Patrick Jr Mcbride
CSC 449/549 — Advanced Topics in Artificial Intelligence
Programming Assignment 3
December 2, 2022

Program 3 Results

Sarsa(Lambda)

For this assignment I was tasked with implementing SARSA(Lambda) to solve the mountain car problem. Following the methods described in https://people.cs.umass.edu/˜pthomas/papers/Konidaris2011a.pdf I implemented SARSA(Lambda) with two different bases.

To give me access to easy rendering of the simulated problem I decide to use OpenAI's Gym library which includes a simulator for the mountain car problem. The mountain car simulator has the same state space as the one described in Sutton and Barto.

My program is written in Python 3.10 and uses gym, numpy, matplotlib and pygame.

During my testing of my Fourier basis, I found that the lambda, alpha, and gamma value used in the paper also worked for my program. However, when testing my RBF implementation I needed larger alpha values to get satisfactory results.
For all of my testing I ran without rendering to the screen to reduce training times.

Program Usage

After installing the libraries in the requirements.txt file open the mtCarTrainer.py file. This is the start point of the program and also where the training setting can be modified. The program is initially setup to run 1000 training episodes with a order 3 Fourier basis. It will render the last 3 episodes to the screen and it will output the learning plot and Cost-To-Go plot.

The following are the adjustable parameters and what they do:

makePlots:

True - Outputs the learning plot and Cost-To-Go plot.

False - No plots are made.

numOfRenderedRuns:

Number of runs at the end of training that will be rendered.

episodes = 1000:

Number of training episodes to run.

basis:

“Fourier" - Use Fourier basis

“Radial" - Use Radial basis

order:

Set the order of the basis

Results For Fourier Basis

For these tests the following values were used:

alpha = 0.001

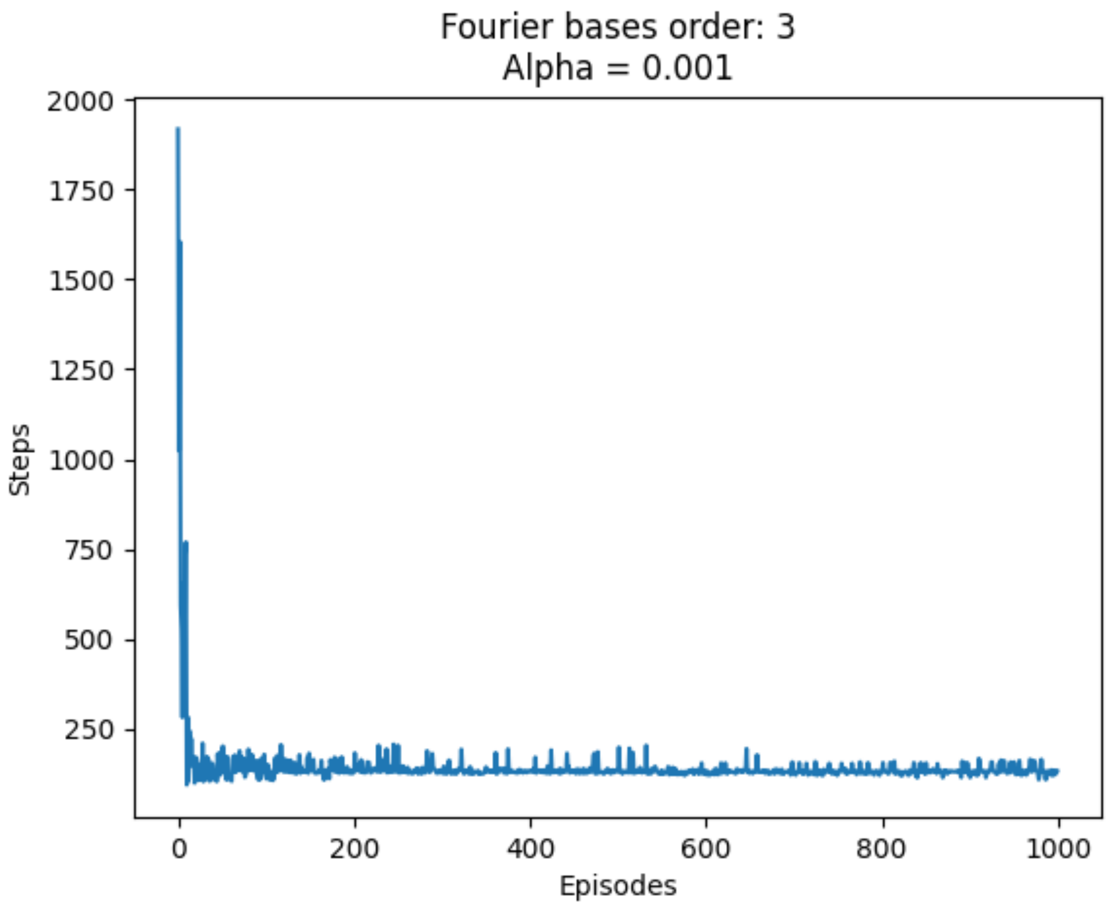
gamma = 1.0

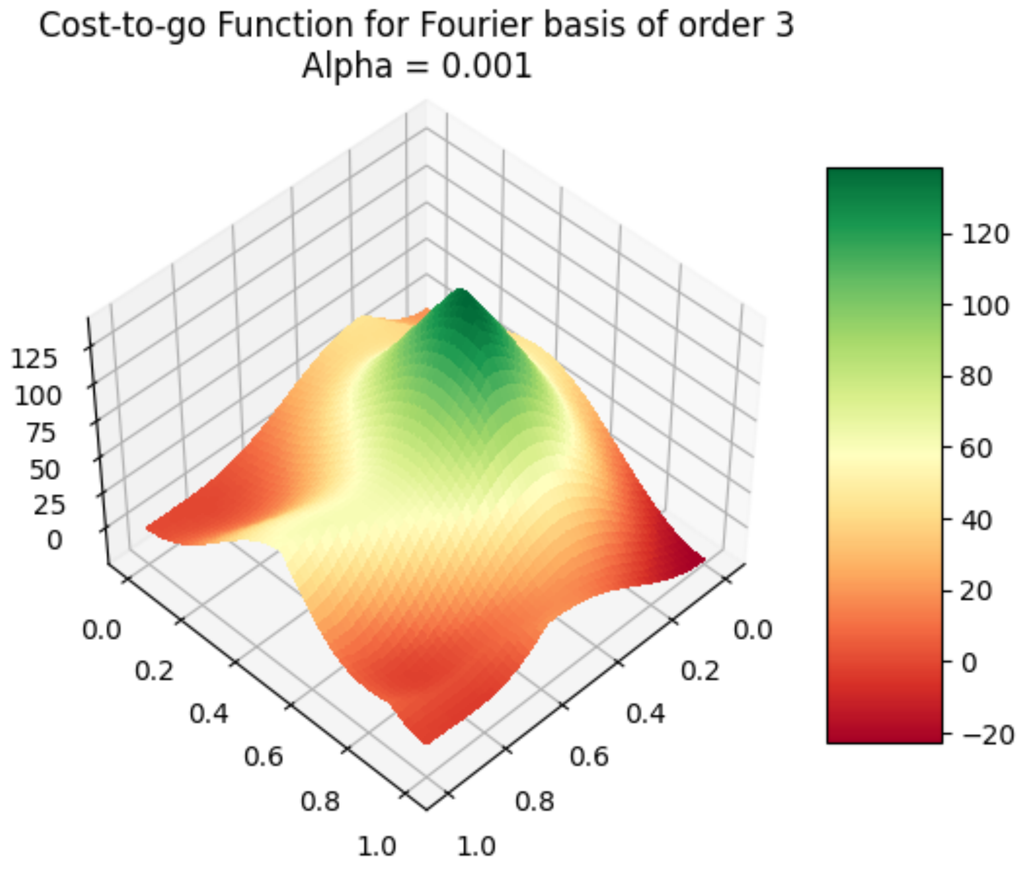
lamb = 0.9

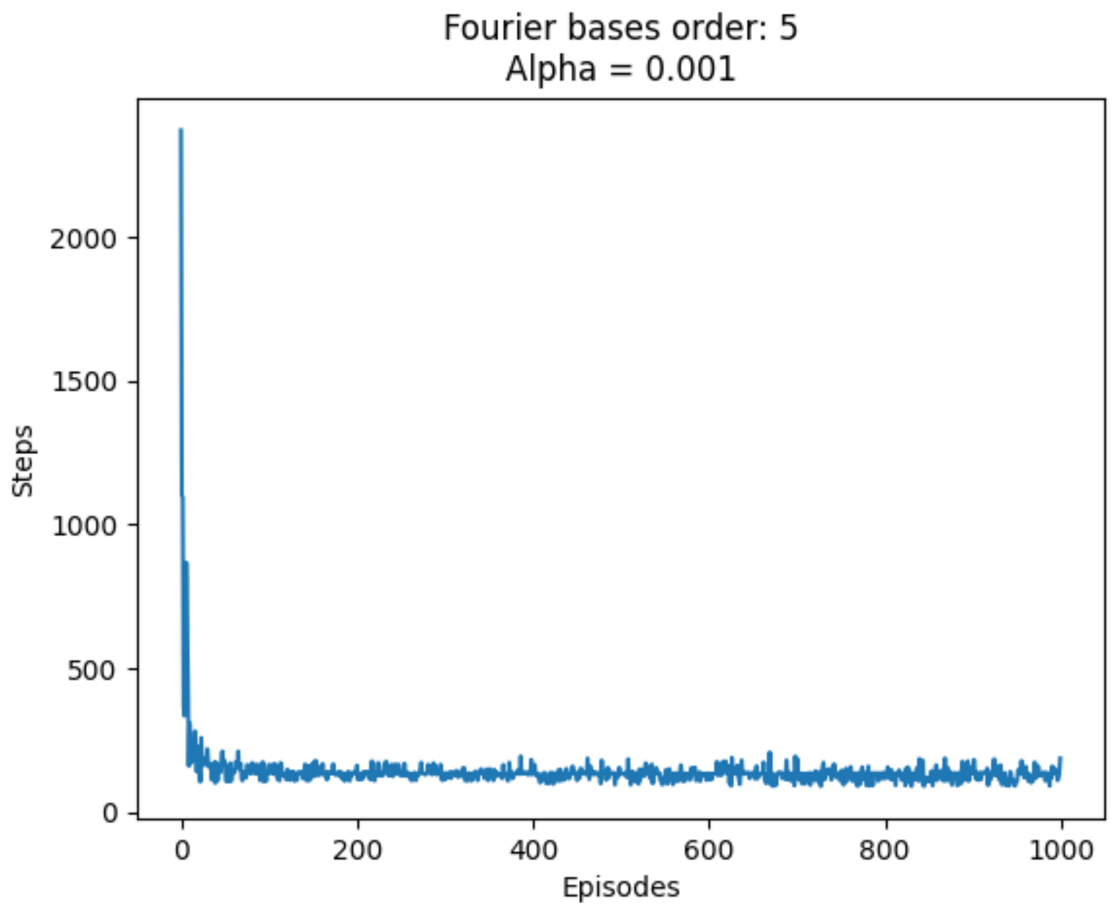
epsilon = 0.0

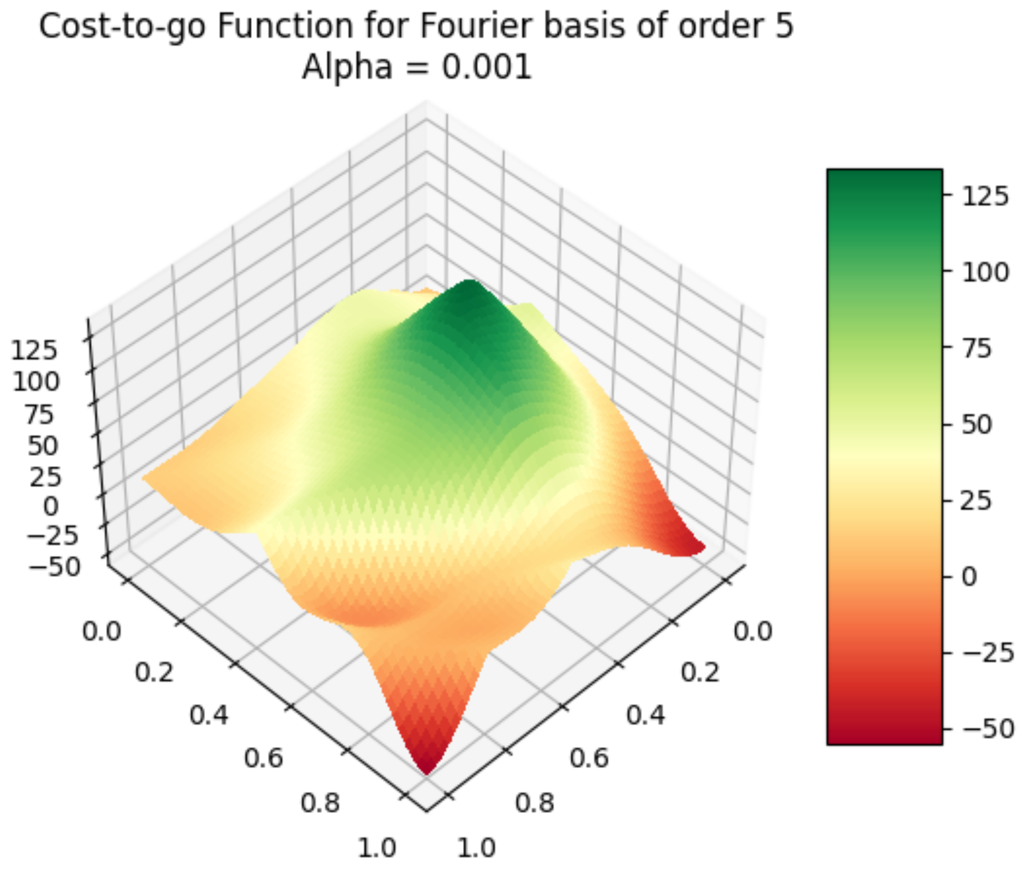
episodes = 1000

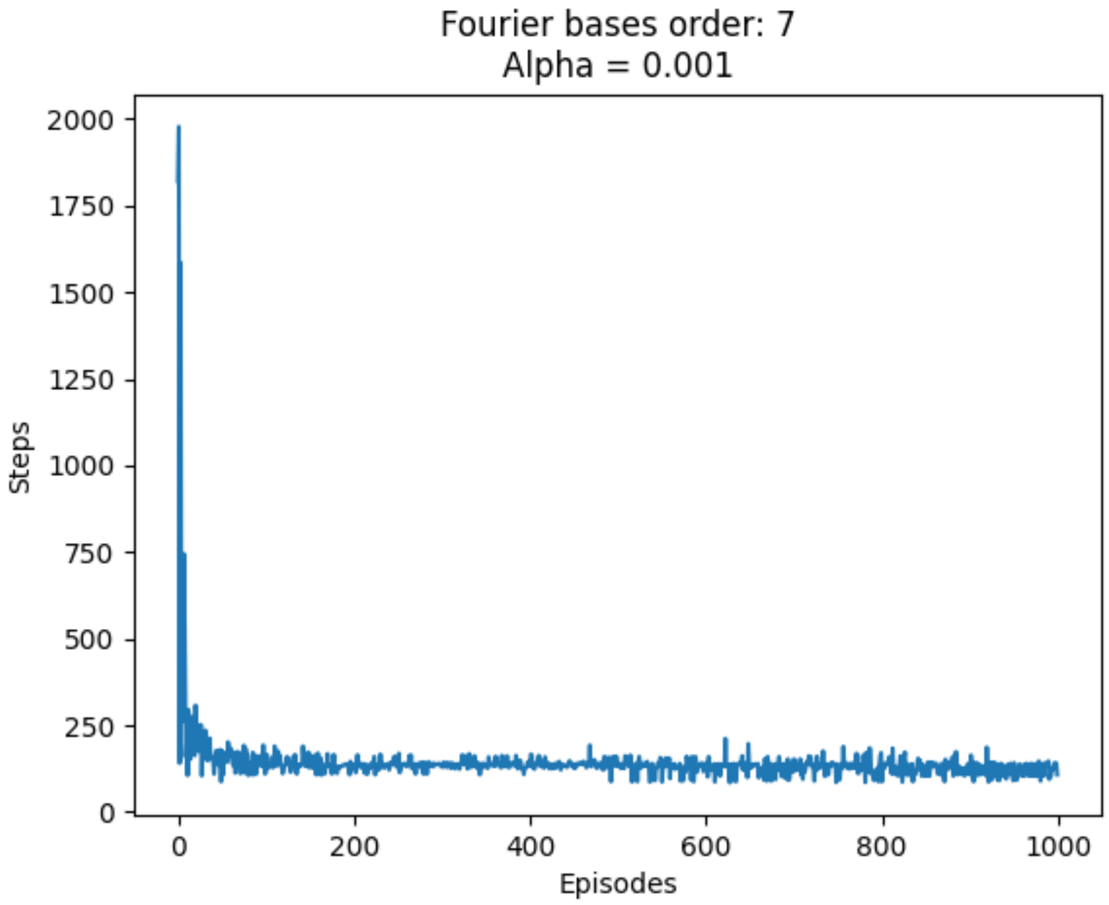
Performance for each of the orders that I tested was fairly similar.

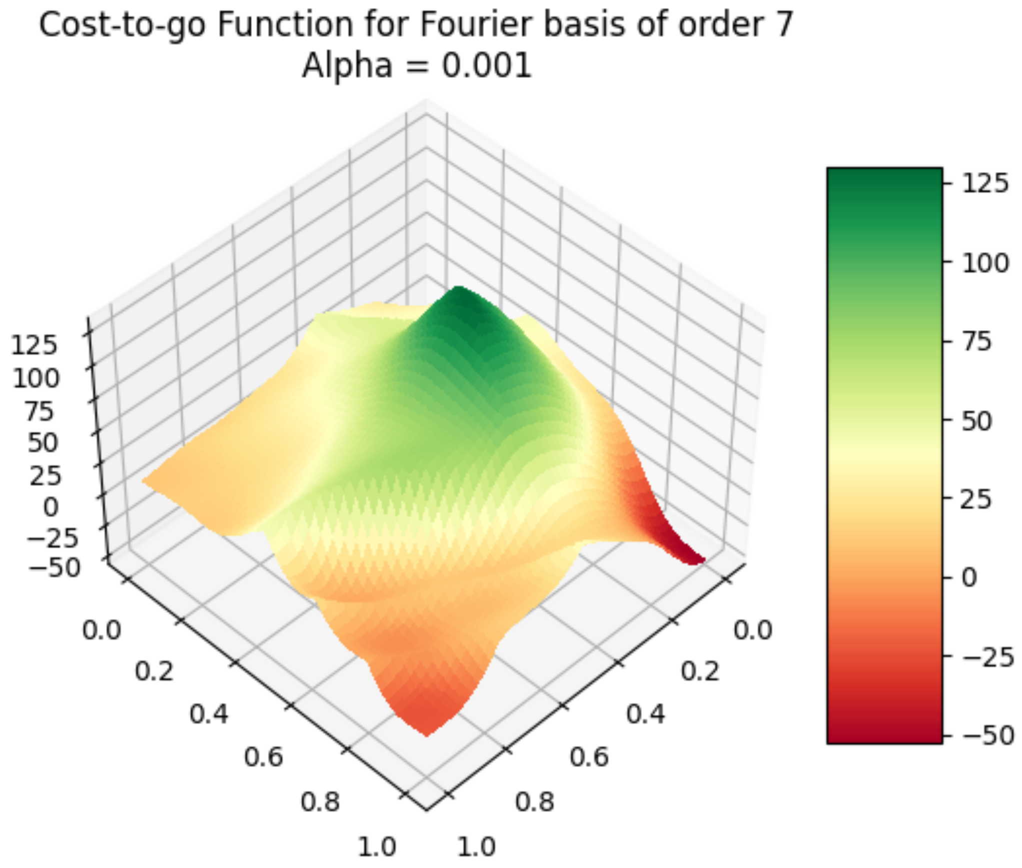

Fourier bases order: 3
Alpha = 0.001
Steps
Episodes
2000
1750
1500
1250
1000
750
500
250
0
200
400
600
800
1000


Cost-to-go Function for Fourier basis of order 3
Alpha = 0.001
125
100
75
50
25
0
0.0
0.2
0.4
0.6
0.8
1.0
120
100
80
60
40
20
0
−20


Fourier bases order: 5
Alpha = 0.001
Steps
Episodes
0
500
1000
1500
2000
0
200
400
600
800
1000


Cost-to-go Function for Fourier basis of order 5
Alpha = 0.001
125
100
75
50
25
0
−25
−50
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
125
100
75
50
25
0
−25
−50


Fourier bases order: 7
Alpha = 0.001
Steps
2000
1750
1500
1250
1000
750
500
250
0
0
200
400
600
800
1000
Episodes


Cost-to-go Function for Fourier basis of order 7
Alpha = 0.001
125
100
75
50
25
0
−25
−50
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
125
100
75
50
25
0
−25
−50

Results For Radial Basis

For these tests the following values were used:

alpha = 0.01

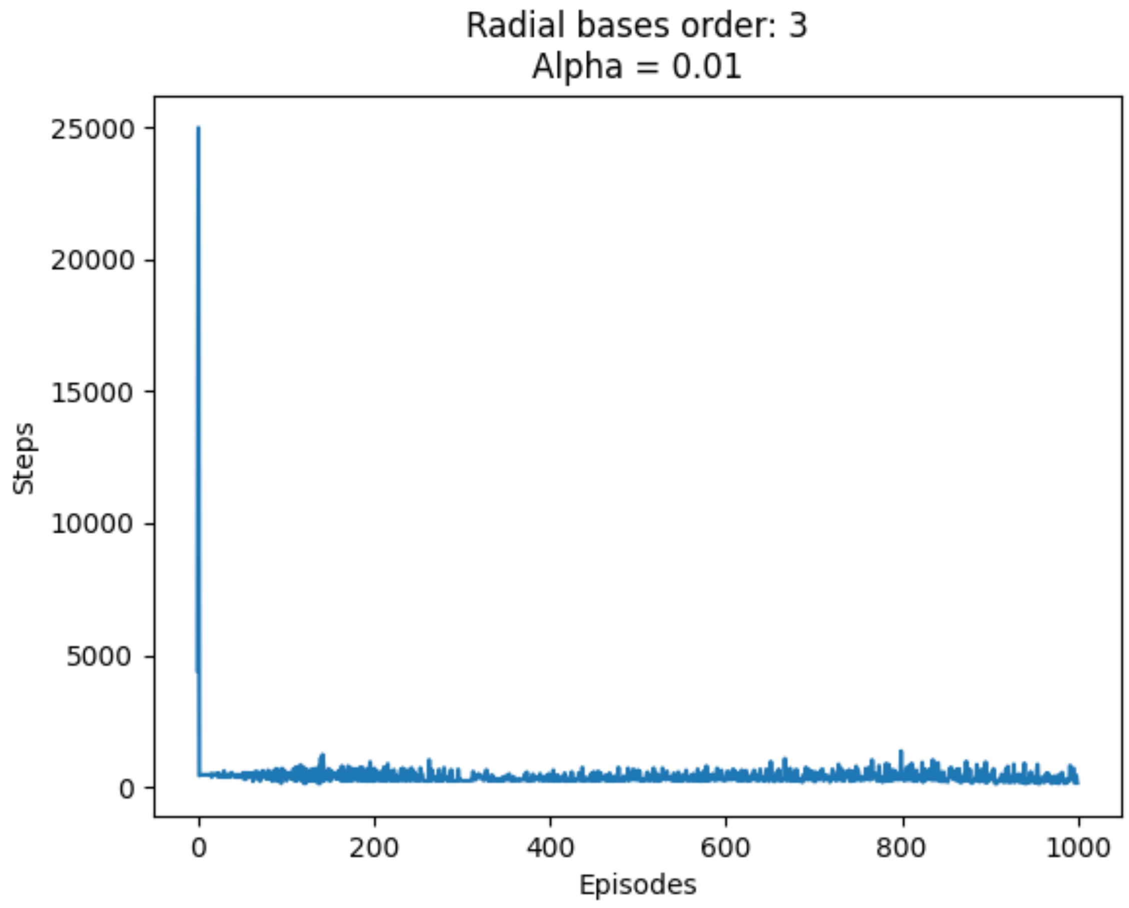
gamma = 1.0

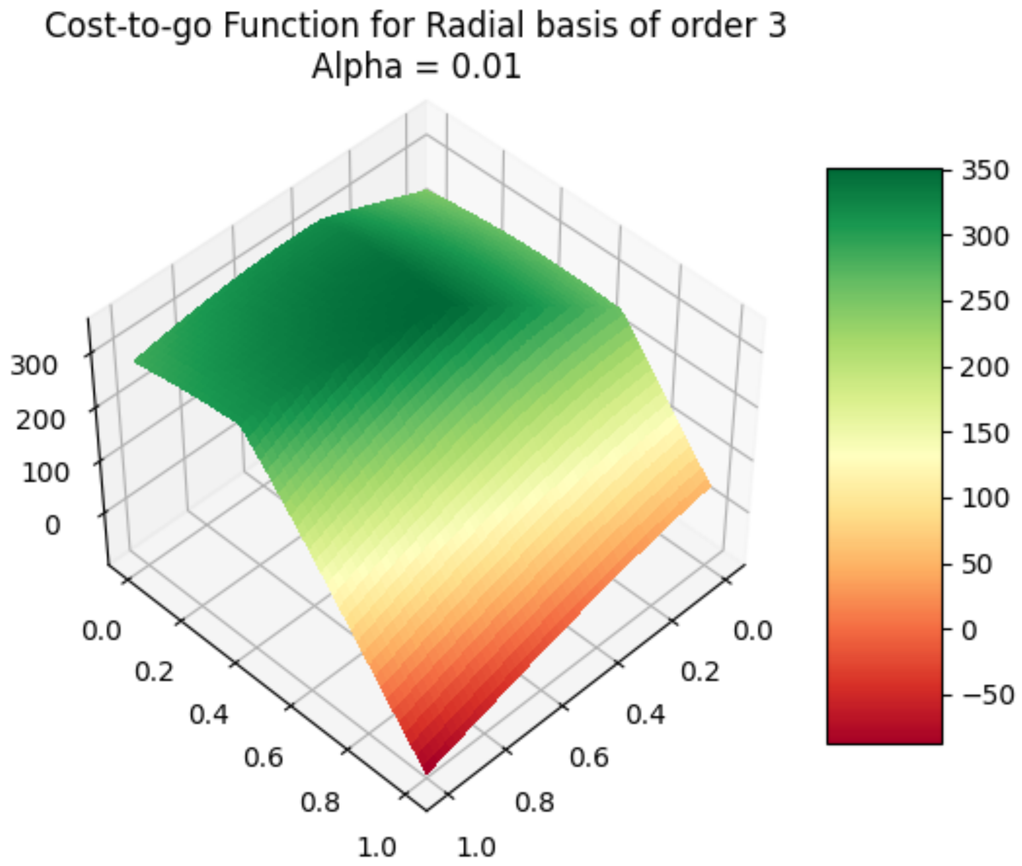
lamb = 0.9

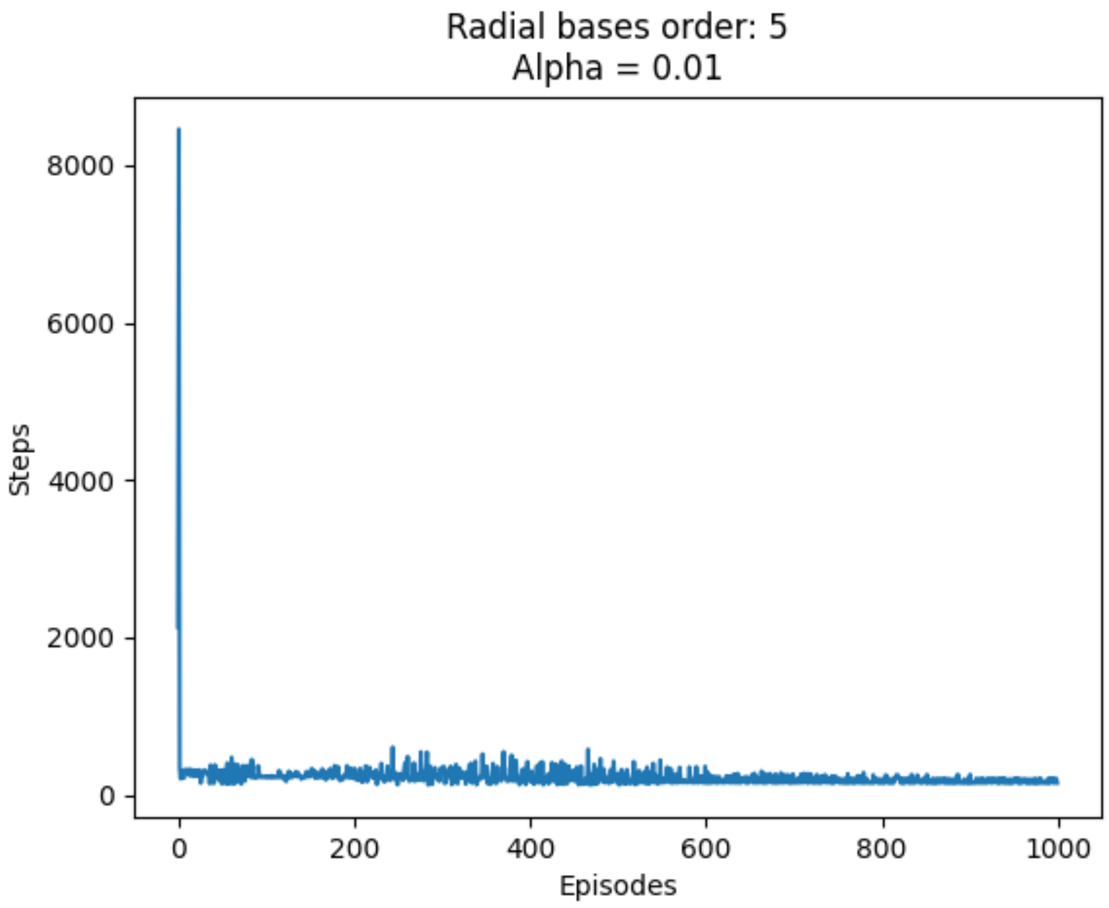
epsilon = 0.0

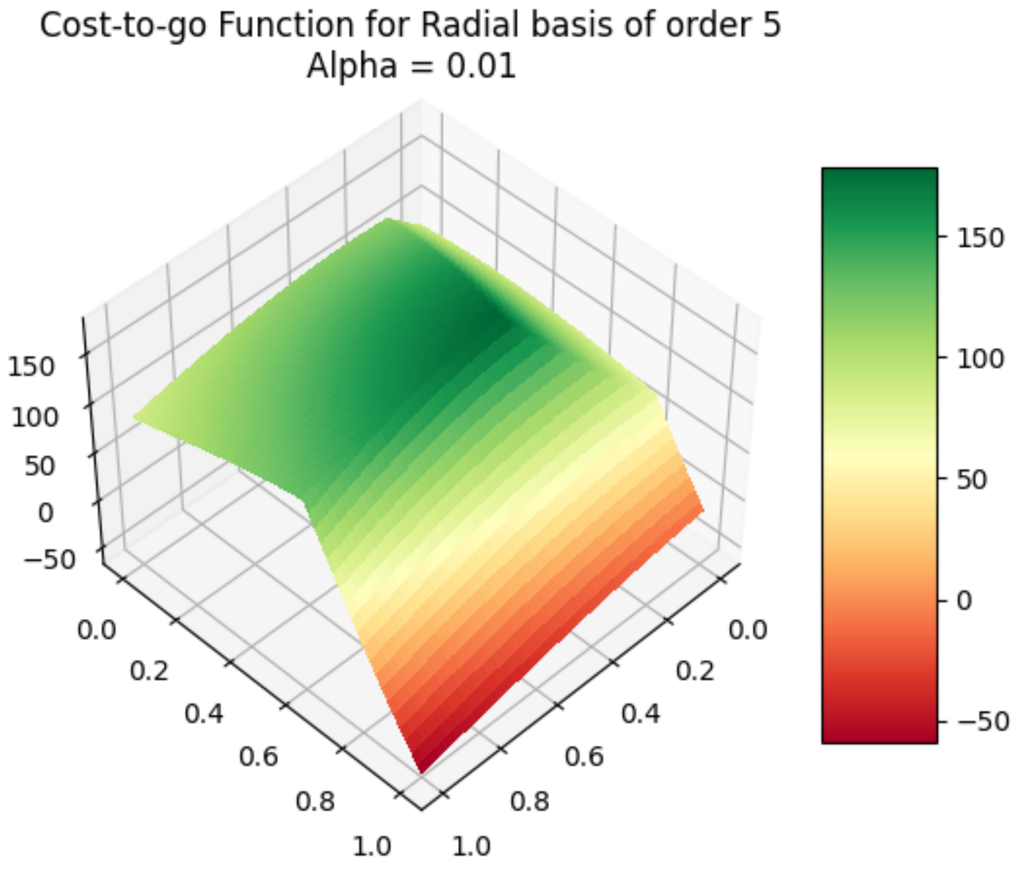
episodes = 1000

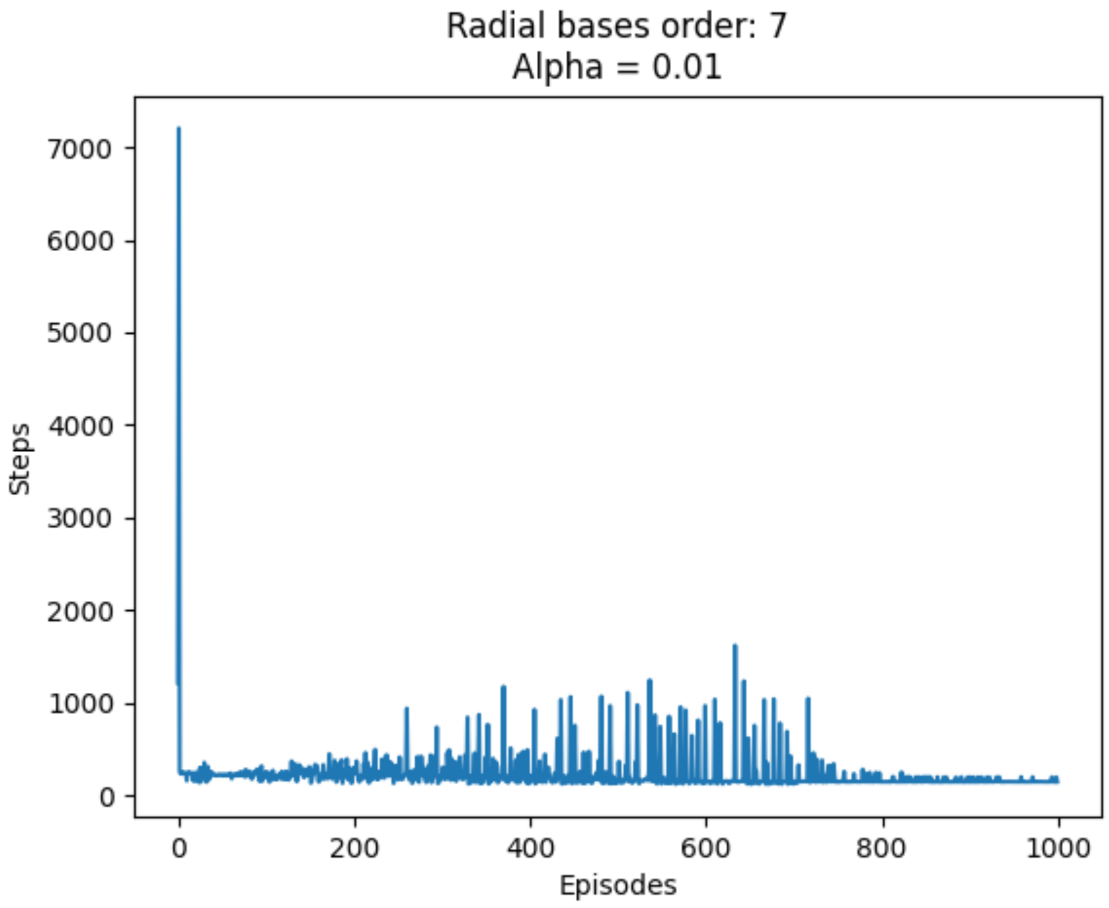
The best fit for radial basis was the order 11 basis. It had excellent results when compared to the order 3, 5 and 7 tests, and had similar performance to the Fourier basis.

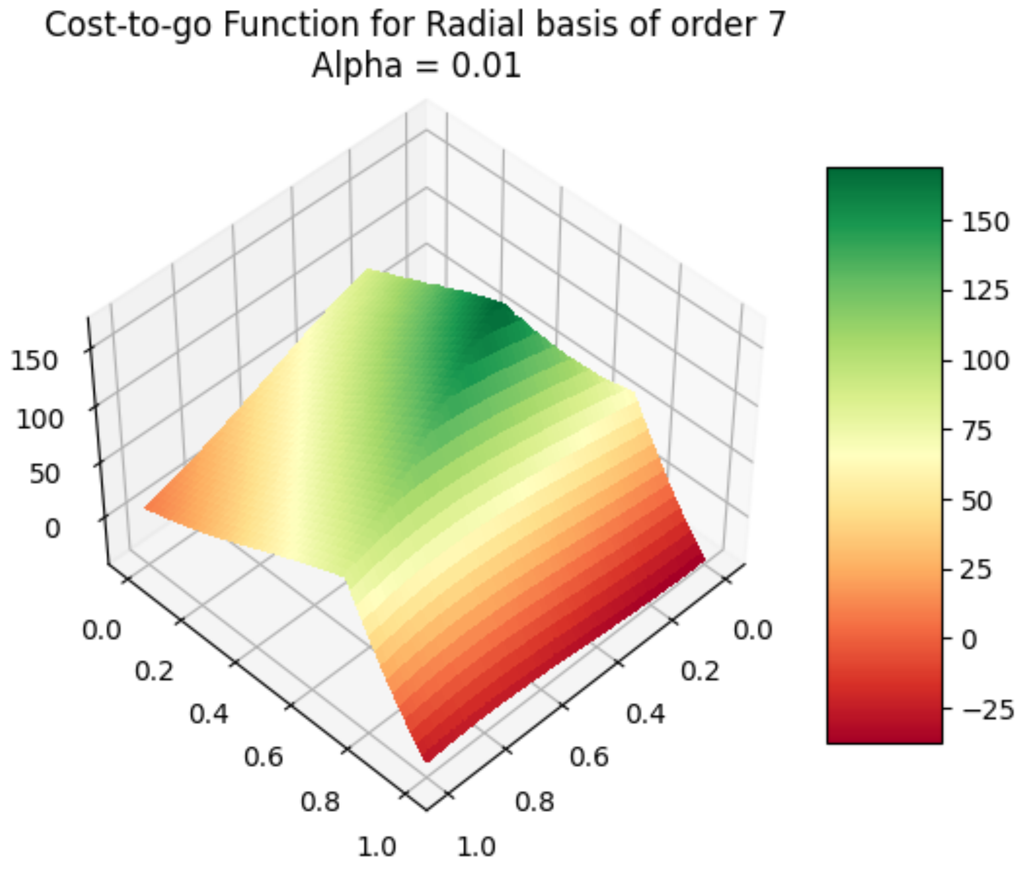

Radial bases order: 3
Alpha = 0.01
Steps
Episodes
25000
20000
15000
10000
5000
0
0
200
400
600
800
1000

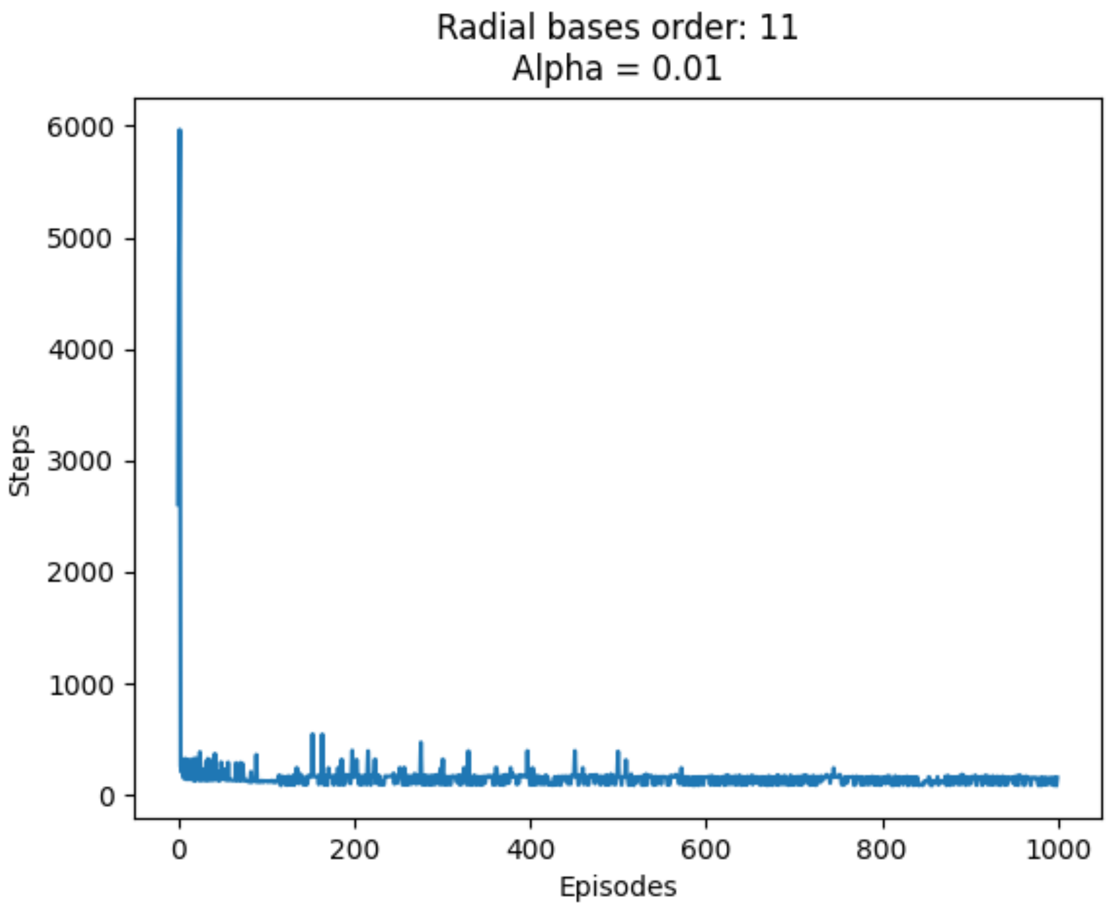

Cost-to-go Function for Radial basis of order 3
Alpha = 0.01
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
350
300
250
200
150
100
50
0
−50

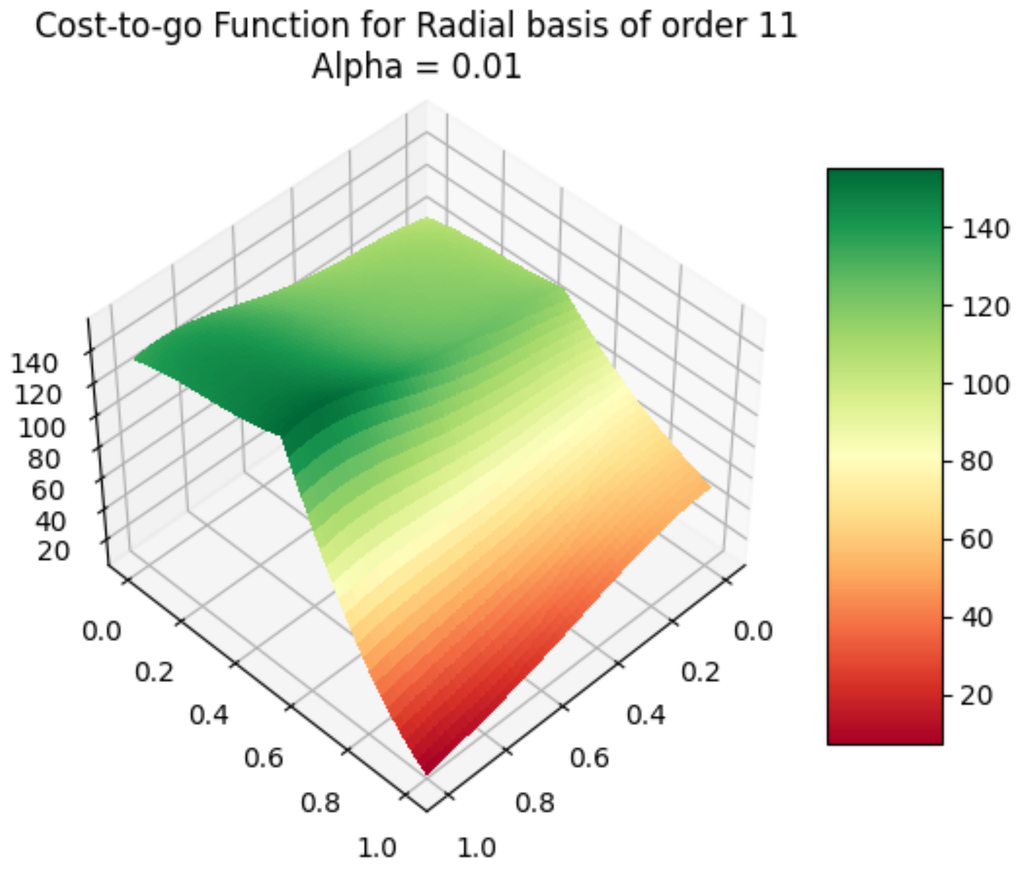

Radial bases order: 5
Alpha = 0.01
Steps
0
2000
4000
6000
8000
Episodes
0
200
400
600
800
1000


Cost-to-go Function for Radial basis of order 5
Alpha = 0.01
150
100
50
0
−50
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
150
100
50
0
−50


Radial bases order: 7
Alpha = 0.01
Steps
Episodes
0
1000
2000
3000
4000
5000
6000
7000
0
200
400
600
800
1000


Cost-to-go Function for Radial basis of order 7
Alpha = 0.01
0
50
100
150
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
150
125
100
75
50
25
0
−25


Radial bases order: 11
Alpha = 0.01
Steps
6000
5000
4000
3000
2000
1000
0
0
200
400
600
800
1000
Episodes


Cost-to-go Function for Radial basis of order 11
Alpha = 0.01
140
120
100
80
60
40
20
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
140
120
100
80
60
40
20

Questions

The Mountain Car contains a negative step reward and a zero goal reward. What would happen if γ was less than 1 and the solution was many steps long?

Initial training performance would be significantly hindered since the large negative reward that should be given to the actions taken at the beginning of the episode would end up being discounted.

What would happen if we had a zero-step cost and a positive goal reward, for the case where γ = 1, and the case where γ < 1?

In the case of gamma = 1, there is no incentive for the agent to learn an efficient policy since it values an episode of 100 steps the same as an episode of 10000 steps. So while it would learn a policy that would eventually get the car up the mountain, it would be incredibly inefficient and slow to train.

In the case of gamma < 1, the policy learned would also start out being inefficient it may learn a more efficient policy over time since the wasted actions taken at the beginning would be discounted.