The Problem

Rectangle ABCD is inscribed in a circle of radius R, find the largest possible area of the rectangle.

This code solves the problem using an optimization method called the "golden section method." It is a method for minimizing a one-dimensional function iteratively by dividing the interval containing the function's minimum in the golden ratio (phi ≈ 1.618) until a specified convergence criterion (tolerance) or maximum number of iterations is reached.

The provided solutions uses my own implementation of built-in fminbnd matlab function.

Expected Output

Area = $2R^2$

For R = 5, rectangle area equals 50. As we can see, the rectangle is always a square with equal sides.

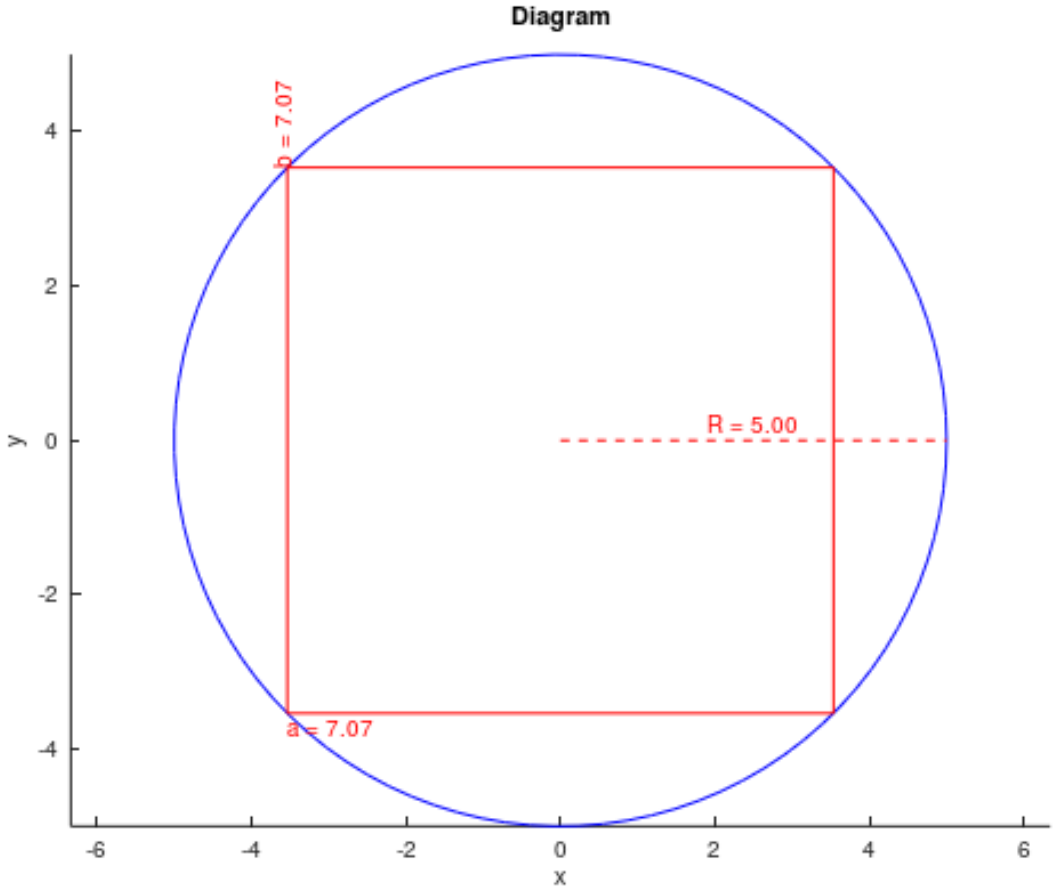
Diagram
b = 7.07
R = 5.00
a = 7.07
y
x
4
2
0
-2
-4
-6
-4
-2
0
2
4
6

Solution

1. Initialize the variable R, define the fun function, which is the objective function, with a value of -a * sqrt(R^2 - a^2).

```
R = 5;

fun = @(a) -1 * a * sqrt(R^2 - a^2); % Objective function
lb = 0; % Lower bound of the variable 'a'
ub = R; % Upper bound of the variable 'a'
```

2. Call the calculate_minbnd function which is my implementation of

   fminbnd function

   2.1. First, the parameters are defined: tolerance for convergence and maximum number of iterations.

   2.2. Initial points are initialized:

   - x1 takes the value of the lower bound.
   - x4 takes the value of the upper bound.
   - x2 is calculated as x4 - (x4 - x1) / phi, where phi is the golden ratio (1 + sqrt(5)) / 2.
   - x3 is calculated as x1 + (x4 - x1) / phi.

```
% Initial points
x1 = lb;
x4 = ub;
x2 = x4 - (x4 - x1) / phi;
x3 = x1 + (x4 - x1) / phi;
```

   2.3. The function values at the initial points are evaluated

```
% Evaluate function values at initial points
f1 = fun(x1);
f2 = fun(x2);
f3 = fun(x3);
f4 = fun(x4);
```

2.4.The main while loop begins, which will repeat as long as the difference between x4 and x1 is greater than the tolerance, and the iteration count is less than the maximum number of iterations

```
iter = 0;
while abs(x4 - x1) > tolerance && iter < max_iter
```

2.5.Inside the while loop, a comparison is made between the function values f2 and f3

```
iter = 0;
while abs(x4 - x1) > tolerance && iter < max_iter
    if f2 < f3
        x4 = x3;
        x3 = x2;
        x2 = x4 - (x4 - x1) / phi;
        f3 = f2;
        f2 = fun(x2);
    else
        x1 = x2;
        x2 = x3;
        x3 = x1 + (x4 - x1) / phi;
        f2 = f3;
        f3 = fun(x3);
    end
    iter = iter + 1;
end
```

2.6.After the while loop finishes, the final solution is chosen

```
% Choose the final solution
if f2 < f3
    x = x2;
    fval = f2;
else
    x = x3;
    fval = f3;
end
```

2.7. The function returns found solution and the function value at the found solution.

3. Calculate the value of b as the square root of R^2 - a^2.
4. Finally, all the necessary commands are used to generate the graph

The code:

```
R = 5;

fun = @(a) -1 * a * sqrt(R^2 - a^2); % Objective function
lb = 0; % Lower bound of the variable 'a'
ub = R; % Upper bound of the variable 'a'

function [x, fval] = calculate_minbnd(fun, lb, ub)
    tol = 1e-8; % Tolerance for convergence
    max_iter = 1000; % Maximum number of iterations

    % Golden ratio
    phi = (1 + sqrt(5)) / 2;

    % Initial points
    x1 = lb;
    x4 = ub;
    x2 = x4 - (x4 - x1) / phi;
    x3 = x1 + (x4 - x1) / phi;

    % Evaluate function values at initial points
    f1 = fun(x1);
    f2 = fun(x2);
    f3 = fun(x3);
    f4 = fun(x4);

    iter = 0;
    while abs(x4 - x1) > tol && iter < max_iter
        if f2 < f3
            x4 = x3;
            x3 = x2;
            x2 = x4 - (x4 - x1) / phi;
```

```
            f3 = f2;
            f2 = fun(x2);
        else
            x1 = x2;
            x2 = x3;
            x3 = x1 + (x4 - x1) / phi;
            f2 = f3;
            f3 = fun(x3);
        end
        iter = iter + 1;
    end

    % Choose the final solution
    if f2 < f3
        x = x2;
        fval = f2;
    else
        x = x3;
        fval = f3;
    end
end

[a, fval] = calculate_minbnd(fun, lb, ub);

% Solving b
b = sqrt(R^2 - a^2);

% Calculating the area
area = 4 * a * b;
disp('The largest area: ');
disp(area);

a = R * sqrt(2);
b = R * sqrt(2);

% plotting
theta = linspace(0, 2*pi, 100);
x_circle = R * cos(theta);
y_circle = R * sin(theta);

figure;
hold on;
axis equal;
xlabel('x');
ylabel('y');
title('Diagram');

plot(x_circle, y_circle, 'b');
plot([0, R], [0, 0], 'r--', 'LineWidth', 1);

rectangle('Position', [-a/2, -b/2, a, b], 'EdgeColor', 'r', 'FaceColor', 'none');
```

```
text(R/2, 0.2, sprintf('R = %.2f', R), 'Color', 'r', 'HorizontalAlignment',
'center');
text(-a/2, -b/2-0.2, sprintf('a = %.2f', a), 'Color', 'r', 'HorizontalAlignment',
'left');
text(-a/2-0.2, b/2, sprintf('b = %.2f', b), 'Color', 'r', 'Rotation', 90,
'VerticalAlignment', 'top');
```