Kevin San Gabriel
301342241
MACM 316 D100 Benjamin Adcock
Computing Assignment 1

Acknowledgements: Emma Hughson, Michael Lam, Brandon Frison

(a)

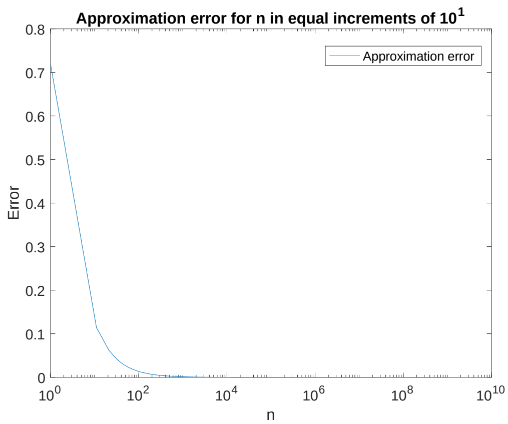

Figure 1. Error vs n graph which illustrates the change in error when approximating e with $e_n = (1+1/n)^n$ for increasing n values in equal increments of 10^1.

Note for figure 1:
- Error decreases linearly from 1 to 10^1
- The error quickly levels off to zero after n = 10^1

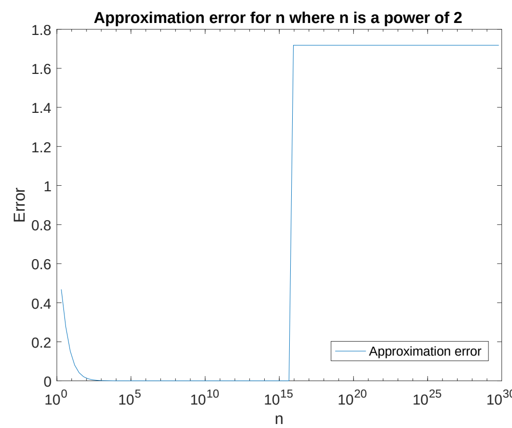

Figure 2: Error vs n graph which illustrates the change in error when approximating e with $e_n = (1+1/n)^n$ for Increasing n values which are restricted to powers of 2.

Note for figure 2:
- Similar in behaviour to figure 1
- Sudden increase in error when n = 10^{16} caused by cancellation error

(b) In each figure, the magnitude of the error in the approximation of e with e_n decreases linearly as the value of n increases from 1 to 10^1. As the value of n increases past 10^1, the error continues to decrease, however, the rate of decline of the error slows down as the magnitude of the error approaches zero.

In the case where n is being restricted to a power of 2 up to a maximum n value of 10^{30} (figure 2), the error changes to 1.7 x 10^0 when the value of n increases past 10^{16} where it then levels off and remains fixed for all n values between 10^{16} and 10^{30} inclusive.

(c) Since e_n approximates e, the approximation error observed when n = 1 to n = 10^{16} will be the result of the round-off error caused by the floating-point subtraction between e and e_n. The sudden increase in the magnitude of the error when the value of n increases past 10^{16} is caused by a cancellation error because e_n becomes nearly equal to e for those very large numbers of n.

(d) If e were to be replaced by e^c and e_n were to be replaced by $e_n^c = (1+ c/n)^n$ for some positive number c, the range of error values would start to vary depending on the value of c. If c were to be large, the initial error would be large as well and thus the range of error values would be large. If c were to be small, the range of error values would be small. In both cases, c being large or small, the algorithm would display similar behaviour to (b) for increasing values of n with the only exception being the range of error values yielded from the approximations.

Code

```
% (a)
% Algorithm for figure 1
function a = a1()
    e = exp(1);
    error_array = zeros(1, length(1:power(10,1):power(10,9)));

    i = 1;
    for n = 1:power(10,1):power(10,9)
        en = power((1+(1/n)),n);
        error_array(i) = abs(e - en);
        i = i + 1;
    end

    semilogx(1:power(10,1):power(10,9), error_array);
    xlabel('n','fontsize',16);
    ylabel('Error','fontsize',16);
    title('Approximation error for n in equal increments of 10^1');
    legend('Approximation error');
    set(gca,'FontSize',14);

    a = "Computation complete.";
end

% Algorithm for figure 2
function a = a1()
    e = exp(1);
    error_array = zeros(1, 99);
    i = 1;

    for n = 1:99
        en = power((1+(1/power(2,n))),power(2,n));
        error_array(i) = abs(e - en);
        i = i + 1;
    end

    semilogx(power(2,1:99), error_array);
    xlabel('n','fontsize',16);
    ylabel('Error','fontsize',16);
    title('Approximation error for n where n is a power of 2');
    legend('Approximation error')
    set(gca,'FontSize',14);

    a = "Computation complete.";
end

% (d)
% Algorithm for part d
function a = a1()

    c = 5;
    e = power(exp(1),c);
    error_array = zeros(1, length(1:power(10,1):power(10,9)));

    i = 1;
    for n = 1:power(10,1):power(10,9)
        en = power((1+(c/n)),n);
        error_array(i) = abs(e - en);
        i = i + 1;
    end

    semilogx(1:power(10,1):power(10,9), error_array);
    xlabel('n','fontsize',16);
    ylabel('Error','fontsize',16);
    set(gca,'FontSize',14);

    a = "Computation complete.";
end
```