Robotic Welding Arm Assembly for Manufacturing Automation

Sannjay Balaji

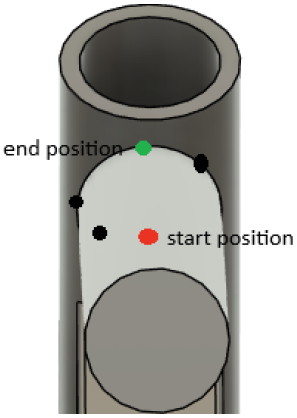
Potential Field:

The robotic manipulator begins at a specified start position (Weld Point 1) and is required to navigate to an end position (Weld Point 3) in a cluttered workspace. Weld points within the environment act as obstacles that the robotic arm must avoid. The task involves developing a path-planning algorithm that calculates attractive forces from the goal and repulsive forces from obstacles to determine an optimal collision-free path.

The MATLAB script provided simulates the path-planning process using the Potential Field Method. Below is a detailed breakdown of the code, which includes parameter definitions, force calculations, and path visualization.

Parameter Definitions

- **Attractive and Repulsive Constants**: The constants k_att and k_rep control the strength of the attractive force pulling the robotic arm towards the goal and the repulsive force pushing it away from obstacles. In this implementation, k_att is set to 5.0, and k_rep is set to 40.0, ensuring a stronger goal-directed force while keeping obstacle avoidance effective but less dominant.
- **Influence Distance (d0)**: The influence distance (d0) for the repulsive field is set at 50 millimetres, meaning that obstacles within this range will exert a repelling influence on the robotic arm.
- **Goal and Obstacle Positioning**: The goal (Weld Point 3) is located at [1330.01, 0, 190.32] millimetres, and the initial position (Weld Point 1) is [1170.01, 0, 190.32] millimetres. The obstacles, represented by weld points, are defined as points in space with coordinates near the mid-section of the manipulator's path.



#This is a pictorial representation of the start and end points; The weld will take place on the point of contact of the 2 tubes

Force Calculations

- **Attractive Force Calculation**: The attractive force (F_att) is calculated based on the distance from the current position to the goal. The attractive force pulls the robotic arm closer to the target point, and its magnitude is proportional to the distance between the current position and the goal. The attractive constant k_att is set to 5.0, which determines the strength of the force. A higher value of k_att would increase the pull towards the goal, while a lower value would reduce it.

  F_att = -k_att * (current_pos - goal);

In the equation, the negative sign indicates that the attractive force is directed towards the goal. The force is calculated as the difference between the current position and the goal, scaled by k_att.

- **Repulsive Force Calculation**: The repulsive force (F_rep) is determined by calculating the distance between the current position and the obstacles. If an obstacle is within the influence distance (d0), a repulsive force is computed to prevent collisions. The influence distance d0 is set to 50 millimeters, meaning that obstacles closer than 50 millimeters will exert a repelling influence on the robotic arm. The repulsive constant k_rep is set to 40.0, which determines the magnitude of the repulsive force. A higher value of k_rep would result in stronger obstacle avoidance, while a lower value would weaken it.

```
F_rep = [0, 0, 0];

for i = 1:size(obstacles, 1)

    d_obs = norm(current_pos - obstacles(i, :));

    if d_obs < d0

        penalty = (d0 / d_obs);

        F_rep = F_rep + penalty * k_rep * (1/d_obs - 1/d0) * (1/d_obs^2) * (current_pos - obstacles(i, :)) / d_obs;

    end

end
```

In this calculation, d_obs represents the distance between the current position and an obstacle. If this distance is less than d0, the repulsive force is calculated to push the robotic arm away from the obstacle. The term (1/d_obs - 1/d0) ensures that the force decreases as the distance to the obstacle increases, and becomes zero when the distance is greater than d0. The penalty factor (d0 / d_obs) is used to scale the repulsive force, providing a stronger repulsive effect when the obstacle is very close.

The repulsive force is accumulated for each obstacle, ensuring that the total repulsive force considers all obstacles within the influence distance.

Path Planning Algorithm

The script runs a while-loop that iteratively moves the robotic arm towards the goal. The path planning process involves several key steps:

- **Force Calculation:** At each iteration, the attractive force (F_att) and repulsive force (F_rep) are calculated based on the current position of the robotic arm, the goal position, and the obstacles. The total force (F_total) acting on the robotic arm is the sum of the attractive and repulsive forces.

```
F_total = F_att + F_rep;
```

The combined force determines the direction in which the robotic arm should move. The attractive force pulls the arm towards the goal, while the repulsive force pushes it away from obstacles.

Adaptive Step Size: An adaptive step size is used to control the movement of the robotic arm. The step size is adjusted based on the magnitude of the repulsive force,

ensuring smoother movement and better obstacle avoidance. The adaptive step size is calculated as follows:
adaptive_step_size = step_size / (1 + norm(F_rep));

The step size is divided by (1 + norm(F_rep)), which means that as the repulsive force increases, the step size decreases. This allows the robotic arm to slow down when it is close to obstacles, providing more precise movements and reducing the risk of collisions. The initial step_size is set to 30 millimeters, providing a balance between fast movement and precise control.

- **Random Perturbation:** To prevent the robotic arm from getting stuck in local minima, a small random perturbation is added to the position update. This perturbation helps the arm escape situations where the attractive and repulsive forces balance out, leading to no movement.
random_perturbation = (rand(1, 3) - 0.5) * 1.0;

The random perturbation is generated by creating a random vector with values between -0.5 and 0.5, scaled by 1.0. This introduces a small random movement, helping the robotic arm to overcome local minima and continue progressing towards the goal.

- **Position Update:** The new position of the robotic arm is calculated by applying the adaptive step size to the total force, along with the random perturbation.

current_pos = current_pos + adaptive_step_size * F_total / norm(F_total) + random_perturbation;

The total force is normalized to ensure that the movement is in the correct direction, and the adaptive step size controls the distance moved. The random perturbation is added to introduce variability, enhancing the robustness of the path planning.

- **Iteration Control:** The while-loop continues until the robotic arm is within a specified tolerance of the goal position or until the maximum number of iterations is reached. The tolerance is set to 15 millimeters, ensuring that the robotic arm stops when it is sufficiently close to the goal.

while norm(current_pos - goal) > tolerance && iteration < max_iterations

The maximum number of iterations (max_iterations) is set to prevent the algorithm from running indefinitely in case the goal cannot be reached. This ensures that the path planning process terminates gracefully if the robotic arm becomes stuck.

Visualization

The script includes two main visualizations:

- **Path Visualization**: During each iteration of the while-loop, the current position of the robotic arm is plotted in real time to visualize its path towards the goal. The goal position is marked with a red star (r*), while obstacles are represented by black squares (ks). The robotic arm's path is traced using blue circles (bo), and the final position is marked by a green star (g*). This visualization provides an intuitive understanding of how the robotic arm moves towards the goal while avoiding obstacles.

```
figure;

hold on;

plot3(goal(1), goal(2), goal(3), 'r*', 'MarkerSize', 10); % Goal position

plot3(obstacles(:, 1), obstacles(:, 2), obstacles(:, 3), 'ks', 'MarkerSize', 10, 'MarkerFaceColor', 'k'); % Obstacles

plot3(current_pos(1), current_pos(2), current_pos(3), 'bo'); % Starting position

axis equal;

grid on;

xlabel('X-axis (millimeters)');

ylabel('Y-axis (millimeters)');

zlabel('Z-axis (millimeters)');

title('Potential Field Path Planning for Welding Points');
```

- **3D Potential Field Visualization**: A 3D visualization of the potential field is provided to illustrate the influence of attractive and repulsive forces across the workspace. This visualization helps in understanding the potential landscape that guides the robotic arm's movement.

  The potential field is calculated over a grid of points, where each point represents a location in the workspace. The potential is a combination of attractive potential towards the goal and repulsive potential away from obstacles.

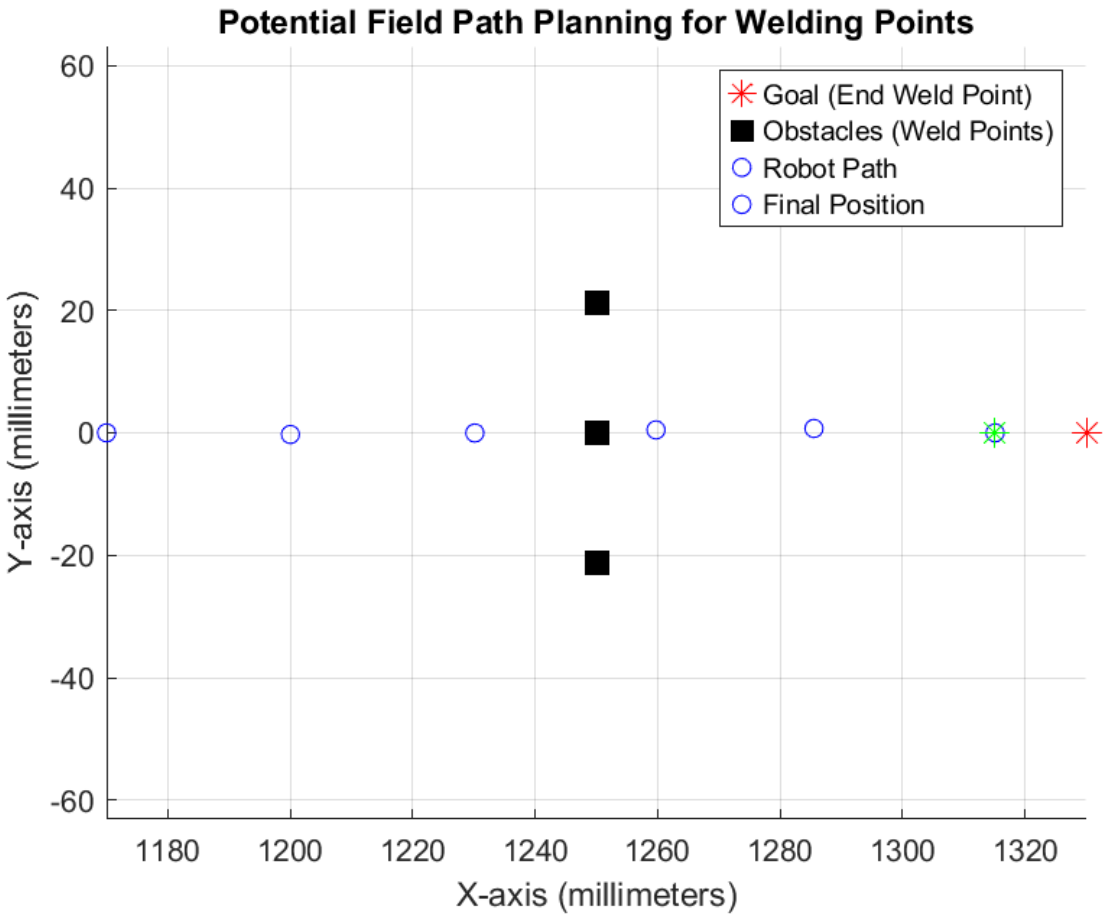
```
function visualize_potential_field(goal, obstacles, k_att, k_rep, d0)

% Create a grid for visualization

[X, Y] = meshgrid(1100:10:1400, -100:5:100);

Z = zeros(size(X));

% Calculate potential field for each point in the grid

for i = 1:size(X, 1)

        for j = 1:size(X, 2)

                pos = [X(i, j), Y(i, j), goal(3)];

                % Attractive potential

                U_att = 0.5 * k_att * norm(pos - goal)^2;

                % Repulsive potential

                U_rep = 0;

                for k = 1:size(obstacles, 1)

                        d_obs = norm(pos - obstacles(k, :));

                        if d_obs < d0
```

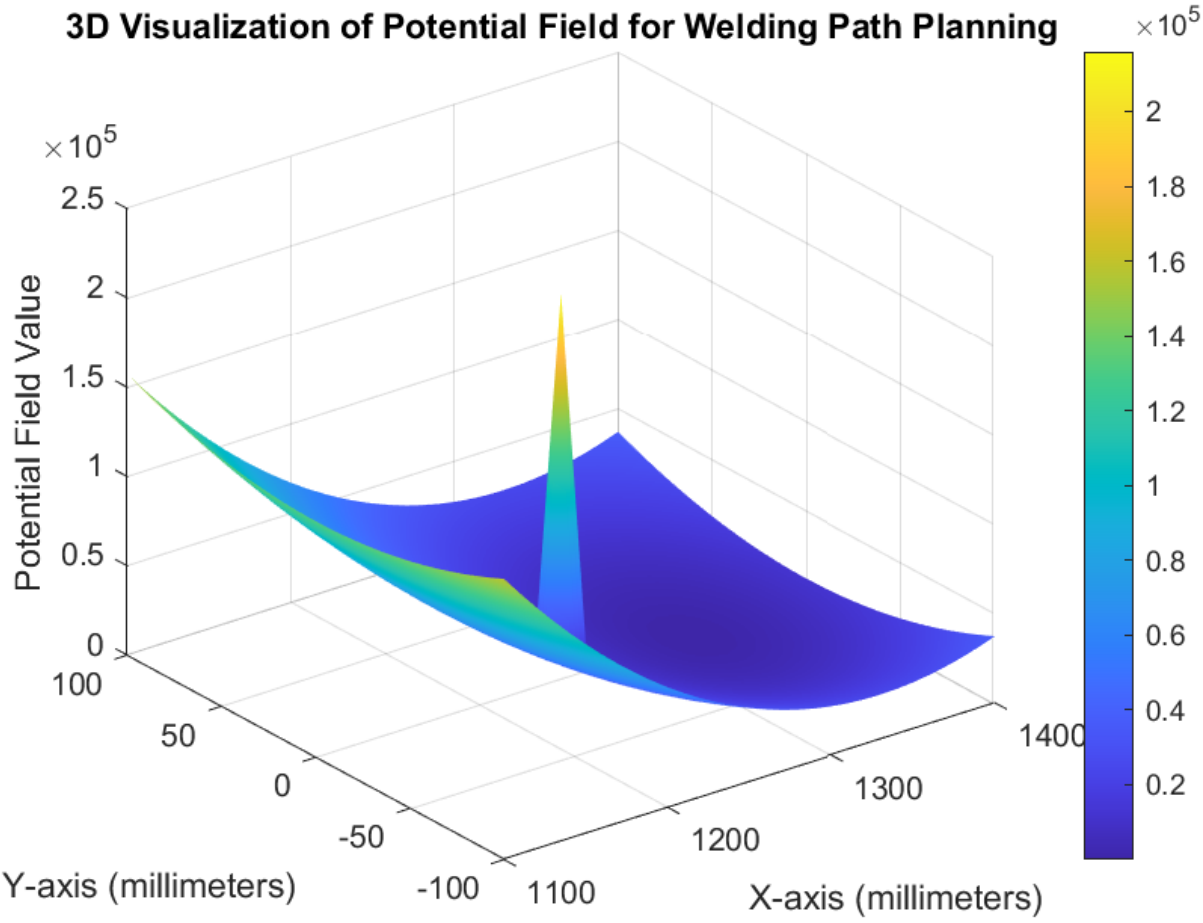
```
                    U_rep = U_rep + 0.5 * k_rep * (1/d_obs - 1/d0)^2;
                end
            end
            % Total potential
            Z(i, j) = U_att + U_rep;
        end
    end
    % Plot the potential field
    figure;
    surf(X, Y, Z);
    shading interp;
    colorbar;
    xlabel('X-axis (millimeters)');
    ylabel('Y-axis (millimeters)');
    zlabel('Potential Field Value');
    title('3D Visualization of Potential Field for Welding Path Planning');
    end
```

In this visualization:

- The **attractive potential (U_att)** is calculated as 0.5 * k_att * norm(pos - goal)^2, which increases quadratically with the distance from the goal. This ensures a smooth pull towards the target.
- The **repulsive potential (U_rep)** is calculated for each obstacle within the influence distance d0. The value of U_rep increases significantly as the position gets closer to the obstacle, thereby creating a "hill" in the potential field that the robotic arm will naturally avoid.
- The final potential field (Z) is the combination of attractive and repulsive potentials, providing a complete representation of the forces at play in the workspace.

This 3D surface plot helps in visualizing the potential landscape that drives the robot's movement, showing areas of high potential around obstacles and a valley towards the goal.


Potential Field Path Planning for Welding Points
Goal (End Weld Point)
Obstacles (Weld Points)
Robot Path
Final Position
Y-axis (millimeters)
X-axis (millimeters)
60
40
20
0
-20
-40
-60
1180
1200
1220
1240
1260
1280
1300
1320


3D Visualization of Potential Field for Welding Path Planning
×10^5
Potential Field Value
2.5
2
1.5
1
0.5
0
Y-axis (millimeters)
100
50
0
-50
-100
X-axis (millimeters)
1100
1200
1300
1400
×10^5
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2

(a) Varying the Range for the Repulsive Fields

For this part of the evaluation, we will choose two different values for the influence distance (d0) of the repulsive fields and analyze their effects on the path taken by the robot.

1. **Repulsive Range Variation 1:**
    - **d0 = 50 mm (Current Value)**: The current influence distance is set to 50 mm.
        - **Observation**: The current value provides an intermediate influence, where obstacles create repulsive fields when the robot comes within 50 mm of them. This means the robot will change its trajectory only when it comes relatively close to obstacles.
2. **Repulsive Range Variation 2:**
    - **d0 = 100 mm**: We will increase the influence distance to 100 mm.
        - **Observation**: With a larger d0, obstacles influence the robot from a further distance. This often results in the robot trying to avoid obstacles earlier, making its path more conservative (i.e., keeping a larger distance from obstacles).
        - **Result**: As a result, the path will be less direct, and the robot will take a more convoluted path to avoid obstacles earlier.

Explanation:

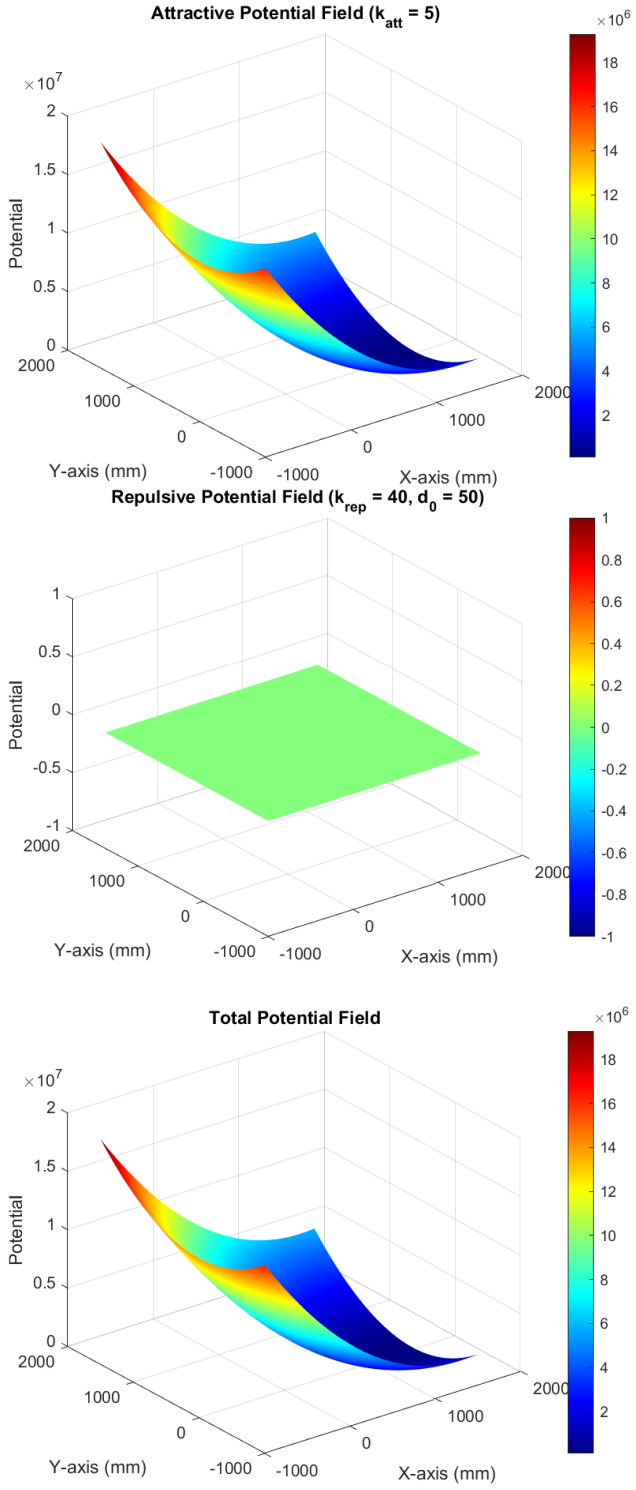
- Increasing d0 makes the repulsive force more conservative, meaning the robot will be influenced by obstacles earlier. This often results in a smoother path but could make it unnecessarily long.
- Decreasing d0 allows the robot to come closer to obstacles, which could result in a more direct path but at the risk of having sudden, sharp changes in trajectory or even getting stuck.

(b) Varying the Attractive and Repulsive Constants

For this part, we will vary both the attractive constant (k_att) and repulsive constant (k_rep) to see how their relative magnitudes affect the path taken.

1. Attractive and Repulsive Constants Variation 1:

- **k_att = 5.0**, **k_rep = 40.0** (Current Values):
    - **Observation**: The current value of k_att results in a moderate pull towards the goal, while k_rep has a relatively strong influence on avoiding obstacles. The robot finds a path with reasonable avoidance of obstacles and moves toward the goal efficiently.

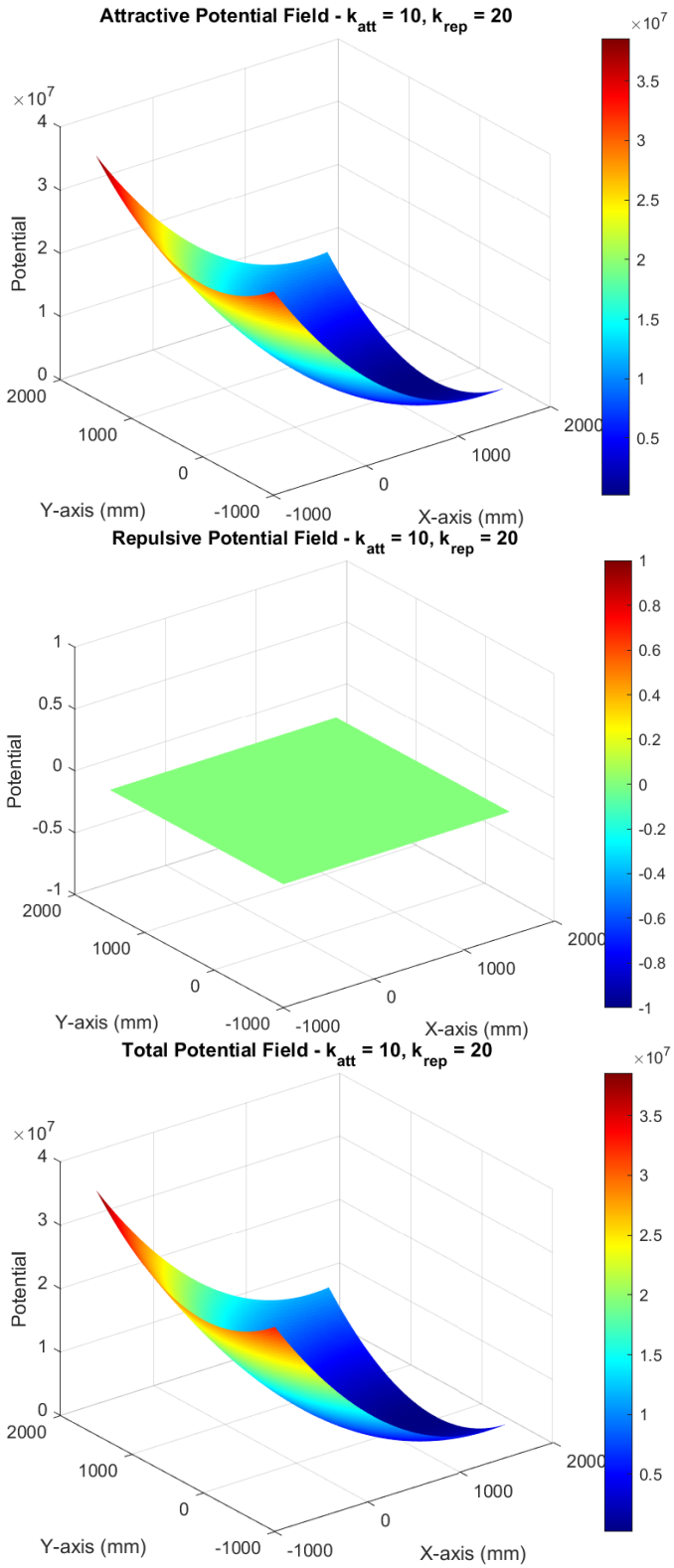


2. Attractive and Repulsive Constants Variation 2:

- **k_att = 10.0**, **k_rep = 20.0** (Increased Attractive and Decreased Repulsive Forces):
    - **Observation**: With an increased attractive constant and a decreased repulsive constant, the robot feels a stronger pull towards the goal and is less concerned with avoiding obstacles. This could lead to a more aggressive approach to reaching the goal, potentially bringing it closer

to obstacles and increasing the risk of collisions. However, the path will be more direct.

- **Result**: The robot may attempt to take a more direct path to the goal and potentially get closer to obstacles.



Explanation:

- **Higher k_{att} and Lower k_{rep}**: The robot is more aggressive in reaching the goal, but it can take risks in getting too close to obstacles. This can make the path faster but less safe.
- **Lower k_{att} and Higher k_{rep}**: The robot will prioritize avoiding obstacles, which could lead to a longer path with hesitation. The robot may also get stuck in local minima if the repulsive forces dominate.

Appendix:

Program:-

```
clc;
function potential_field_path_planning_welding()
    % Define constants
    k_att = 5.0;         % Attractive constant to significantly strengthen goal pull
    k_rep = 40.0;        % Repulsive constant to weaken obstacle avoidance
    d0 = 50;             % Decreased influence distance for repulsive field (in millimeters)
    % Define goal position (Weld Point 3)
    goal = [1330.01, 0, 190.32]; % End position in millimeters
    % Define obstacles as a set of points (in millimeters)
    obstacles = [
        1250.01, 0, 190.32;    % Middle Weld Point
        1250.01, -21.35, 190.32; % Left Side Weld Point
        1250.01, 21.35, 190.32   % Right Side Weld Point
    ];
    % Initial position (Weld Point 1)
    current_pos = [1170.01, 0, 190.32]; % Start position in millimeters
    % Define step size for movement (in millimeters)
    step_size = 30; % Increased step size for faster movement
    % Set tolerance for reaching the goal (in millimeters)
    tolerance = 15; % Slightly reduced tolerance for more precise stopping
    % Plot the environment
    figure;
    hold on;
    plot3(goal(1), goal(2), goal(3), 'r*', 'MarkerSize', 10); % Goal position
    plot3(obstacles(:, 1), obstacles(:, 2), obstacles(:, 3), 'ks', 'MarkerSize', 10,
'MarkerFaceColor', 'k'); % Obstacles
    plot3(current_pos(1), current_pos(2), current_pos(3), 'bo'); % Starting position
    axis equal;
    grid on;
    xlabel('X-axis (millimeters)');
```

```
    ylabel('Y-axis (millimeters)');
    zlabel('Z-axis (millimeters)');
    title('Potential Field Path Planning for Welding Points');
    % Iteratively move towards the goal
    iteration = 0; % to avoid infinite loop
    max_iterations = 2000; % Allow enough iterations for goal completion
    while norm(current_pos - goal) > tolerance && iteration < max_iterations
        % Calculate attractive force (towards goal)
        F_att = -k_att * (current_pos - goal);
        % Calculate repulsive force (from obstacles)
        F_rep = [0, 0, 0];
        for i = 1:size(obstacles, 1)
            d_obs = norm(current_pos - obstacles(i, :));
            if d_obs < d0
                % Repulsive force calculation with reduced penalty for closer distances
                penalty = (d0 / d_obs); % Reduced penalty to allow smoother movement
                F_rep = F_rep + penalty * k_rep * (1/d_obs - 1/d0) * (1/d_obs^2) * ...
                        (current_pos - obstacles(i, :)) / d_obs;
            end
        end
        % Total force (combined attractive and repulsive)
        F_total = F_att + F_rep;

        % Normalize the total force and apply step size to update position
        if norm(F_total) > 0
            % Adaptive step size based on force balance
            adaptive_step_size = step_size / (1 + norm(F_rep));
            % Adding a small random perturbation to escape local minima
            random_perturbation = (rand(1, 3) - 0.5) * 1.0; % Increased random noise for better
escape
            current_pos = current_pos + adaptive_step_size * F_total / norm(F_total) +
random_perturbation;
        end
```

```
        % Plotting the new position at every iteration for a more elaborate path
        plot3(current_pos(1), current_pos(2), current_pos(3), 'bo');
        iteration = iteration + 1;
    end
    % Plot the final position
    plot3(current_pos(1), current_pos(2), current_pos(3), 'g*', 'MarkerSize', 10); % Final
position
    legend('Goal (End Weld Point)', 'Obstacles (Weld Points)', 'Robot Path', 'Final Position');
    hold off;
    if iteration >= max_iterations
        disp('Reached maximum number of iterations. Path might be stuck.');
    end
    % Visualize Potential Field in 3D
    visualize_potential_field(goal, obstacles, k_att, k_rep, d0);
end
function visualize_potential_field(goal, obstacles, k_att, k_rep, d0)
    % Create a grid for visualization
    [X, Y] = meshgrid(1100:10:1400, -100:5:100);
    Z = zeros(size(X));
    % Calculate potential field for each point in the grid
    for i = 1:size(X, 1)
        for j = 1:size(X, 2)
            pos = [X(i, j), Y(i, j), goal(3)];
            % Attractive potential
            U_att = 0.5 * k_att * norm(pos - goal)^2;
            % Repulsive potential
            U_rep = 0;
            for k = 1:size(obstacles, 1)
                d_obs = norm(pos - obstacles(k, :));
                if d_obs < d0
                    U_rep = U_rep + 0.5 * k_rep * (1/d_obs - 1/d0)^2;
                end
            end
```

```
            % Total potential
            Z(i, j) = U_att + U_rep;
        end
    end
    % Plot the potential field
    figure;
    surf(X, Y, Z);
    shading interp;
    colorbar;
    xlabel('X-axis (millimeters)');
    ylabel('Y-axis (millimeters)');
    zlabel('Potential Field Value');
    title('3D Visualization of Potential Field for Welding Path Planning');
end

potential_field_path_planning_welding();
```