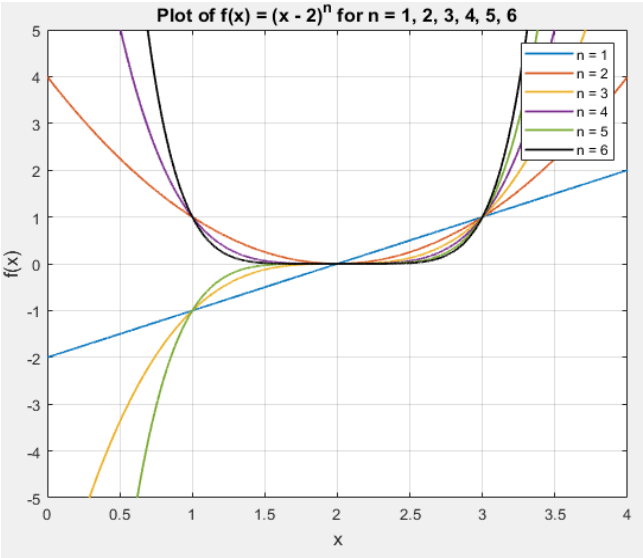
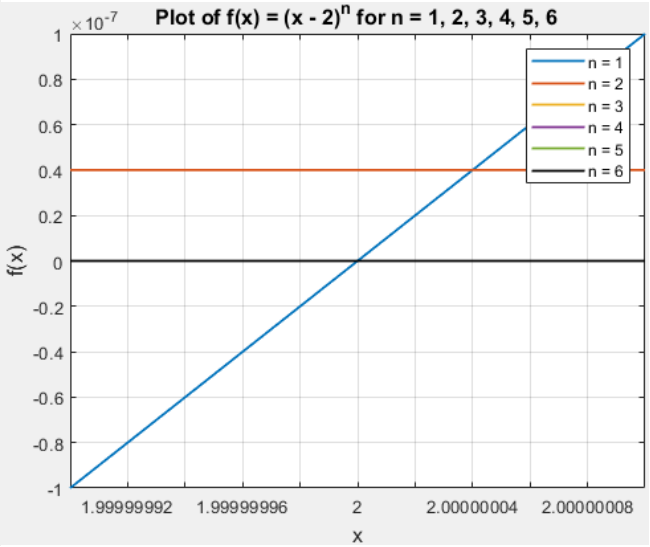
MACM 316: CA1 **AUTHOR: Andy Liu STUDENT ID: 301472847**

Q1 (Zoom to see figures): The figures below show the polynomial function $(x-2)^n$ for n = 1,2,3,4,5,6. From both the zoomed out and zoomed in figures, there is next to no visible floating-point error. These plots will represent the "exact result". One thing of note is that the function $(2-2)^2$ for some reason does not equal to 0 like the rest of the functions. It instead differs by a very small margin. I was not able to figure out why this is the case but assume it's due to an unknown floating-point rounding error occurring.

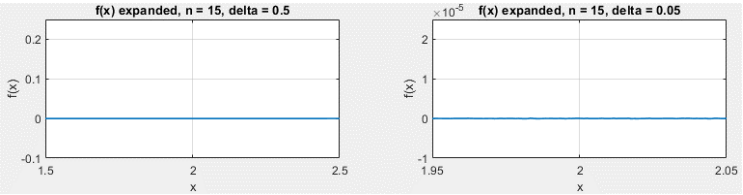


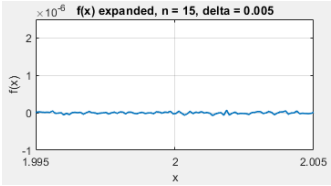


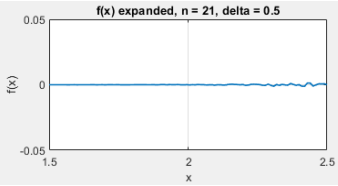
Q2: My implementation of fexpand does not allow wrong inputs like non-numeric characters, negative numbers for n, and non-integers for n. As seen below, results are correct and error messages work.

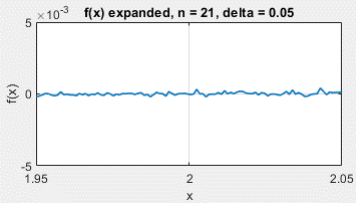
```
>> A1
Where a = 12 and n = 5, f(7) = -3125
Error using fexpand
Inputs a, n, and x must be numeric, and n must be a non-negative integer.

Error in A1 (line 38)
result = fexpand(expand_a1, expand_n1, expand_x1);
```

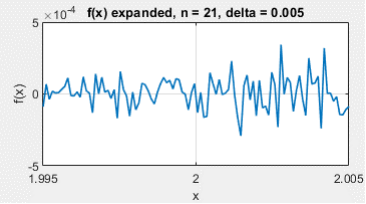
Q3 (Zoom to see figures): The below figures show the expanded $(x-2)^n$ function graphed using n = 15 and n = 21. The graphs are "zoomed in" with progressively smaller x-intervals, $x \in [2-\delta,\ 2+\delta]$ for $\delta = 0.5, 0.05, 0.005$. From the graphs, we can see that as both n and delta increase, the "straight" lines in the graph become more jagged and erratic. This is due to floating-point rounding errors becoming more prevalent in the curves. The reason this occurs when delta increases is because the x-interval also becomes smaller leading to higher accuracy. This higher accuracy allows small details in rounding errors to be seen. As n increases, however, the erratic behaviour seen in the lines are much more pronounced and seemingly increases exponentially as n increases. This is because the expanded form of the function uses more arithmetic operations like multiplication, division, and exponentiations which lead to an accumulation of floating-point errors that increases with n. The exponentially larger values calculated as n increases will also increase the severity of errors. Double-precision floating-point arithmetic simply isn't accurate enough in its calculations. These errors aren't in the "exact result" $(x-2)^n$ plotted above because the arithmetic is simpler and within the limits of MATLAB. The smallest n value for which the below plots differ significantly from the "exact" plots of $(x-2)^n$ is n = 15. Starting from n = 15, the floating-point errors become noticeable in the delta = 0.05 and delta = 0.005 plots. The reason I chose n = 15 to display as a plot is for this exact reason. I then chose n = 21 because it clearly depicts the increase in floating-point errors as n increases.











Q1 and Q2: Done in separate files where Q2 is done in fexpand.m

```
% Computing Assignment #1: Q1 and Q2
% Author: Andy Liu
% ID: 301472847

% Q1: EXACT RESULT
x = linspace(0, 4, 10000);
n_values = [1,2,3,4,5,6];
line_colors = [
    0.00, 0.45, 0.74;  % Blue
    0.85, 0.33, 0.10;  % Red
    0.93, 0.69, 0.13;  % Yellow
    0.49, 0.18, 0.56;  % Purple
    0.47, 0.67, 0.19;  % Green
    0.00, 0.00, 0.00   % Black
];

for i = 1:length(n_values)
    n = n_values(i);
    y = (x - 2).^n;
    plot(x, y, 'DisplayName', ['n = ' num2str(n)], 'Color', line_colors(i, :), 'LineWidth', 1.3);
    hold on;
end

% Add labels, title, and set axis limits
xlabel('x', 'FontSize', 12);
ylabel('f(x)', 'FontSize', 12);
title('Plot of f(x) = (x - 2)^n for n = 1, 2, 3, 4, 5, 6', 'FontSize', 12);
legend('show');
grid on;
xlim([0, 4]);
ylim([-5,5]);

% Q2
expand_a = 12; expand_n = 5; expand_x = 7;
result = fexpand(expand_a, expand_n, expand_x);
disp(['Where a = 12 and n = 5, f(', num2str(expand_x), ') = ', num2str(result)]);
%expand_a1 = 3; expand_n1 = -12; expand_x1 = 5;
%result = fexpand(expand_a1, expand_n1, expand_x1);
%disp(['Where a = 3 and n = -12, f(', num2str(expand_x1), ') = ', num2str(result)]);
```

```
% Computing Assignment #1: fexpand.m
% Author: Andy Liu
% ID: 301472847

function fx = fexpand(a, n, x)
    % Check for invalid inputs
    if ~isnumeric(a) || ~isnumeric(n) || ~isnumeric(x) || n < 0 || floor(n) ~= n
        error('Inputs a, n, and x must be numeric, and n must be a non-negative integer.');
    end

    % Initialize polynomial value and compute the expanded form using the binomial theorem
    fx = 0;
    for k = 0:n
        binomial_coeff = nchoosek(n, k);
        fx = fx + (-a).^k * binomial_coeff * x.^(n - k);
    end
end
```

Q3: MATLAB implementation below

```
% Computing Assignment #1: Q3
% Author: Andy Liu
% ID: 301472847

delta_values = [0.5, 0.05, 0.005];
n_values = [15, 21];
y_limits = [[-0.1, 0.25]; [-0.00001, 0.000025]; [-0.000001, 0.0000025]; [-0.05, 0.05]; [-0.005, 0.005]; [-0.0005, 0.0005]];

% Define the x-intervals for zooming in near x = 2
x_intervals = cell(length(delta_values), 1);
for i = 1:length(delta_values)
    delta = delta_values(i);
    x_intervals{i} = linspace(2 - delta, 2 + delta, 100);
end

figure;
plot_legend = cell(length(delta_values) * length(n_values), 1);
plot_index = 1;

for n_index = 1:length(n_values) % 2 loops that create plots for each n and delta
    n = n_values(n_index);
    for delta_index = 1:length(delta_values)
        delta = delta_values(delta_index);
        x = x_intervals{delta_index};

        % Calculate y using the fexpand function
        y = fexpand(2, n, x);  % Here, a = 2

        subplot(length(delta_values), length(n_values), plot_index);
        plot(x, y, 'LineWidth', 1.2);
        title(['f(x) expanded, ', 'n = ', num2str(n), ', delta = ', num2str(delta)]);
        xlabel('x');
        ylabel('f(x)');
        grid on;

        if (n_index == 2) % chooses limits based on plot to better show lines
            limit = y_limits(delta_index + 3, :);
        else
            limit = y_limits(delta_index, :);
        end
        ylim([limit(1), limit(2)]);

        plot_legend{plot_index} = ['n = ', num2str(n), ', delta = ', num2str(delta)];
        plot_index = plot_index + 1;
    end
end
```