Using the code:

On line 8, k controls the order of the basis function.

The code is currently using a polynomial basis. To change to fourier, change the “poly” function on lines 86, 101, and 110 to “fourier” using the same parameters. Finally, change the “polyGradient” function on line 143 to “gradientlog” again keeping the same parameters.

The code when run will print out the number of time steps every 10th episode took, as well as the state when it reaches the top of the hill. Additionally, the code will print 999 every time it gets cut off at 1000 episodes, regardless of if it is the 10th episode or not.

1: (Learning curves are attached. Curve3, curve5, and curve 7 correspond to fourier bases of orders 3, 5, and 7 respectively, and curvePoly corresponds to a 7th order polynomial basis function.) All 3 fourier bases showed similar behavior: lines of common values in the several hundreds, with a few points at 0 signifying not finishing the episode before 1000 time steps. I could not get the function to converge. The polynomial did not converge either, but showed much different behavior. It had a much higher rate of episodes that did not finish before 1000 time steps, but the about ⅔ that did were all extremely fast compared to the fourier with slightly under 100 time steps each.

The eligibility updates I am using are intended for a binary feature vector, while the one we use is continuous, which is why the code is not working. In order to have the function converge correctly, I would need to change my eligibility in the updates of the weights to work with continuous features. I could likely accomplish this somehow by keeping a history of the states and weights, but am unsure how the updates to the weights would be applied.

2: I could not get the surface plot function to work in time, and it is commented out near the bottom.

3: If gamma was less than 1, the function would value immediate reward slightly more than long term reward, as opposed to when it equals 1 future rewards have the same value as current/ short term rewards. Since these episodes are generally long, this would cause difficulties making the function converge, since it is not looking long term as much, and looking at the short term does not help much.

If we had rewards of 0 and 1, since we start with weights of very close to 0, delta would be 0 + (about 0) + (about 0), and our weights then wouldn’t change much, and it would take forever to get a small change to the value function. Adding a gamma of less than one would just further this problem, and it would take a long time to have non-zero weights.