## Contents

## Zack Humphries

COMP 521 HW1 Problem 2

```
clc;       % clear command window
clear;     % removes all saved variables
close all; % close any open windows
```

## Part 1

Both 2 and 5 terms match well with f(x), however using 5 terms is much closer to matching f(x) than just 2 terms, however, the 5 terms will trend further from f(x) the further we get away from 0=a.

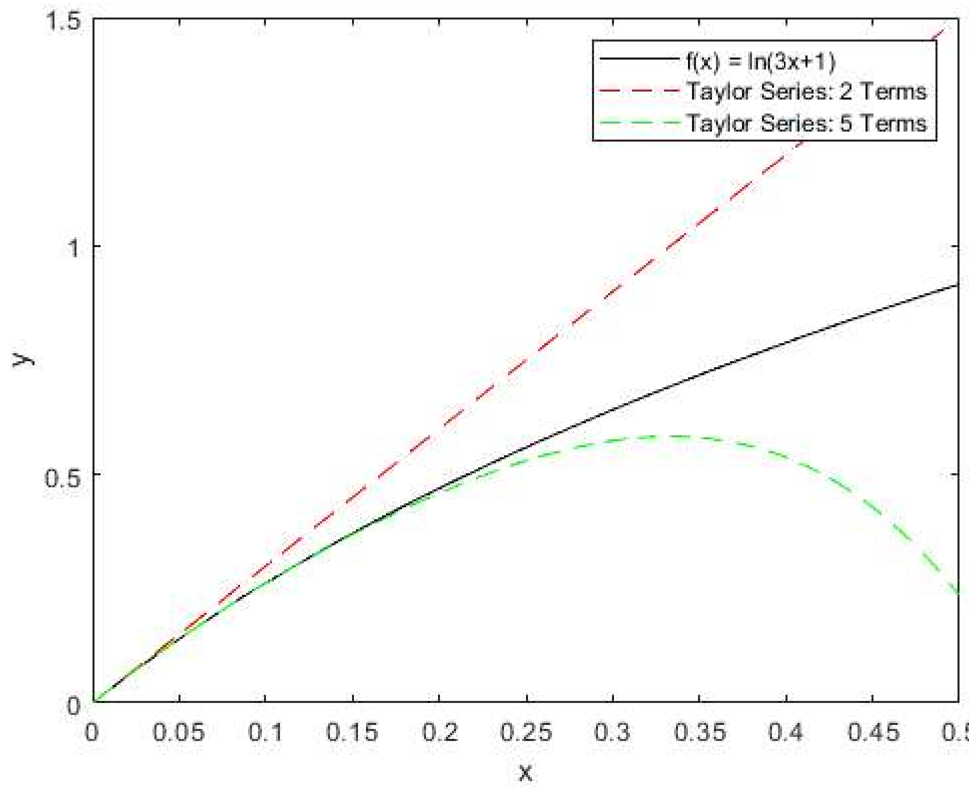
```
delta_x = 0.001;
x_list = 0 : delta_x : 0.5;

% f(x) = f(a) + f'(a)(x-a)/1! + f''(a)((x-a)^2)/2! + ....

y_2_terms = y_values(x_list, 2);
y_5_terms = y_values(x_list, 5);
y_actual = actual_function(x_list);

plot(x_list, y_actual, 'k-')
hold on;
plot(x_list, y_2_terms, 'r--')
hold on;
plot(x_list, y_5_terms, 'g--')

legend("f(x) = ln(3x+1)", "Taylor Series: 2 Terms", "Taylor Series: 5 Terms")
xlabel("x")
ylabel("y")
hold off
```



## Part 2

5 terms are going to have a less error than 2 terms because for each additional term, we get closer to matching the function completely

```
error_y_2_terms = y_2_terms - y_actual;
error_y_5_terms = y_5_terms - y_actual;

L2_norm_2_terms = L2_norm(error_y_2_terms)
L2_norm_5_terms = L2_norm(error_y_5_terms)

figure(2)
plot(x_list, abs(error_y_2_terms), 'r--')
hold on;
plot(x_list, abs(error_y_5_terms), 'g--')
legend("Error Vector Values: 2 Terms", "Error Vector Values: 5 Terms")
xlabel("x")
ylabel("Error Value")
hold off;
```

## Functions Used for Part 1

```
% Step 1: Take each x (x_list(element) and number of terms needed (terms)
% and feed into the function, get_value(...). Will eventally return with all y values
% cooresponding to that xi+h
function value_list = y_values(x_list, terms)
    value_list=zeros(1,length(x_list));
    for element=1:length(x_list)
        value_list(element) = get_value(x_list(element), terms);
    end
end
```

```
% Step 2: Feed x into the function,
% taylor_series(...). Will calculate the first 6 terms of the taylor
% series in an array. Then cuts off the array depending on how many terms
% are needed. Then sums those the terms together to get approximate f(x)
function taylor_value = get_value(x, terms)
    taylor_list = taylor_series(x);
    taylor_terms = taylor_list(1:terms);
    taylor_value = sum(taylor_terms);
end

% Step 3: Calculates taylor series for each derivative corresponding to
% f^(n)(0) and feeds to taylor_poly for rest of calculation
function taylor_list = taylor_series(x)
    f0 = taylor_poly(log(3*0 + 1),0, x);
    f1 = taylor_poly(3*((3*0 +1)^(-1)),1, x);
    f2 = taylor_poly(-9*((3*0 +1)^(-2)),2, x);
    f3 = taylor_poly(54*((3*0 +1)^(-3)),3, x);
    f4 = taylor_poly(-486*((3*0 +1)^(-4)),4, x);
    f5 = taylor_poly(5832*((3*0 +1)^(-5)),5, x);

    taylor_list = [f0, f1, f2, f3, f4, f5];
end

% Step 4: Same thing as step 3 but now also includes (x^n)/n!  (num=n)
function taylor_part = taylor_poly(equation, num, x)
    taylor_part = (1/factorial(num))*(equation*(x^(num)));
end

% Step 5: Compares to ln((3*x)+1)
function real_y = actual_function(x_list)
    real_y = zeros(1,length(x_list));
    for n=1:length(x_list)
        real_y(n) = log(3*(x_list(n)) + 1);
    end
end
```

## Functions used for Part 2

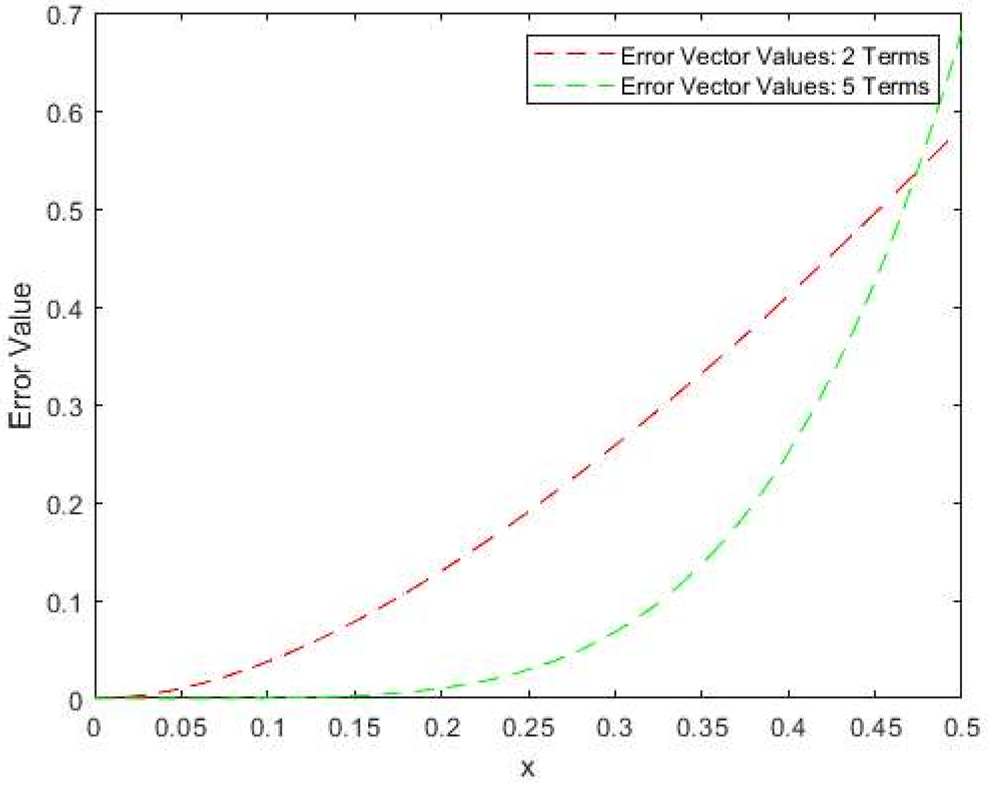
```
function L2_error = L2_norm(y_terms)
    L2_error = dot(y_terms, y_terms)^(1/2);
end
```

```
L2_norm_2_terms =

    6.3944


L2_norm_5_terms =

    4.8563
```



*Published with MATLAB® R2022a*