# Assignment 2
# SC627

Bhavini Jeloka
18D100007

## I. Introduction to the Potential Function Based Planner

The potential function based approach for path planning draws from the concept of electrostatic attraction and repulsion. The bot is 'attracted' to the goal and 'repelled' from the obstacle. Along the same lines, the bot desires to minimise its potential. The attractive and repulsive potentials can be formulated using various functions. In our case, the attractive potential is either a quadratic or a conical function based on its vicinity to the goal while the repulsive potential is the sum of the all the obstacle repulsive potentials that are activated when the bot is in the vicinity (more about this in further sections). The gradient of these potentials is utilized in defining the next step and evaluates to zero (minimum) at the goal.

### A. Algorithm Initiation

We use the functions in the helper code to implement the logic explained above. When the algorithm is initiated it takes as its input, the obstacles, the goal and start point. Then, the gradient of the potential (sum of attractive and repulsive potentials) at the point is calculated. If its norm is greater than a certain tolerance, the bot is expected to move. At each instant, this gradient is calculated to decide the direction of motion. For calculating these gradients, we invoke the functions built earlier.

### B. Dealing with Obstacles

If the bot nears an obstacle (beyond a threshold), the obstacle begins to influence the bot via repulsion. Consequently, the bot would move along a direction that maximises its distance from the obstacle. In order to achieve this, we find the point on the obstacle that is closest to the bot, and move in the direction normal to it. This is precisely what the repulsive gradient does. This enables collision avoidance.

### C. Termination

As mentioned earlier, the algorithm terminates when the gradient is zero. To allow for precision errors, we introduce a tolerance on the norm of the function.

## II. Numerical Results and Discussion

We test our algorithm in a 2D environment with two obstacles and obtain the path as portrayed in Figure 1.

On a closer look, we see that the algorithm's termination point is significantly deviated from the goal (4.72, 3.34). The cause for such a large error margin could be attributed to:

1) The algorithm might have reached a local minima where the gradient is zero. This can prevent the bot from moving further.
2) The norm of the gradient has a small magnitude (still, greater than the tolerance) which increases the run time of the algorithm. Thus, in order to terminate the algorithm, we impose an empirical tolerance value (derived from previous runs). An alternative approach would be to terminate the algorithm once the bot is within a $\delta$ ball of the goal.

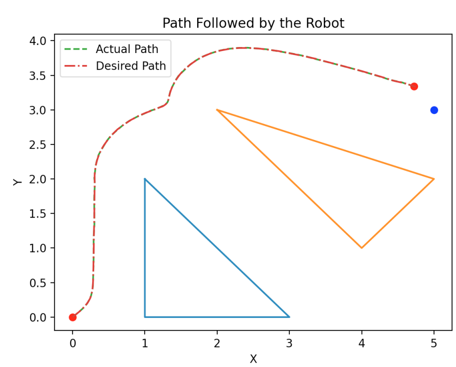


Fig. 1. Potential Function Based Planning