```
%Define our mathematical functions
f0 = @(x) x.^3;
f1 = @(x) sin(0.5*x);
f2 = @(x) abs(sin(2*x));
f3 = @(x) cos(x);

%Define our intervals
I1 = [0, pi/3];
I2 = [0, 2*pi];

%Store the actual integral's values in a vector, where index 1 is for interval I1, and
index 2 is for interval I2
truef1 = [2-sqrt(3), 4];    %0.2679491924311227, 4
truef2 = [3/4, 4];          %.25, 4
truef3 = [sqrt(3)/2, 0];     %0.8660254037844386, 0

%Compute approximation of functions
y0 = trapezoidrule(f0, 0, 1, 100); %Test case with fixed a, b, N
outputf1 = approximateIntegral(f1, I1, I2, 1000);
outputf2 = approximateIntegral(f2, I1, I2, 1000);
outputf3 = approximateIntegral(f3, I1, I2, 1000);

%Compute the absolute errors of the functions
errorf1 = absoluteError(outputf1, truef1);
errorf2 = absoluteError(outputf2, truef2);
errorf3 = absoluteError(outputf3, truef3);

%Output approximations to the console
disp("Approximations")
fprintf("  Test case: %.12f\n",y0);   %Test case
fprintf("  f1(x):\n     I1: %.12f\n     I2: %.12f\n",outputf1(2,end), outputf1(3,end));
%f1
fprintf("  f2(x):\n     I1: %.12f\n     I2: %.12f\n",outputf2(2,end), outputf2(3,end));
%f2
fprintf("  f3(x):\n     I1: %.12f\n     I2: %.12f\n",outputf3(2,end), outputf3(3,end));
%f3

%Output actual to the console
disp("Actual Values")
fprintf("  f1(x):\n     I1: %.12f\n     I2: %.12f\n",truef1(1), truef1(2));   %f1
fprintf("  f2(x):\n     I1: %.12f\n     I2: %.12f\n",truef2(1), truef2(2));   %f2
fprintf( "f3(x):\n     I1: %.12f\n     I2: %.12f\n",truef3(1), truef3(2));   %f3

%Output errors to the console
disp("Absolute erros")
fprintf("  f1(x):\n     I1: %.12f\n     I2: %.12f\n",errorf1(1,end), errorf1(2,end));   %
f1
fprintf("  f2(x):\n     I1: %.12f\n     I2: %.12f\n",errorf2(1,end), errorf2(2,end));   %
f2
fprintf( "f3(x):\n     I1: %.12f\n     I2: %.12f\n",errorf3(1,end), errorf3(2,end));   %
f3
```

```
%Plot the absolute error of our approximations vs N
close all;
hold on;
grid on;
axis on;
ylabel("log(|error|)");
xlabel("log(N)");
title("Loglog plot of absolute error vs N for interval I1");
plot(log100(outputf1(1,:)), log100(errorf1(1,:)), "r*",'DisplayName', 'f1(x) = sin(0.5↙
x)');
plot(log100(outputf2(1,:)), log100(errorf2(1,:)), "m*",'DisplayName', 'f2(x) = |sin(2x)↙
|');
plot(log100(outputf3(1,:)), log100(errorf3(1,:)), "b*",'DisplayName', 'f3(x) = cos(x)');
legend;
hold off;
figure;

hold on;
grid on;
axis on;
ylabel("log(|error|)");
xlabel("log(N)");
title("Loglog plot of absolute error vs N for interval I2");
plot(log100(outputf1(1,:)), log100(errorf1(2,:)), "r*",'DisplayName', 'f1(x) = sin(0.5↙
x)');
plot(log100(outputf2(1,:)), log100(errorf2(2,:)), "m*",'DisplayName', 'f2(x) = |sin(2x)↙
|');
plot(log100(outputf3(1,:)), log100(errorf3(2,:)), "b*",'DisplayName', 'f3(x) = cos(x)');
legend;
hold off;

%Find the rate of convergence for each integral, on each interval
pFits = zeros(6,2);
pFits(1,:) = polyfit(log100(outputf1(1,:)), log100(errorf1(1,:)), 1);
pFits(2,:) = polyfit(log100(outputf1(1,:)), log100(errorf1(2,:)), 1);
pFits(3,:) = polyfit(log100(outputf1(1,:)), log100(errorf2(1,:)), 1);
pFits(4,:) = polyfit(log100(outputf1(1,:)), log100(errorf2(2,:)), 1);
pFits(5,:) = polyfit(log100(outputf1(1,:)), log100(errorf3(1,:)), 1);
pFits(6,:) = polyfit(log100(outputf1(1,:)), log100(errorf3(2,:)), 1);
orders = -1*pFits(:,1);     %multiply by -1 as this is order of growth, and we want order↙
of convergence

%Display the rate of convergence for each each error approaching 0
disp("Rate of convergence for errors")
fprintf("  f1(x):\n     I1: %.12f\n     I2: %.12f\n",orders(1), orders(2));   %f1
fprintf("  f2(x):\n     I1: %.12f\n     I2: %.12f\n",orders(3), orders(4));   %f2
fprintf("  f3(x):\n     I1: %.12f\n     I2: %.12f\n",orders(5), orders(6));   %f3

%Define out computational functions
function output = approximateIntegral(f, I1, I2, Nmax)
```

```
    output = zeros(1, Nmax/100);
    for N=100:100:Nmax
        output(1,N/100) = N;                                              %[N0    , N1
, ... ,Nj    ]
        output(2,N/100) = trapezoidrule(f, I1(1), I1(2), N);        %output = [g(N0), g
(N1), ... ,g(Nj)] for I1, where g(x) approximates integral f(x)
        output(3,N/100) = trapezoidrule(f, I2(1), I2(2), N);                 %[g(N0), g
(N1), ... ,g(Nj)] for I2
    end
end

function output = absoluteError(approx, actual)  %approx is expected to be in the format
of output from approximateIntegral
    output = zeros(2, length(approx(1,:)));
    for i=1:length(actual)                                          %output = [|f(x)-g1
(x)|, |f(x)-g2(x)|, ... , |f(x)-gN(x)|] for I1
        for j=1:length(approx(1,:))                                          %[|f(x)-g1
(x)|, |f(x)-g2(x)|, ... , |f(x)-gN(x)|] for I2
            output(i,j) = abs(actual(i) - approx(i+1,j));           %Where gj(x)
approximates the integral f(x) with j subdivisions
        end
    end
end

%Define log100(x) since we increment by 100, implying h = 100
function y =  log100(x)
    y = log(x)/log(100);
end
```