The step I did and some of the descriptions of function is as follows
Steps:

1 Calculate $\pi(a\,|\,s)$for every action and s when theta is fixed. For this problem both state space and action space is small, so it is easy to derive its formula.
Note: here theta is the parameter of distribution of action, eg, with prob of theta to go left and therefore (1 - theta) to go right.

2 Minimize the cost function $\sum(w(s') - w(s) - \pi(a\,|\,s))^2$. For this problem, state space is small, so we optimize it directly. Regard each w(s) as a variable and obtain its minimizer just as taught in calculus.
Note: Here I use the minimization function in MATLAB, details are hidden

3. Plug the expression computed in 1,2 and plug in expression of policy gradient then compute it. In this way we get an approximation of policy gradient

4. Repeat 1-3 for 1000 times. Compute its mean as approximation of policy gradient, and compute the variance of these 1000 numbers as variance
Note: Theta is fixed all the time in 1-4, the step of alter theta has not come yet

5. Do similar things for baseline method and calculate its variance. Prove that our method is better thus an improvement.

Here are descriptions of these codes and some explanations, in case it helps:

Explanation:
1 Here the code is specified to solve the problems of the "Riverswim" of that paper, where there are 6 states and 2 actions at each state. If implementing other problems, we can make changes to the code accordingly, which is accessible.
2. The base function I choose to approximate w(s) is polynomial basis. We can try other options if necessary.

Descriptions:
Generate_Theta: Generate the initialized parameters with corresponding scale.

Generate_MC: Given Theta and length of track N, generate a markov chain. The function returns both the state paths and actions in order.

cal_f_a_s: Given state and the action made at that state, the function calculated gradient of log(f(a|s)). It will be used in the expression of cost function.

base_function: Given a vector x, the function calculates the power of the vector element wise from power 0 to n(n is an input), I.e. Its i th column is x.^(i - 1). This matrix will be used in the expression of cost function(Not used for this small example)

Optimumq.m: Given a column y, use the polynomial basis generated by (f(s') - f(s)) to approximate it. y will be input as gradient of log(f(a|s)) in 'compute_all_coeff' later. The output of the function is the optimum coefficient (Not used for this small example)

compute_all_coeff: Return a matrix containing all coefficients of w(s). It is a matrix because w(s) is a vector function, and the ith column of the outputs corresponds to the coefficients of its ith component, i.e. partial derivative w.r.t the ith component of Theta.(Not used for this small example)

Cost function: Compute the cost function and its gradient in preparation for optimization process.(used for this small example)

Fmincg: a optimization function I used from Andrew Ng. (used for this small example)

Policy: Compute policy gradient (used for this small example)

Here are all of the descriptions. Above all it can solve w(s) numerically. The next step I will do is to make sure it works indeed, if not then I will make revisions(cost function, Theta, base function, etc). Thanks so much!