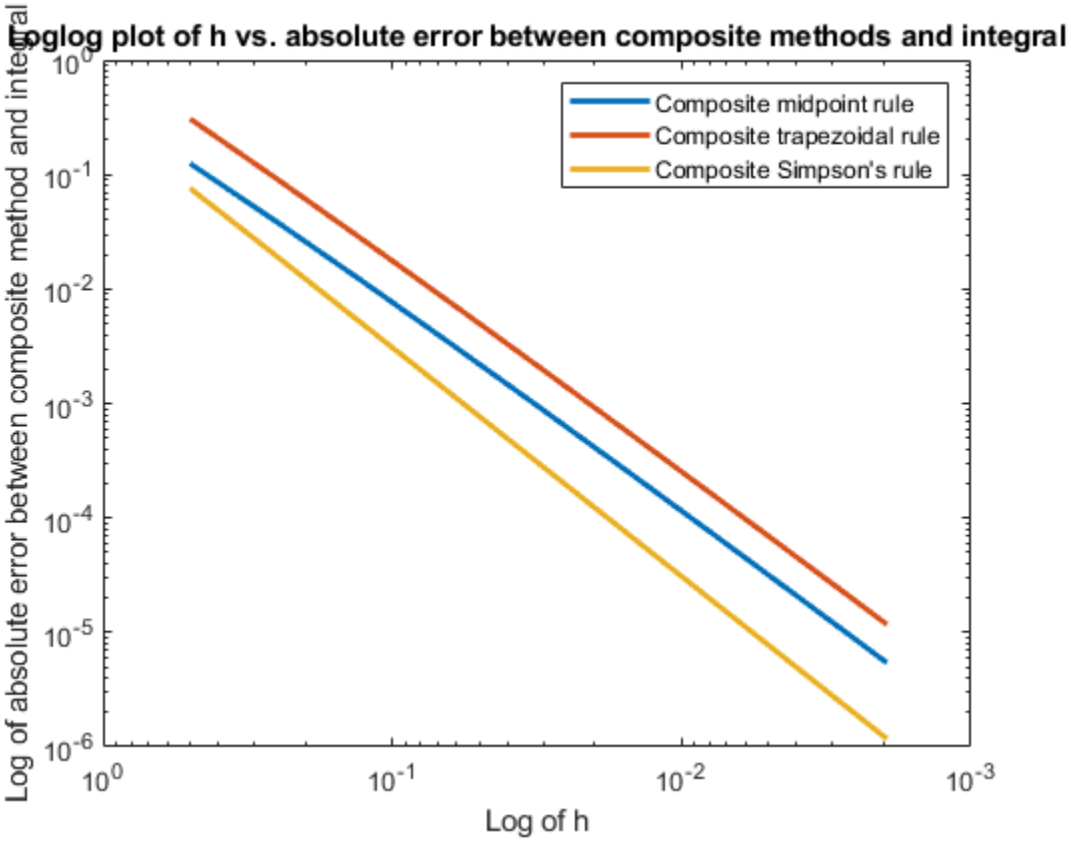
```
clc;
clear;

a = -5;
b = 5;
k = 25;
n = 2*k;
f = @(x) 1./(1+x.^(2));


% Problem 1c
tol1 = 1e-6;
tol2 = 1e-4;

n_trap = 1291;
n_simp = 108;

val_comp_trap = comp_trap(a, b, f, n_trap);
val_comp_simp = comp_simp(a, b, f, n_simp);

% Run quad with a tolerance of 1e-6
[val_integral1, fcnt1]  = quad(f, a, b, tol1);
fprintf("Absolute error between integral (tol = %e) and composite trapezoidal
 rule (n = %d): %e\n", ...
        tol1, n_trap, abs(val_integral1-val_comp_trap));
fprintf("Absolute error between integral (tol = %e) and composite Simpson's
 rule (n = %d): %e\n", ...
        tol1, n_simp, abs(val_integral1-val_comp_simp));
fprintf("Number of function evaluations needed for integral (tol = %e): %d
\n", ...
        tol1, fcnt1)

% Run quad with a tolerance of 1e-4
[val_integral2, fcnt2] = quad(f, a, b, tol2);
fprintf("\n");
fprintf("Absolute error between integral (tol = %e) and composite trapezoidal
 rule (n = %d): %e\n", ...
        tol2, n_trap, abs(val_integral2-val_comp_trap));
fprintf("Absolute error between integral (tol = %e) and composite Simpson's
 rule (n = %d): %e\n", ...
        tol2, n_simp, abs(val_integral2-val_comp_simp));
fprintf("Number of function evaluations needed for integral (tol = %e): %d
\n", ...
        tol2, fcnt2)
```

Problem 2

```
clear;
g = @(x) x.*log(x);
a_g = 1e-16;
b_g = 1;
```

```
n_vals = 2.^(1:9)';
h = (b_g-a_g)./(n_vals);

% Actual value of integral
val_integral_g = 1;

% Arrays to store the absolute value of the errors
abs_errs = zeros(length(n_vals), 3);

for i = 1:length(n_vals)
    val_comp_mid_g = -4*comp_mid(a_g, b_g, g, n_vals(i));
    val_comp_trap_g = -4*comp_trap(a_g, b_g, g, n_vals(i));
    val_comp_simp_g = -4*comp_simp(a_g, b_g, g, n_vals(i));

    abs_errs(i, 1) = abs(val_comp_mid_g - val_integral_g);
    abs_errs(i, 2) = abs(val_comp_trap_g - val_integral_g);
    abs_errs(i, 3) = abs(val_comp_simp_g - val_integral_g);
end

figure;
loglog(h, abs_errs, "LineWidth", 2);
legend("Composite midpoint rule", "Composite trapezoidal rule", "Composite
 Simpson's rule");
xlabel("Log of h");
ylabel("Log of absolute error between composite method and integral");
title("Loglog plot of h vs. absolute error between composite methods and
 integral")
set(gca, "xdir", "reverse");

function val = comp_mid(a, b, f, n)
    nodes = linspace(a, b, n+1)';
    h = (b-a)/n;
    nodes = 0.5*(nodes(1:end-1) + nodes(2:end));
    val = h*sum(f(nodes));
end

function val = comp_trap(a, b, f, n)
    nodes = linspace(a, b, n+1)';
    h = (b-a)/n;
    f_nodes = f(nodes);
    f_nodes(1) = 0.5*f_nodes(1);
    f_nodes(end) = 0.5*f_nodes(end);
    val = h*sum(f_nodes);
end

function val = comp_simp(a, b, f, n)
    nodes = linspace(a, b, n+1)';
    h = (b-a)/n;
    f_nodes = f(nodes);
    f_nodes(2:2:n) = 4*f_nodes(2:2:n);
    f_nodes(3:2:n-1) = 2*f_nodes(3:2:n-1);
    val = (h/3)*sum(f_nodes);
end
```

```
Absolute error between integral (tol = 1.000000e-06) and composite trapezoidal
 rule (n = 1291): 1.077025e-07
Absolute error between integral (tol = 1.000000e-06) and composite Simpson's
 rule (n = 108): 2.504883e-07
Number of function evaluations needed for integral (tol = 1.000000e-06): 81

Absolute error between integral (tol = 1.000000e-04) and composite trapezoidal
 rule (n = 1291): 6.295931e-06
Absolute error between integral (tol = 1.000000e-04) and composite Simpson's
 rule (n = 108): 6.438717e-06
Number of function evaluations needed for integral (tol = 1.000000e-04): 41
```



Published with MATLAB® R2021b