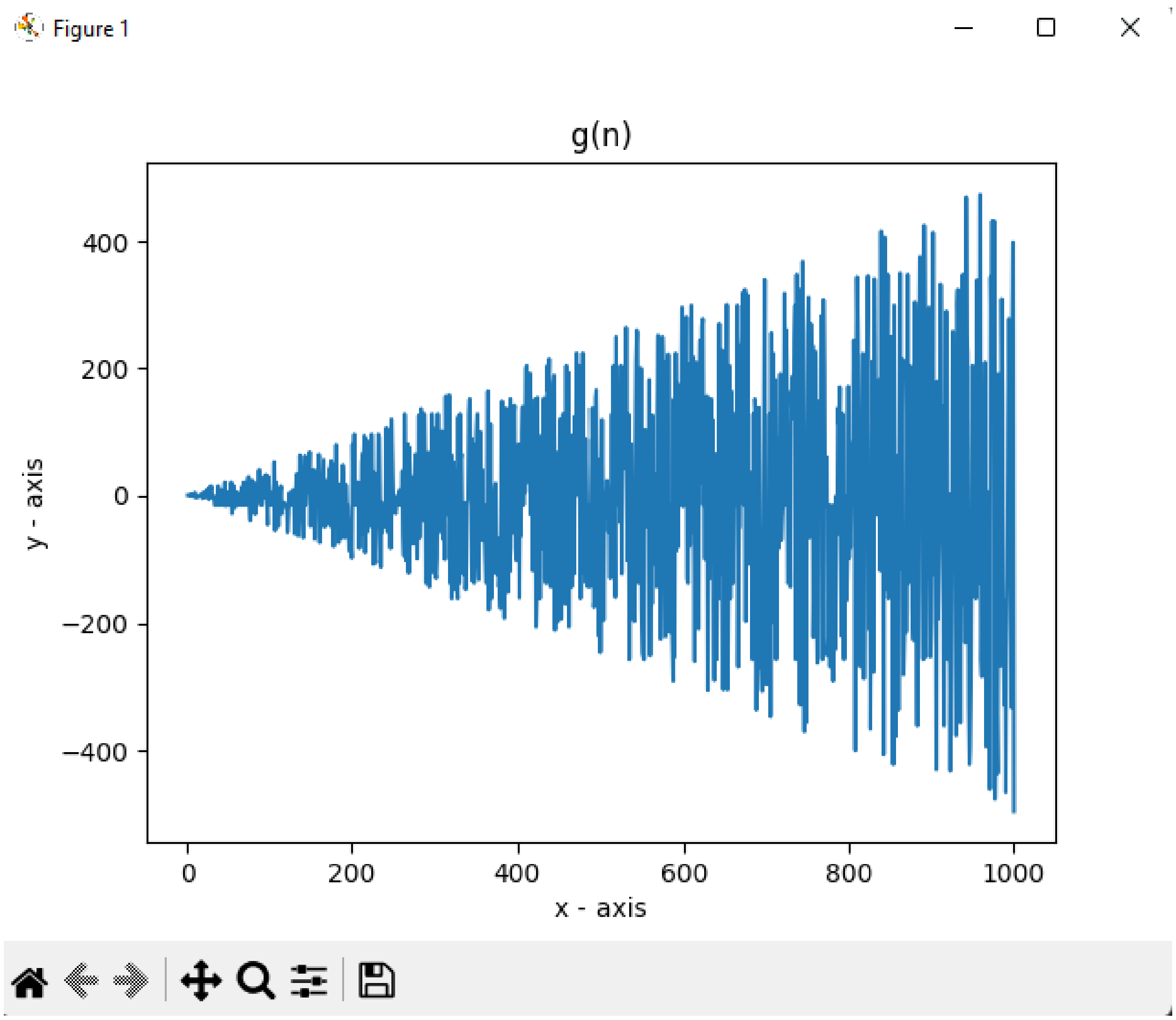
Question 1)

a)



b)

For values {1,2,4,8,16,32,64,128,256,512}, g(n) = 0

c)

Due to floating point arithmetic, most decimal fractions cannot be represented exactly as binary fractions. Because of this, the decimal floating point numbers are only approximated by the binary floating point numbers which are stored in the machine.

For instance in python;

```
.1 + .1 + .1 == .3
False
```

f(n) = 0 when we use distribution law and we should have gotten g(n) = 0 for all values of n. However since python doesn't do simplification and directly computes (n+1)/n and so on, due to rounding errors, we get nonzero for majority of ns.

d)

Since we have binary representation of floating-point numbers, we get correct result(g(n) = 0) only the values which are power of 2's. Thus; we have exponential growth in size of g(n) since g(n) = 0 only for $n \in 2^a$ where $a \in \mathbb{N}$ (i,e more values between 512-1024 than 2-4).

Question 2)

a) 1005000.005

b)

The technique of pairwise execution is used to accelerate the calculation of long sequences of associative operations (for instance, mass summations). At each stage, the current array is first partitioned into pairs, and then the elements in each pair are summed.

c)

Naïve Summation: Single=1002466.7,Double=1005000.0049999995

Compensated Summation: Single= 1005000.0,Double=1005000.005

Pairwise Summation: Single=1005000.0,Double=1005000.0049999999

d)

In terms of time complexity:

Naïve Summation: O(n)

Compensated Summation: O(n)

Pairwise Summation: O(n)

The relative error bound for naive summation that grows as O(εn), compensated summation has O(ε),which is independent of n and pairwise summation has an error that grows as O(εlogn) on worst-case. That's why we get the best result in compensated summation and worst in naïve summation.

e)

We got more accurate results for double precision since we have 64bit representation for numbers instead of 32bits. Even in both cases, we can not represent every single reel numbers without losing any precision, with 64bit representation rounding errors are much more smaller. For instance in naïve summation with single precision, we had more than 2000 difference between the result and the correct answer. However in double precision, we almost got the correct result even naïve summation is the worst algorithm compared to others.

For more accurate results, we should always use double precision if space complexity isn’t an issue and also use the compensated summation algorithm since it is less error-prone O(ε).

Bartu Kılıçkaya / 2380640