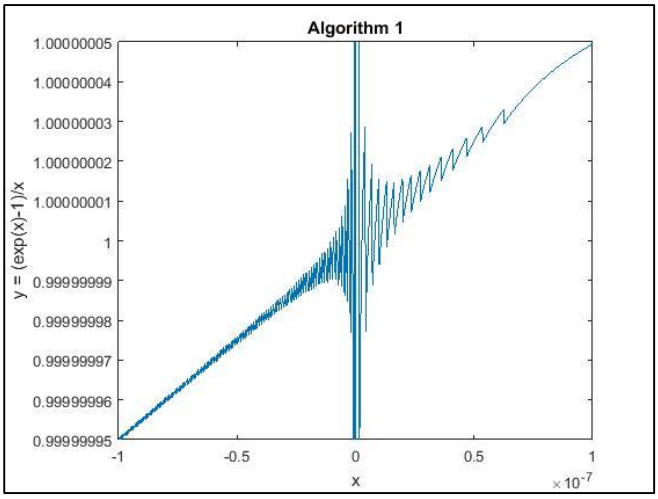
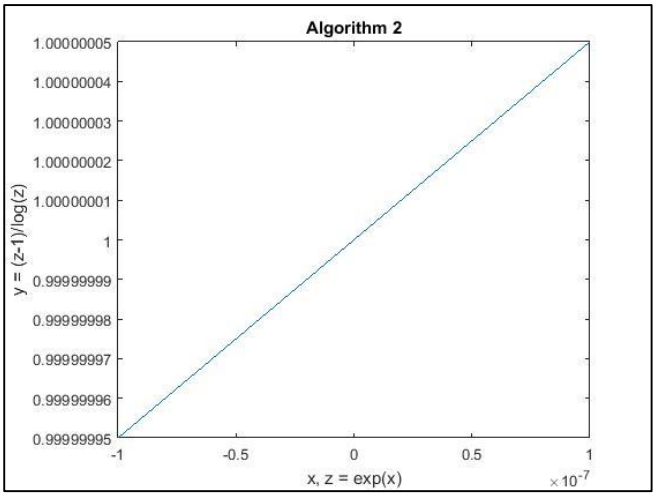
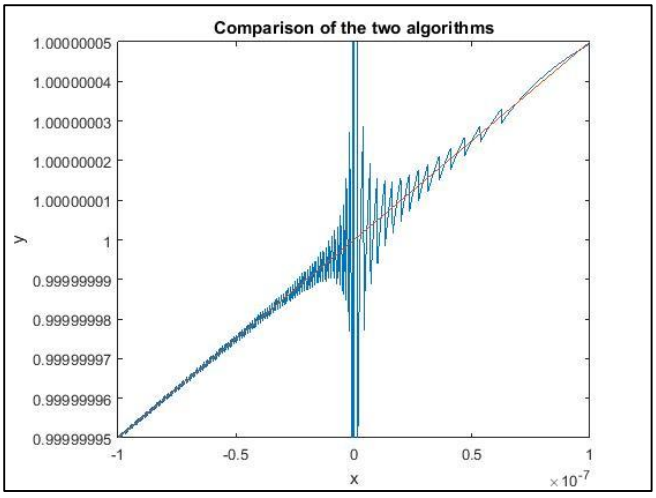
Name: Jerry Chienyu Chen
Student Number: 301228216

MACM 316 – Computing Assignment 1 Report







As shown from the figures, the effect of floating point round-off error on the same formula calculated by two different algorithms is quite vivid. Showing on the Algorithm 1 figure, as x value approaches from the outer edges from both sides towards x = 0, the y values differs a lot from points to points. This implies that the error is amplified when dividing by small values of x. On the other hand, showing on the Algorithm 2 figure, as x value approaches from the outer edges from both sides towards x = 0, the y values remain stable and precise unlike algorithm 1. Now looking at figure 3 the comparison between the two algorithms, we can see that both algorithms converge to roughly the same values as it approaches to the outer edges.

The reason behind the amplified error of algorithm 1 is that the x values that are further apart from 0 are not represented as precisely as the x values near 0 due to the round off error of the representation of floating point numbers in MATLAB. So once we do the operation on the numerator e^x-1, the error magnifies. And then afterwards as we divide it with an extremely small value of x, the error amplifies dramatically into the result showing on figure 1.

On the other hand, the reason why that the second algorithm is not having an amplified error is because the error from the Taylor expansion on the e^x term on the numerator isn't amplified by the denominator like algorithm 1. This is because that there is another Taylor expansion error on the log() term at the denominator. So relatively when an error divided by a roughly equal size error, the error will not be amplified.

Therefore, relatively speaking, algorithm 2 is more robust than algorithm 1.