```
%%define values used to approximate derivative
x = 1;
h = [0.1    0.05    0.025   0.0125  0.00625 0.003125];

%%mathematically calculate exact value of second derivative at x
y2 = second_derivative(x);

%%approximate second derivative and find order using forward approximation
%Error reduces linearly, and error/h remains the same as h is reduced,
%showing that this method has first order convergence.
forward = forward_approx(@f,x,h);
forward_error = abs(forward - y2);
forward_error_h = forward_error ./ h;
forward_error_h2 = forward_error_h ./ h;
forward_error_order = error_order(@f,@forward_approx,@second_derivative,x,h);
T1 = table(h', forward', forward_error', forward_error_h', forward_error_h2', forward_error_order');
T1.Properties.VariableNames = {'h', 'forward_approximation', 'error', ...
    'error_h', 'error_h2', 'order'};
T1

%%approximate second derivative and find order using centered approximation
%Error reduces quadratically, and error/h reduces linearly, and error/h^2
%remains relatively constant. This is evidence of second order convergence.
centered = centered_approx(@f,x,h);
centered_error = abs(centered - y2);
centered_error_h = centered_error ./ h;
centered_error_h2 = centered_error_h ./ h;
centered_error_order = error_order(@f,@centered_approx,@second_derivative,x,h);
T2 = table(h', centered', centered_error', centered_error_h', centered_error_h2', centered_error_order');
T2.Properties.VariableNames = {'h', 'centered_approximation', 'error', ...
    'error_h', 'error_h2', 'order'};
T2

%%redefine h with step sizes 1/2^j for j = 0:64
for index = 1:65
    h(index) = 1 / 2 ^ (index - 1);
end

%%reapproximate second derivative and recalculate error using new h values
forward = forward_approx(@f,x,h);
forward_error = abs(forward - y2);
centered = centered_approx(@f,x,h);
centered_error = abs(centered - y2);

%%plots error with logarithmic scales for axes
%At a sufficiently small value of h, the error caused by truncating the
%Taylor series becomes insignificant compared to the system's round off
%error. For the forward difference method, this occurs about when h reaches
%10^-5 and for the centered difference method, this occurs about when h
%reaches 10^-4. This shift in error dominance occurs more quickly for the
%centered difference method because it has a higher order of convergence
%and the value of h does not need to be as small to achieve the same degree
%of accuracy as forward difference approximation.
%When the round off error does become dominant, it seems to erratically
```

```
%measure as zero at some points. My best estimation for this is because the
%computer system used to measure round off error of the methods also
%suffers from round off error itself. Occasionally, the round off errors of
%the system and the approximation will match. From the perspective of
%the computer's logic, it will believe that there is zero error when this
%occurs.
loglog(h,forward_error,'-o');
xlabel('h');
ylabel('error');
hold on
loglog(h,centered_error,'-o');
title('Error of second derivative approximation as h varies');
legend('forward approximation', 'centered approximation');



%%define cos(x) as function of choice
function y = f(x)
    y = cos(x);
end

%%define -cos(x) as second derivative of function
function f2 = second_derivative(x)
    f2 = -cos(x);
end

%%implementation of forward difference approximation
function f2 = forward_approx(f,x,h)
    f2 = [];
    for delta = h
        approximation = f(x + 2 * delta) - 2 * f(x + delta) + f(x);
        approximation = approximation / delta .^ 2;
        f2 = [f2 approximation];
    end
end

%%implementation of centered difference approximation
function f2 = centered_approx(f,x,h)
    f2 = [];
    for delta = h
        approximation = f(x + delta) - 2 * f(x) + f(x - delta);
        approximation = approximation / delta .^ 2;
        f2 = [f2 approximation];
    end
end

%%measures error of a given method for a given h value
function order = error_order(func,approximation,second_derivative,x,h)
    order = [];
    for delta = h
        error_h = abs(approximation(func,x,delta) - second_derivative(x));
        error_2h = abs(approximation(func,x,2*delta) - second_derivative(x));
        order = [order log2(error_2h / error_h)];
    end
end
```

T1 =

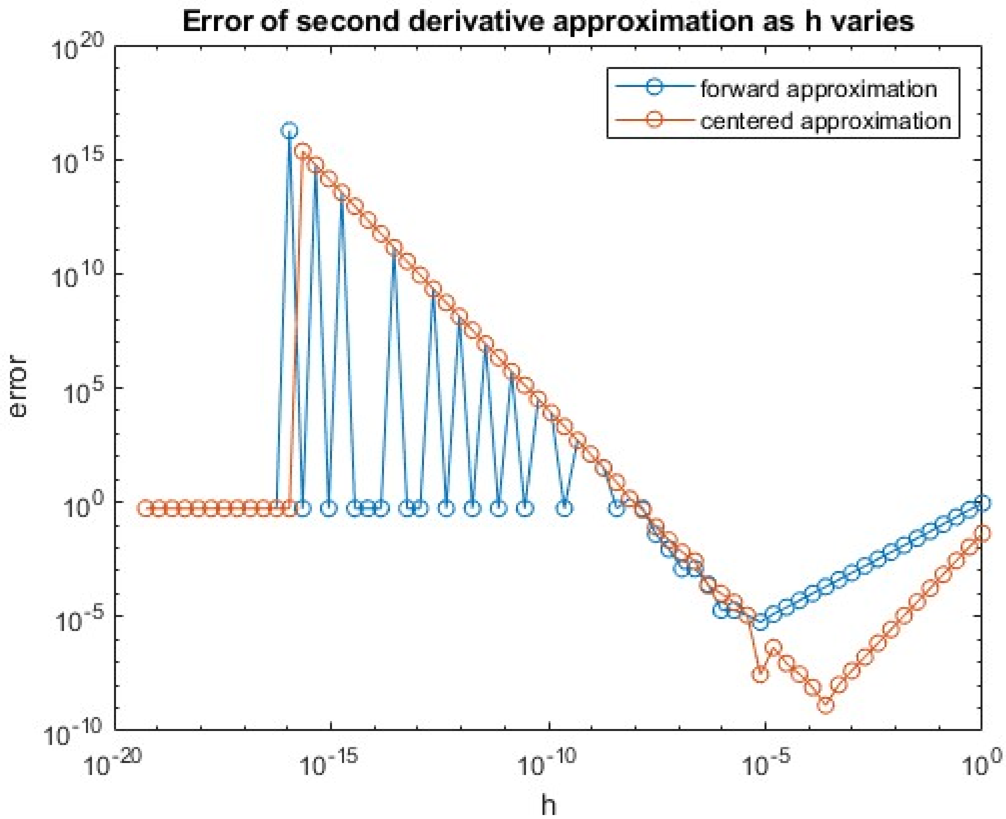
6×6 table

h	forward_approximation	error	error_h	error_h2	order
0.1	-0.45322	0.087084	0.87084	8.7084	1.0407
0.05	-0.49747	0.042835	0.8567	17.134	1.0236
0.025	-0.51907	0.02123	0.84922	33.969	1.0127
0.0125	-0.52974	0.010567	0.84538	67.63	1.0065
0.00625	-0.53503	0.0052715	0.84343	134.95	1.0033
0.003125	-0.53767	0.0026327	0.84245	269.59	1.0017

T2 =

6×6 table

h	centered_approximation	error	error_h	error_h2	order
0.1	-0.53985	0.0004501	0.004501	0.04501	1.9986
0.05	-0.54019	0.00011255	0.0022511	0.045021	1.9996
0.025	-0.54027	2.814e-05	0.0011256	0.045024	1.9999
0.0125	-0.5403	7.0351e-06	0.00056281	0.045025	2
0.00625	-0.5403	1.7588e-06	0.00028141	0.045025	2
0.003125	-0.5403	4.397e-07	0.0001407	0.045025	2



Published with MATLAB® R2019a