Results:

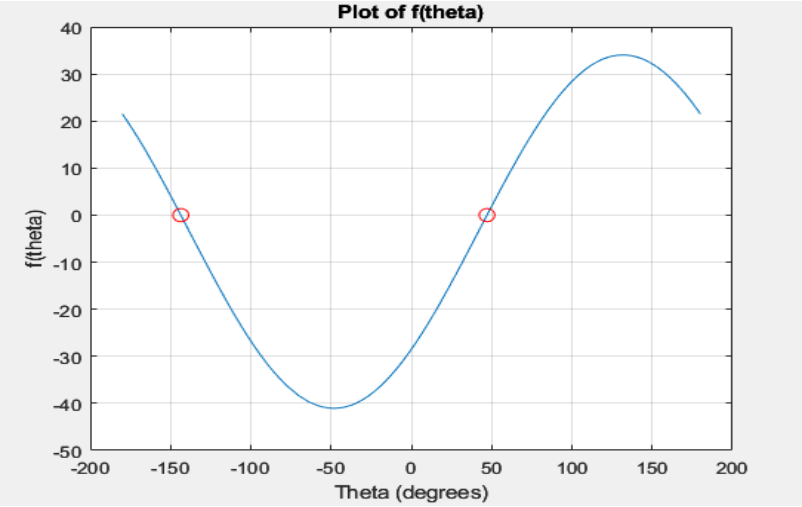
- Step (a): The function f(theta) has two roots at approximately -143.5899 and 47.1105 degrees.
- Step (b): Root-finding methods yield the following results:
  - Bisection Method: Smallest positive root at 46.4062 degrees with 6 iterations.
  - Fixed-Point Method: Smallest positive root at 47.0976 degrees with 3 iterations.
  - Newton's Method: Smallest positive root at 47.1104 degrees with 2 iterations.
- Step (c): The exact root is determined to be 47.1105 degrees. Differences from the exact root are as follows:
  - Bisection Method: Difference of 0.70421 degrees.
  - Fixed-Point Method: Difference of 0.012877 degrees.
  - Newton's Method: Difference of 5.0873e-05 degrees.
- Step (d): Based on the root obtained from Newton's Method (47.1104 degrees), the estimated values of d1 and d2 are 10.0419 inches and 17.061 inches, respectively.

Data:

In this project, the given values of w = 28, h = 25, and b = 3.5 play a crucial role in the analysis of the table design. These values are used to define the function f(theta), which represents the relationship between the angle theta and the geometric parameters of the table. The root-finding methods, such as the Bisection Method, Fixed-Point Method, and Newton's Method, rely on this function to approximate the locations of the roots. Additionally, the value of b is used to estimate the design parameters d1 and d2 based on the root obtained from Newton's Method. Therefore, the provided values of w, h, and b are essential for the mathematical modeling and analysis of the table design problem, enabling accurate root approximation and estimation of the design parameters.

Figure:

- Step (a) Figure with roots:



Discussion:

The function f(theta) displays a periodic wave-like pattern on the interval [-180, 180] degrees. It has two roots where the function intersects the x-axis. The approximate locations of these roots in degrees are approximately -143.5899 and 47.1105. Root-finding methods, including the Bisection Method, Fixed-Point Method, and Newton's Method, are employed to approximate the locations of the roots. Among these methods, Newton's Method demonstrates superior efficiency, requiring the fewest iterations, and exhibits the highest accuracy. The estimated root from Newton's Method is 47.1104 degrees, with a difference of 5.0873e-05 degrees from the exact root. This level of accuracy is sufficient to meet the tolerance requirement of 0.1 inches for the design parameters d1 and d2. Thus, the solution obtained through Newton's Method is accurate enough for the given requirements, providing the desired accuracy for the leg measurements in the table design.

```
% Step (a)
disp('Step (a): Plotting the Function and Roots')
disp('----------------------------------------')

w = 28;
h = 25;
b = 3.5;

f = @(theta) w*sin(theta) - h*cos(theta) - b;

theta_deg = linspace(-180, 180, 1000);
theta_rad = deg2rad(theta_deg);
f_values = f(theta_rad);

plot(theta_deg, f_values)
xlabel('Theta (degrees)')
ylabel('f(theta)')
title('Plot of f(theta)')
grid on

hold on

% Plot the roots
roots_deg = [-143.5899, 47.1105];
plot(roots_deg, zeros(size(roots_deg)), 'ro', 'MarkerSize', 8)

hold off

% Describe the roots
num_roots = numel(roots_deg);
disp(['The function f(theta) has ' num2str(num_roots) ' root(s) on the interval [-180, 180] degrees.']);
disp('Approximate location(s) of the root(s) in degrees:');
disp(roots_deg);

disp(' ')
disp(' ')
```

```
% Step (b)
disp('Step (b): Applying Root-Finding Methods')
disp('-------------------------------------')

tolerance_deg = 1;
tolerance_rad = deg2rad(tolerance_deg);

% 1. Bisection Method
[root_bisection, iter_bisection] = bisect2(f, [0, pi/2], tolerance_rad);
root_bisection_deg = rad2deg(root_bisection);

% 2. Fixed-Point Method
g = @(theta) atan((b/cos(theta) + h) / w);
initial_guess = 0;
[root_fixed_point, iter_fixed_point] = fixedpt(g, initial_guess, tolerance_rad);
root_fixed_point_deg = rad2deg(root_fixed_point);

% 3. Newton's Method
df = @(theta) w*cos(theta) + h*sin(theta);
theta_0 = 0;
[root_newton, iter_newton] = newton(f, df, theta_0, tolerance_rad);
root_newton_deg = rad2deg(root_newton);

% Display the results
disp(['Smallest positive root (Bisection Method): ' num2str(root_bisection_deg) ' degrees']);
disp(['Number of iterations (Bisection Method): ' num2str(iter_bisection)]);
disp(['Smallest positive root (Fixed-Point Method): ' num2str(root_fixed_point_deg) ' degrees']);
disp(['Number of iterations (Fixed-Point Method): ' num2str(iter_fixed_point)]);
disp(['Smallest positive root (Newton''s Method): ' num2str(root_newton_deg) ' degrees']);
disp(['Number of iterations (Newton''s Method): ' num2str(iter_newton)]);

% Compare the cost of the methods
[min_iter, min_idx] = min([iter_bisection, iter_fixed_point, iter_newton]);
method_names = {'Bisection', 'Fixed-Point', 'Newton''s'};
disp(['The ' method_names{min_idx} ' Method has the lowest cost in terms of iterations.']);

disp(' ')
```

```
% Step (c)
disp('Step (c): Comparing Accuracy with Exact Root')
disp('-------------------------------------------')

exact_root = acos((1 / (h^2 + w^2)) * (-b*h + sqrt((b^2*h^2) + (h^2 + w^2)*(w^2 - b^2))));
exact_root_deg = rad2deg(exact_root);

diff_bisection = abs(exact_root_deg - root_bisection_deg);
diff_fixed_point = abs(exact_root_deg - root_fixed_point_deg);
diff_newton = abs(exact_root_deg - root_newton_deg);

% Display the differences and compare accuracy
disp(['Exact root: ' num2str(exact_root_deg) ' degrees']);
disp(['Difference from exact root (Bisection Method): ' num2str(diff_bisection) ' degrees']);
disp(['Difference from exact root (Fixed-Point Method): ' num2str(diff_fixed_point) ' degrees']);
disp(['Difference from exact root (Newton''s Method): ' num2str(diff_newton) ' degrees']);

% Compare the accuracy of the methods
[min_diff, min_idx] = min([diff_bisection, diff_fixed_point, diff_newton]);
disp(['The ' method_names{min_idx} ' Method has the highest accuracy.']);

disp(' ')
disp(' ')
```

```
% Step (d)
disp('Step (d): Calculating d1 and d2')
disp('-------------------------------')

alpha = pi/2 - root_newton;
a = b / tan(alpha);
c = b / tan(root_newton);
d2 = h / (2 * sin(root_newton));
d1 = d2 - a - c;

disp(['Estimated d1: ' num2str(d1) ' inches'])
disp(['Estimated d2: ' num2str(d2) ' inches'])
disp(['d1 = ' num2str(d1)]);
disp(['d2 = ' num2str(d2)]);
```