**MACM316 ASSIGNMENT 1** – Ngawang Kyirong 301312227

**a.)**

A close up of a logo

Description automatically generatedA screenshot of a cell phone

Description automatically generated

**b.)**

**-** For the **first** figure, the behaviour shown is best described as the bound of the relative approximation error. This bound represents the maximum value the approximation error will take when using to approximate *e*. The robustness of this algorithm is good for these inputs given that it guarantees an approximation error of at most even though the graph is consistently fluctuating.

**-** For the **second** figure, interesting behaviour is produced at around the n= mark. Instead of the error decreasing as increases, the graph spikes back up to a constant value. In terms of robustness, this algorithm is not good as this is an error due to the machine epsilon.

**c.)**

**-** For the **first** figure, the graph shows that calculating the approximation error of *e* when using: will produce a bound of where *B* is the base and *k* is the precision. In this particular case, the floating-point representation of *e* gives a bounded error of at least . The effects of cancellation error from the floating-point subtraction of the *e* and the approximation of *e* is also on display which produces the fluctuation.

**-** For the **second** figure, the graph displays interesting behaviour around the *k=52* mark. The graph shows that when calculating the absolute approximation error of *e* when using: and , MATLAB cannot distinguish the values due to it reaching machine epsilon because now is indistinguishable from 1 at . Hence, and the graph remains stuck on the constant

**d.)**

**-** Through testing the algorithm with a variety of inputs for *c* with relative and absolute errors, the results show that if *e* was replaced with for some positive *c* and then the absolute approximation error will increase a lot as *c* increases. Contrastingly, the relative error will not increase as much due to the division of the magnitude but it will increase.

MATLAB :

x1 = linspace(0,power(10,9),65000000);

y1 = arrayfun(@(x) abs((exp(1) - power((1 + (1/x)), x)))/abs(exp(1)), x1);

p1 = semilogy(x1, y1);

xlabel('n');

ylabel('relative approximation error');

title('relative approximation error of e using (1 + 1/n)^n for increasing values of n');

x2 = power(2, 0:100);

y2 = arrayfun(@(x) abs((exp(1) - power((1 + (1/x)), x))), x2);

p2 = semilogy(log2(x2),y2);

xlabel('2^k');

ylabel('absolute approximation error');

title('absolute approximation error of e using (1 + 1/n)^n for increasing powers of 2');