CSC336 Assignment 1

Yujie Wu, 1003968904

May 26, 2020

Q1:
see "a1-report.pdf"

Q2:
see "a1.py"

Q3:
(a) - (b) see "a1.py"

(c)truncation error: $\sum_{n=m+1}^{\infty} \frac{1}{2n(2n-1)}$
lower bound: $\sum_{n=m+1}^{\infty} \frac{1}{2n(4n-1)}$
$= \sum_{n=m+1}^{\infty} \frac{1}{4n^2-2n}$
$\geq \sum_{n=m+1}^{\infty} \frac{1}{4n^2}$
$= \sum_{n=1}^{\infty} \frac{1}{4n^2} - \sum_{n=1}^{m} \frac{1}{4n^2}$
$= \frac{1}{4} \sum_{n=1}^{\infty} \frac{1}{n^2} - \sum_{n=1}^{m} \frac{1}{4n^2}$
$= \frac{1}{4} \frac{\pi^2}{6} - \sum_{n=1}^{m} \frac{1}{4n^2}$

plug actual value of m returned from (a): 2897, in to the formula:
$= \frac{1}{4} \frac{\pi^2}{6} - \sum_{n=1}^{2897} \frac{1}{4n^2}$
$= \frac{\pi^2}{24} - 0.4111472354359518$
$= 8.628127610482705 * 10^{-5}$, which obviously greater than machine epsilon for np.float32.
Thus, the truncation error is to blame for the significant relative error observed in (b).

Q4:
(a) When the value of "a" or "b" becomes very large, the "a" or "b" itself may not be overflowed, but the a^2 or b^2 can become overflowed. One way to solve this problem is that the part under the square root can be factorized so that there are no a, b multiplication in the formula. (such as $a * a, a * b, b * b$)

Specification for factorization steps:
$a^2 + b^2$
$= a^2 + 2ab + b^2 - 2ab$
$= (a + b)^2 - 2ab$
$= ((a + b) + \sqrt{2ab}) * ((a + b) - \sqrt{2ab})$

Then apply the square root to the entire factorized part:

$c = \sqrt{((a+b)+\sqrt{2ab}) * ((a+b)-\sqrt{2ab})}$
$c = \sqrt{((a+b)+\sqrt{2ab})} * \sqrt{((a+b)-\sqrt{2ab})}$
$c = \sqrt{((a+b)+\sqrt{2}\sqrt{a}\sqrt{b})} * \sqrt{((a+b)-\sqrt{2}\sqrt{a}\sqrt{b})}$

With this new formula, overflows that caused by $a * a$ or $a * b$ or $b * b$ will be eliminated efficiently.

(b) - (c): see "a1.py"

(d) For more than two x_{is}, we can use the similar idea as above, which is trying to make $(x_1 + x_2 + x_3 + ... + x_n)^2$. Thus, the formula will be looks like:

$$\sqrt{(x_1 + x_2 + x_3 + ... + x_n)^2 - 2x_1x_2 - 2x_1x_3 - ... - 2x_1x_n - 2x_2x_3 - ... - 2x_2x_n - ... - 2x_{n-1}x_n}$$

Then, we can treat $(x_1 + x_2 + x_3 + ... + x_n)^2$ as a and
$\sqrt{2x_1x_2 - 2x_1x_3 - ... - 2x_1x_n - 2x_2x_3 - ... - 2x_2x_n - ... - 2x_{n-1}x_n}$ as b and apply the formula:
$a^2 - b^2 = (a+b)(a-b)$
WLOG, I will use n = 4 (a, b, c, d) as demonstration for below steps:
The formula looks like $\sqrt{(a+b+c+d)^2 - 2ab - 2ac - 2ad - 2bc - 2bd - 2cd}$ when n = 4,
=
$\sqrt{(a+b+c+d+\sqrt{2ab-2ac-2ad-2bc-2bd-2cd})(a+b+c+d-\sqrt{2ab-2ac-2ad-2bc-2bd-2cd}}$
$= \sqrt{(a+b+c+d+\sqrt{abc(\frac{1}{c}-\frac{1}{b}-\frac{d}{bc}-1-\frac{d}{c}-\frac{d}{b})})(a+b+c+d-\sqrt{abc(\frac{1}{c}-\frac{1}{b}-\frac{d}{bc}-1-\frac{d}{c}-\frac{d}{b})}}$
$= \sqrt{(a+b+c+d+\sqrt{abc(\frac{1}{c}-\frac{1}{b}-\frac{d}{bc}-1-\frac{d}{c}-\frac{d}{b})})}\sqrt{(a+b+c+d-\sqrt{abc(\frac{1}{c}-\frac{1}{b}-\frac{d}{bc}-1-\frac{d}{c}-\frac{d}{b})}}$
=
$\sqrt{(a+b+c+d+\sqrt{a}\sqrt{b}\sqrt{c}\sqrt{(\frac{1}{c}-\frac{1}{b}-\frac{d}{bc}-1-\frac{d}{c}-\frac{d}{b})})}\sqrt{(a+b+c+d-\sqrt{a}\sqrt{b}\sqrt{c}\sqrt{(\frac{1}{c}-\frac{1}{b}-\frac{d}{bc}-1-\frac{d}{c}-\frac{d}{b})}}$
Then, the overflow issue can be eliminated.

However, this algorithm can potentially cause some underflow and cancellation issues. For example, if $c \approx b$, then $\frac{1}{c} - \frac{1}{b}$ may causes cancellation error, also if b and c are very large and d is very small, the value of $\frac{d}{bc}$ may be smaller than N_{min}, which means underflow. These are the possible reasons why numpy does not use this algorithm as the implementation of norm function.