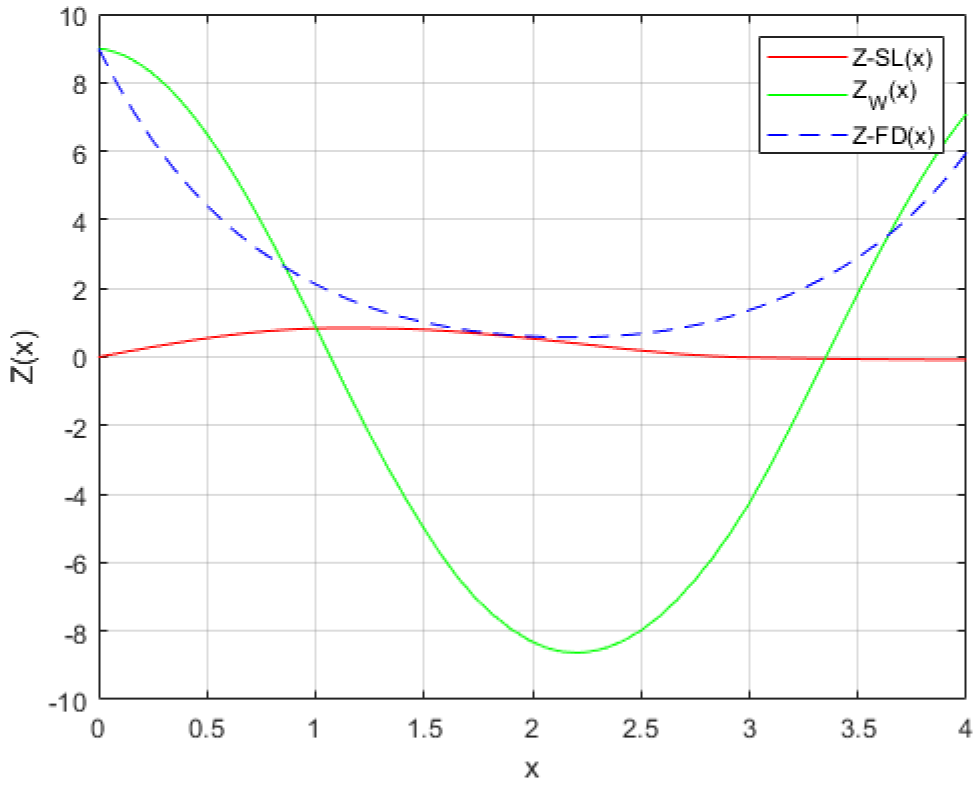
```
% Parameters
N_max = 50; % Maximum number of terms in the series
w = 1; % Weight function
x_values = linspace(0, 4, 100); % x values to evaluate the solution at
alpha_squared = 2;
A = 9;
B = 8;

% Q(x) function
Q = @(x) (0.*(x>=0 & x<1) + ...
          abs(sin(6.*(x-1))).*(x>=1 & x<2) + ...
          (sqrt(x-2) - sin(6)).*(x>=2 & x<3) + ...
          0.*(x>=3 & x<=4));

% Eigenfunctions yn
yn = @(n, x) sin(((2*n - 1) * pi) / 8 * x);

% Q_hat, N_hat, Z_hat functions with Simpson's rule
dx = x_values(2) - x_values(1);
Q_hat = @(n) simpson(arrayfun(Q, x_values) .* arrayfun(@(x) yn(n, x),
 x_values), dx);
N_hat = @(n) simpson((arrayfun(@(x) yn(n, x), x_values)).^2 * w, dx);

Z_hat = @(n) Q_hat(n) / (2 - ((2*n - 1) * pi / 8)^2);

% Sturm-Liouville solution Z(x)
Z_SL = @(x, N_max) sum(arrayfun(@(n) Z_hat(n) .* yn(n, x) / N_hat(n),
 1:N_max));

% Apply the Sturm-Liouville solution to each x value
Z_SL_values = arrayfun(@(x) Z_SL(x, N_max), x_values);

x_values_fd = linspace(0, 4, N_max);
dx_fd = 4 / (N_max - 1);
Q_fd = arrayfun(Q, x_values_fd) * dx_fd^2;
Q_fd(1) = A;
Q_fd(end) = B * dx_fd;

% Coefficient matrix
M = zeros(N_max, N_max);
for i = 2:N_max-1
    M(i, i-1) = 1;
    M(i, i) = -2 - (alpha_squared * dx_fd^2);
    M(i, i+1) = 1;
end
M(1, 1) = 1;
M(end, end-1) = -1;
M(end, end) = 1;

Z_fd_corrected = M \ Q_fd';

y1 = @(x) cos(sqrt(alpha_squared) * x);
```

```
y2 = @(x) sin(sqrt(alpha_squared) * x);
wronskian = @(f, g, x) (f(x + h) - f(x - h))/(2*h) * g(x) - f(x) * (g(x + h) -
 g(x - h))/(2*h);

h = 1e-5; % Step size for derivative
W = arrayfun(@(x) wronskian(y1, y2, x), x_values);

% Compute integrals for u1 and u2 numerically using Simpson's rule
u1 = zeros(size(x_values));
u2 = zeros(size(x_values));
for i = 2:length(x_values)
    xi = x_values(1:i);
    u1(i) = simpson(arrayfun(@(x) Q(x) * y2(x) / wronskian(y1, y2, x), xi),
 dx);
    u2(i) = simpson(arrayfun(@(x) Q(x) * y1(x) / wronskian(y1, y2, x), xi),
 dx);
end

% Particular solution Zp
Zp = u1 .* arrayfun(y1, x_values) + u2 .* arrayfun(y2, x_values);

% Solve for c1 and c2 using boundary conditions
coeff = [y1(0), y2(0); derivative(y1, 4), derivative(y2, 4)];
rhs = [A - Zp(1), B - derivative(@(x) y2(x), 4)];
c = coeff \ rhs';

% Full solution Z_W
Z_W = c(1) * arrayfun(y1, x_values) + c(2) * arrayfun(y2, x_values) + Zp;

% Plot the results
plot(x_values, Z_SL_values, 'r', 'DisplayName', 'Z-SL(x)');
hold on;
plot(x_values, Z_W, 'g', 'DisplayName', 'Z_W(x)');
plot(x_values_fd, Z_fd_corrected, 'b--', 'DisplayName', 'Z-FD(x)');
xlabel('x');
ylabel('Z(x)');
legend;
grid on;


function result = derivative(f, x)
    h = 1e-5; % Step size for derivative
    result = (f(x + h) - f(x - h)) / (2 * h);
end

% Simpson's rule integrator
function result = simpson(Q, dx)
    result = dx/3 * (Q(1) + Q(end) + 4*sum(Q(2:2:end-1)) +
 2*sum(Q(3:2:end-2)));
end
```



Published with MATLAB® R2023a