Robotics-Reinforcement Learning with Function Approximation

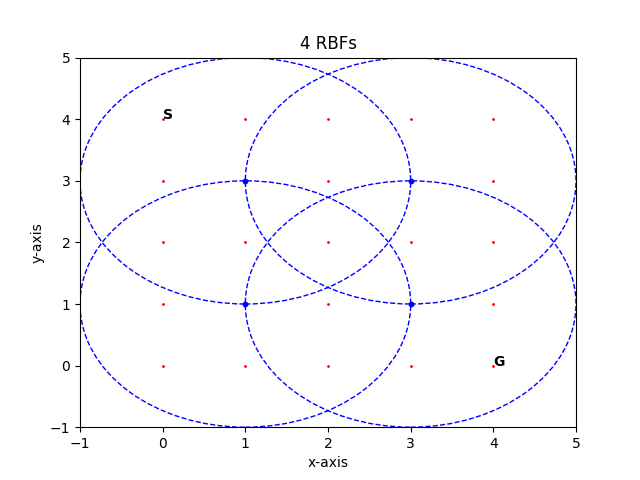
**Submit the project report to cover all of the requirements.**

**Submit your source code with instructions to run your code.**

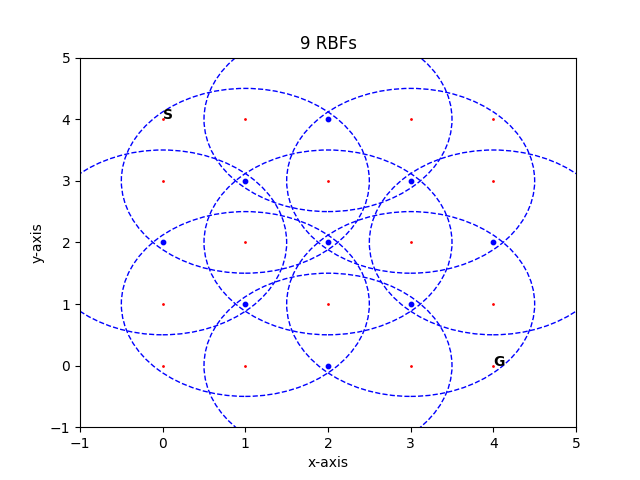
**Project 2: Deadline 2/26/25**

The problem is the same as Project 3a. Using Radial Basis Functions (RBF) to approximate the state space. You can use 4 RBF to approximate the state space. Something is similar to these figures.

**4 RBF cover:**

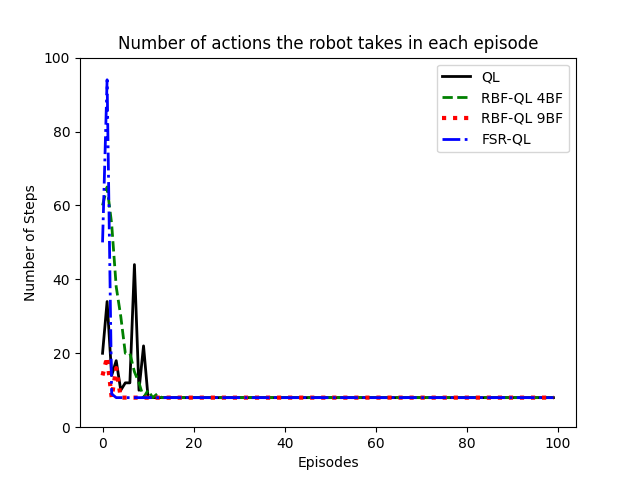


**9 RBF cover:**

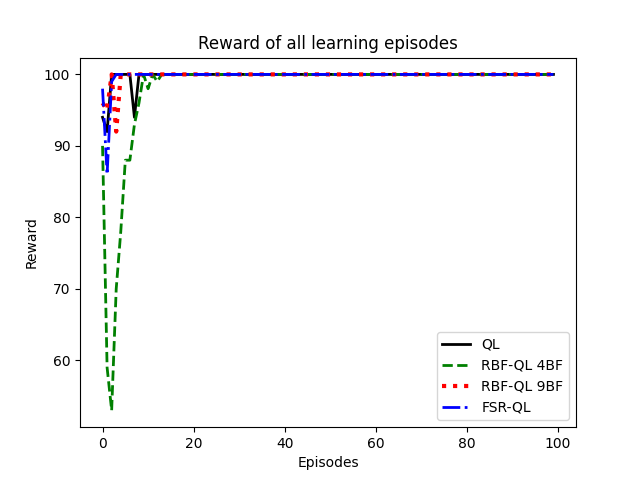


**Requirement:**

1. (UG: 50 points, G: 40 points) Plot the number of actions the robot takes in each episode for normal Q learning (QL), RBF-QL with 4BF, and RBF-QL with 9BF. Something is similar to this figure



1. (UG: 50 points, G: 40 points) Plot the reward of all learning episodes for each method: Q learning (QL), RBF-QL with 4BF, and RBF-QL with 9BF.



1. (Grad: 20 points) For Grad students (CPE671 only): Implement the FSR and plot its results to compare to the RBF-QL with 9BF (see the figures in Sections 1 and 2). Undergrad students are welcome to implement this FSR with +10 points bonus.

**Appendix:**

The normal Q-learning equation is here:

The full pseudo code of the approximated Q-learning (just need to use equation 4) is here:





Note that, Equation (4) is for one dimension state. For 2 or more dimensions, use the equation in the lecture slide, or below:

**For 9 RBF Matlab code**

clc; clear;

n = 5;

state = states(n);

gamma = 0.9;

alpha = 0.65;

action = {[1,0];[-1,0];[0,1];[0,-1]}; %up,down,right,left

goal = [n,n];

goal\_index = search(goal,state);

reward = initialize\_rewards(n,state);

next\_state = [0,0];

tf = 0;

iterations\_9rbf = [];

reward\_func\_9rbf = [];

center = [[1 1]; [1 5]; [2 2]; [2 4]; [3 3]; [4 2]; [4 4]; [5 1]; [5 5]];

sigma = 1.2;

theta = rand(length(center)\*4,1); % theta dimension is 36x1; number 4 hear means 4 actions

**%===================================================================================**

**For 4 RBF Matlab code:**

clc; clear;

n = 5;

state = states(n);

gamma = 0.9;

alpha = 0.89;

action = {[1,0];[-1,0];[0,1];[0,-1]}; %up,down,right,left

goal = [n,n];

goal\_index = search(goal,state);

reward = initialize\_rewards(n,state);

next\_state = [0,0];

tf = 0;

iterations\_4rbf = [];

reward\_func\_4rbf = [];

center = [[2 2]; [2 4]; [4 2]; [4 4]];

sigma = 1.425;

theta = rand(length(center)\*4,1); % theta dimension is 16x1; number 4 hear means 4 actions