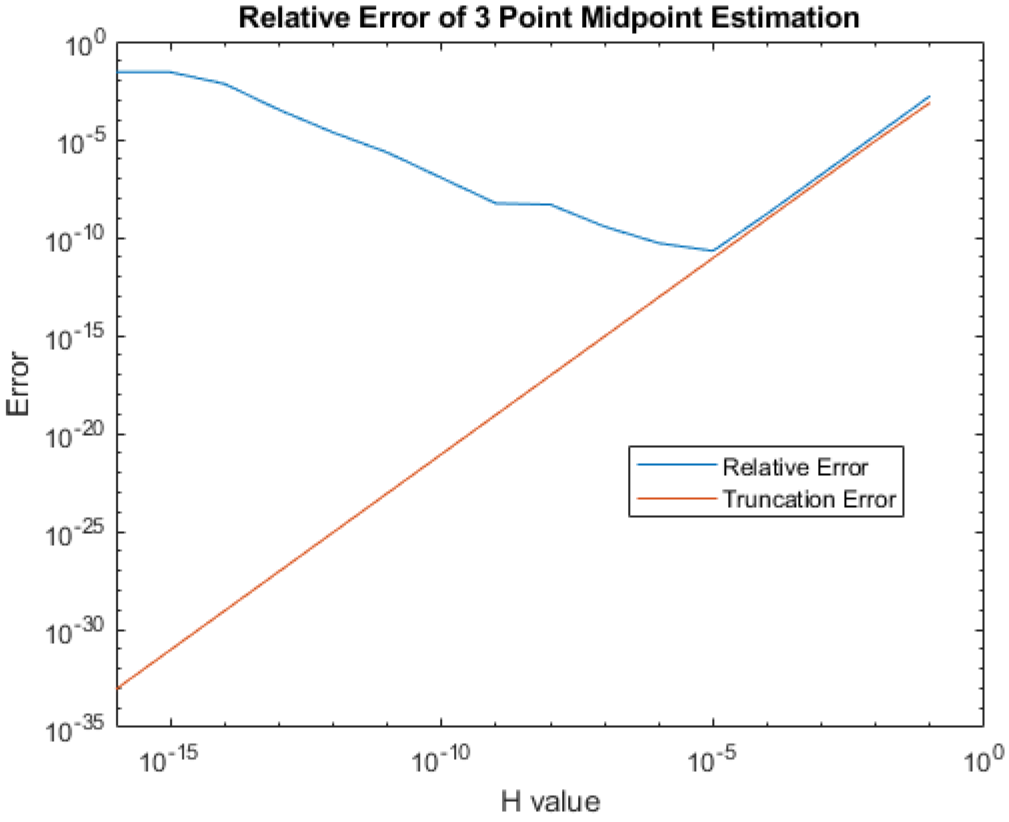
Table of Contents

Question 5

```
%Emmanuel Vargas
%3/12/21
```

Part A

```
%x value to find derivative at
x = 1;

%Create function handle for sin(x) and its derivatives
f = @(x)(sin(x));
fp = @(x)(cos(x)); %first derivative
fp2 = @(x)(-1 * sin(x)); %second derivative
fp3 = @(x)(-1 * cos(x)); %third derivative
fp4 = @(x)(sin(x)); %fourth derivative

%Create vector for n
n = [1 : 1 : 16];

%vector of h values to be tested
h = 10 .^ (-1 .* n);

%vectors to store relative and truncation error at relative h
relErr = zeros(size(h));
truncErr = zeros(size(h));

%actual derivative at x
trueDeriv = fp(x);

%iterate over all h values and calculate the relative error for
%the derivative at point x
for i = 1 : length(h)
    %find points to calculate the midpoint
    a = x - h(i);
    b = x + h(i);

    %calculate estimate using 3 point midpoint formula using current h
    estimate = (1 / (2 * h(i))) * ( f(b) - f(a) );

    %calculate the relative error of the current estimate, store it
    relErr(i) = abs(abs(estimate - trueDeriv) / trueDeriv);
    %find the point at which f'''(x) is max on [x-h, x+h]
    %simply check endpoints, as at x = 1 f'''(x) is nowhere near a
 local
    %maximum, no need to check 0's of 4th derivative
    if fp3(a) >= fp3(b)
        maxNum = fp3(a);
    else
        maxNum = fp3(b);
    end

    %calculate truncation error of current estimate
    truncErr(i) = abs(( (h(i)^2) / 6 ) * maxNum);
end

%plot the relative error against values of h
loglog(h, relErr);
hold on;
loglog(h, truncErr);
xlabel('H value');
ylabel('Error');
legend('Relative Error', 'Truncation Error', 'location', 'best');
title('Relative Error of 3 Point Midpoint Estimation');
```



Part B

```
disp('Based on the graph generated above, the absolute error does
 decrease as h gets smaller. However, after h gets too small the
 absolute error starts to increase again.')
disp('Therefore, the relative error does not follow the truncation
 estimate. I believe this is due to the digital limits of doing
 arithmetic calculations with such small numbers.')
disp('Due to the nature of floating point number implementations,
 there are a limited amount of numbers to represent all decimals.
 Thus, as h gets way too small, there are less and less nearby
 floating point numbers to represent the real number.')
disp('This means that every single time that the estimate gets smaller
 more and more error is introduced as there are no nearby floating
 point numbers to represent the actual number at each step. This is
 why the error increased quite dramatically as the numbers got smaller
 and smaller.')
```

```
Based on the graph generated above, the absolute error does decrease
 as h gets smaller. However, after h gets too small the absolute error
 starts to increase again.
Therefore, the relative error does not follow the truncation estimate.
 I believe this is due to the digital limits of doing arithmetic
 calculations with such small numbers.
Due to the nature of floating point number implementations, there are
 a limited amount of numbers to represent all decimals. Thus, as h
 gets way too small, there are less and less nearby floating point
 numbers to represent the real number.
This means that every single time that the estimate gets smaller more
 and more error is introduced as there are no nearby floating point
 numbers to represent the actual number at each step. This is why the
 error increased quite dramatically as the numbers got smaller and
 smaller.
```

Published with MATLAB® R2020a