ALGORTIHM (SINGLE)
Calculated sum : 1005000.0000000000000000
Error : 0.0049999998882413
Relative Error : 0.0000000049751243
Total execution time : 0.0106170000000000
_____________________________________________
```

Figure 3: Results of summing algorithms in single and double precision.

d) Looking at the figures, we can see that Pairwise algorithm is the slowest one (reason explained in subpart e) and the naive sum algorithm is the fastest. In double precision the results are similar. However naive sum algorithm makes more error on single precision whereas pairwise and compensated sum algorithms are less error prone compared to naive algorithm.

e) From theoritical analysis for the upper bounds for errors, we know that naive sum algorithms performs more poorly than others. Runtimes, on the other hand, is more favorable for naive sum algorithm since it is the most natural sum algorithm for adding elements of arrays. Pairwise sum uses stack and compensated sum uses additional operation to decrease the error that is accumulated.

Pairwise algorithm depends on the base-case where we begin to sum the values. In Figure 3, we have used base-case as 2 (which is very bad in practice since this is a divide-conquer algorithm and bigger the base case, less recursive overhead there is). For comparison, Figure 4 includes the results with different base cases. It can be seen that higher the base case, faster the algorithm so in terms of speed, pairwise is not slow for high base cases. Pairwise is great in terms of parallelism - it is easy to see that we can use multiple process/threads to compute results faster. Since we sum each part independently we can make each processor/thread handle different part in the summation. This is not applicable for naive and compensated sum algorithm.

Additional operation in compensated sum makes it more slower than naive sum, but better since it is designed to decrease the error.

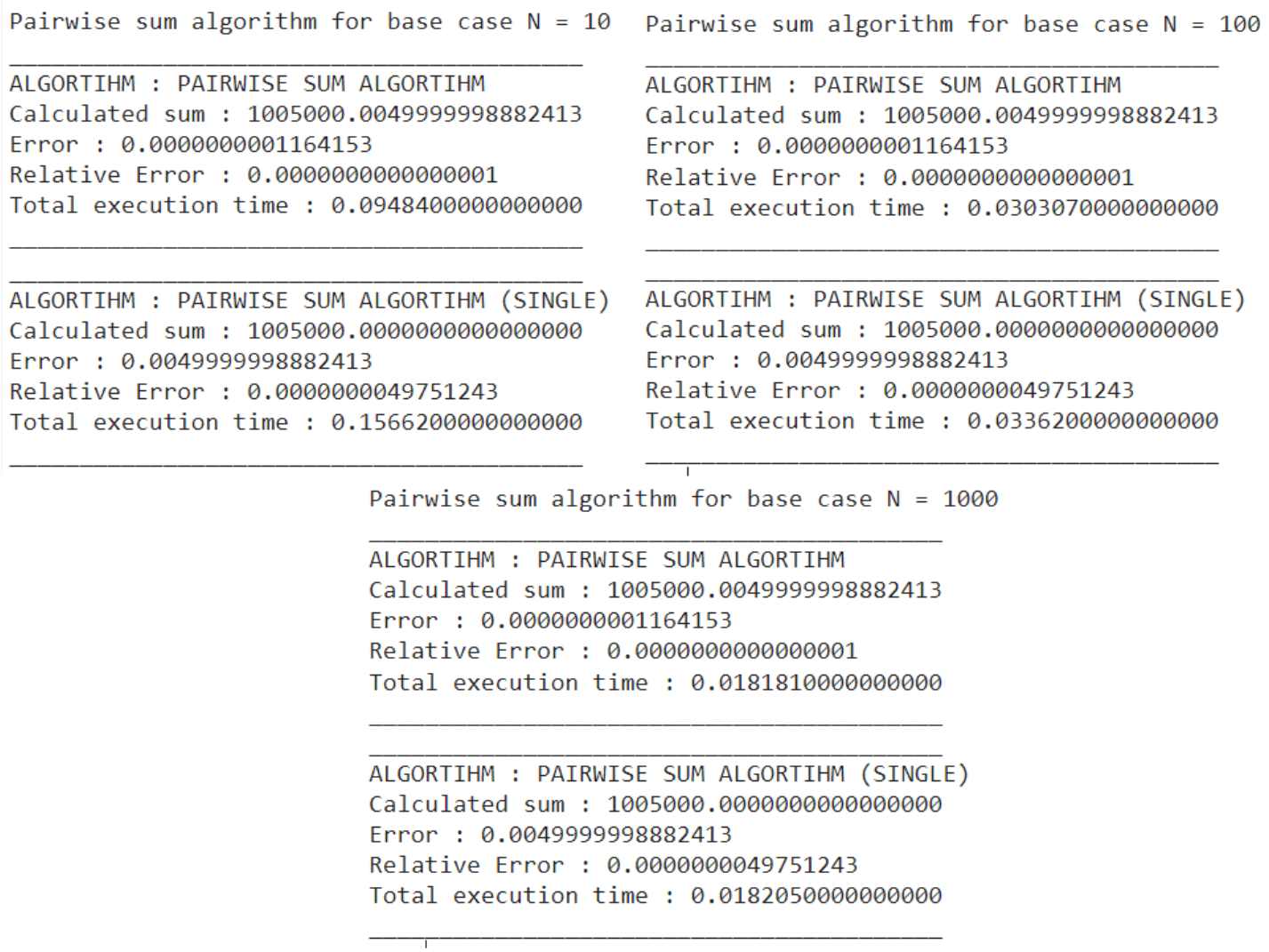

Figure 4: Compariosn of running time for different base cases used for pair-wise summation algorithm.