Etude 6 - Numbers

Pair Members: Anthony Dickson, Marcus Lee

In this report we will consider results that are within 5% of eachother to agree.

1 Harmonic Numbers

Harmonic Numbers The harmonic numbers are defined as the initial partial sums of the series:

$$\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\dots$$

The above can also be represented as follows:

$$H(n) \;=\; \frac{1}{1} \;+\; \frac{1}{2} \;+\; \frac{1}{3} \;+\; \dots \;+\; \frac{1}{n-1} \;+\; \frac{1}{n}$$

Method Using the above formula, we created a java application which calculates $H(n)$ using both single and double precision floating point numbers. Our java application computes $H(n)$ in both order of largest term first ($\frac{1}{1}+\frac{1}{2}+\dots+\frac{1}{n}$) and reverse order ($\frac{1}{n}+\frac{1}{n-1}+\dots+\frac{1}{1}$).

Observations

Largest Term First The results for both single and double precision agree up to somewhere around $n = 10,000,000$. At $n = 10,000,000$ there is a 8.05% difference between the single and double precision results (figure 1.2). At $n = 100,000,000$ the difference is 20.90%, and at $n = 1,000,000,000$ the difference is 32.13%.

Figure 1.1: Results of calculating $H(10,000)$**.**

```
Comparison of Single and Double Precision
Largest Term First
Single Precision: 9.787613
Double Precision: 9.787606036044348
Percent Difference: < 0.01%
agree: true

Reverse Order
Single Precision: 9.787604
Double Precision: 9.787606036044386
Percent Difference: < 0.01%
agree: true
```

Reverse Order When computing harmonic numbers in reverse order, both the single and double precision answers agree. The double precision result also remains consistent with the double precision results computed in order of largest term first.

Although in practice the order in which terms are added together does not matter, it seems to matter quite a bit when using floating point numbers. Its importance can be seen when computing harmonic numbers for where $n > 1,000,000$. Reverse order seems to eliminate some of the inaccuracy of single precision numbers. Based on our observations, reverse order seems to be more accurate than largest term first, however we are not sure on how to test this hypothesis.

Single Precision Inaccuracy We think that the inconsistencies in the single precision results are due to the way binary fractions work. Some fractions cannot not be accurately represented in binary, similar to how fractions like 1/3, 1/6 and 1/9 cannot be accurately represented in decimal.

The fraction $\frac{1}{10}$ can be represented as 0.1 in decimal, or even as $0 \times 10^0 + 1 \times 10^{-1}$. Binary fractions are similar, but instead of base 10, they use base 2. So the binary fraction 0.11 could be represented as $0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$, or $0 + \frac{1}{2} + \frac{1}{4}$, or even 0.75.

This works fine for some numbers (in particular fractions where the denominator is a power of 2), but not all numbers can represented exactly in binary fractions. For example, in binary the fraction $\frac{1}{10}$ is represented in binary as $0.0\overline{0011}$. When converted to decimal, this binary fraction produces the number 0.100000001490116125. From this, it is clear that adding many binary fractions together is bound to be innacurate.

Figure 1.2: Results of calculating $H(10,000,000)$.

```
Comparison of Single and Double Precision
Largest Term First
Single Precision: 15.403683
Double Precision: 16.695311365857272
Percent Difference: 8.05%
agree: false

Reverse Order
Single Precision: 16.686031
Double Precision: 16.695311365859965
Percent Difference: 0.06%
agree: true

Comparison of Computation Orders
Single Precision
Largest Term First: 15.403683
Reverse Order: 16.686031
Percent Difference: 7.99%
agree: false

Double Precision
Largest Term First: 16.695311365857272
Reverse Order: 16.695311365859965
Percent Difference: < 0.01%
agree: true
```

Summary The result is bound to be innaccurate when adding a large number of single precision floating point numbers together.

Double precision numbers seem to have enough precision that they do not suffer as much from the issues that single precision numbers do. The double precision results remained consistent up to the harmonic number where $n = 1,000,000,000$.

Computing harmonic numbers in reverse order seems to remedy some of the innaccuracy of single precision numbers.

2 Standard Deviation

Definitions In this report method one refers to the population standard deviation calculated as follows:

$$\sigma = \sqrt{\frac{\sum (a_i - \bar{a})^2}{n}}$$

and method two refers to the population standard deviation calculated as follows:

$$\sigma = \sqrt{\frac{\sum a_i^2 - (\sum a_i)^2/n}{n}}$$

The fixed value added to each number in a is denoted by c.

Method We created a java application which copmutes the standard deviation using both method one and method two. For each method we computed the result in single and double precision. The values for a were generated randomly for most tests. Some tests added a fixed number to each number in a. Values used for n ranged from 10 to $1,000,000$, with values for the fixed value in a similar range.

Observations From small n values (10-100) to large n values (around 1 million) there seems to be very little difference in the results produced by method one and method two. This can be seen in figures 2.1-3.

However, when large values for c are used method two starts to show some inconsistencies. For example, in figure 2.4 we can observe a difference of about 8.3 (roughly 30% of 27.9) between the single and double preicision results of method two.

Figure 2.1: Test with small n value.
$n = 100$
$c = 0$

Method (1):
Single Precision: $\sigma = 27.901434$
Double Precision: $\sigma = 27.90143365492175$

Method (2):
Single Precision: $\sigma = 27.90079$
Double Precision: $\sigma = 27.900786727259145$

In figure 2.5 a c value many times bigger than n was chosen and again, the single precision computation yielded odd results, this time being way off. This may be due to the term $(\sum a_i)^2/n$ becoming too large, possibly causing an integer overflow.

Figure 2.5: Test with large fixed number.
$n = 100$
$c = 10,000,000$

Method (1):
Single Precision: $\sigma = 27.100739$
Double Precision: $\sigma = 27.100737997331365$

Method (2):
Single Precision: $\sigma = 3,276.8$
Double Precision: $\sigma = 27.092065259038485$

Double precision numbers did not suffer from the problems that single precision numbers did in these experiments. The double precision results remained relatively identical between method one and method two.

Conclusion • The two methods used to compute standard deviation did not always agree.

• The level of precision only sems to matter with negative numbers (figure 2.4) or large numbers (figure 2.5).

• Adding a fixed value to all the elements of a sequence of numbers should not change the standard deviation. Most of our implementations showed this, except for the single precision implementation of method two.

• Method one should generally be preferred over method two because method one did not suffer from issues with large numbers in our experiments.

3 An Identity

For identity task, Java code (Identity class) has considered given equation and calculated output for single precision and double precision data types. Output for same is as per figure 3.1.

As per the above output for different inputs, Single precision till single decimal points satisfy the equation. if there are more than one decimals after points then even for single precision point equations fails even though it is mathematically correct. In case of Double precision it fails every time irrespective of given input.

4 Figures

Figure 1.3: Results of calculating $H(100000)$.
Largest Term First:
Single Precision: 12.090851
Double Precision: 12.090146129863335

Reverse order:
Single Precision: 12.090153
Double Precision: 12.090152

Figure 2.1: Test with small n value.
$n = 100$
$c = 0$

Method (1):
Single Precision: $\sigma = 27.901434$
Double Precision: $\sigma = 27.90143365492175$

Method (2):
Single Precision: $\sigma = 27.90079$
Double Precision: $\sigma = 27.900786727259145$

Figure 2.2: Test with small n and fixed values.
$n = 100$
$c = 10$

Method (1):
Single Precision: $\sigma = 27.901434$
Double Precision: $\sigma = 27.90143365492175$

Method (2):
Single Precision: $\sigma = 27.90079$
Double Precision: $\sigma = 27.900786727259145$

Figure 2.3: Test with large n value.
$n = 775,585$
$c = 0$

Method (1):
Single Precision: $\sigma = 447,564.5$
Double Precision: $\sigma = 447,616.7790107431$

Method (2):
Single Precision: $\sigma = 447,423.94$
Double Precision: $\sigma = 447,476.46364418085$

Figure 2.4: Test with large negative fixed number.
$n = 100$
$c = -10,000$

Method (1):
Single Precision: $\sigma = 27.901434$
Double Precision: $\sigma = 27.90143365492175$

Method (2):
Single Precision: $\sigma = 36.203865$
Double Precision: $\sigma = 27.900786708008454$

Figure 2.5: Test with large fixed number.
$n = 100$
$c = 10,000,000$

Method (1):
Single Precision: $\sigma = 27.100739$
Double Precision: $\sigma = 27.100737997331365$

Method (2):
Single Precision: $\sigma = 3,276.8$
Double Precision: $\sigma = 27.092065259038485$

Figure 3.1 Output:
Single Precision Input x = 2.5 y = 3.5
Single Precision Left Side: 2.5 Right Side: 2.5 Result: $true$
Double Precision Input x = 2.5848 y = 3.5234
Double Precision Left Side: 2.5848 Right Side: 2.584800000000005 Result: $false$

Single Precision Input x = 2.51 y = 3.5
Single Precision Left Side: 2.51 Right Side: 2.5100002 Result: $false$ Double Precision Input x = 2.5848 y = 3.5234
Double Precision Left Side: 2.5848 Right Side: 2.584800000000005 Result: $false$