To overcome policy freezing a new MC approach was used to estimate the policy. This mainly involved using exploration to overcome the effects of policy freezing with two main methods being exploring starts and soft policies.

Instead of starting with a specific state and action, exploring starts result in episodes starting from different positions in the state space by choosing a random state and action such that all state-action pairs have a probability greater than 0. This approach can be seen in the second line in Figure 1.0. By using this approach this causes the system to explore the whole space and when enough episodes are run, the algorithm will eventually realise that the random state-action pair selected is better than the existing policy, overwriting the suboptimal frozen policy. However, this requires randomly selecting the correct action for large state spaces with many suboptimalities, this approach is very inefficient and may take a long time to run.

Building on exploring starts, more exploration can be incorporated into the new MC approach by using soft policies. This is a type of policy that does not converge to a deterministic final function but instead retains an element of randomness when selecting actions. This allows the algorithm to explore and avoid freezing on suboptimal policies as it finds better ones to replace it with. An example of how this could be done is using ϵ-greedy soft policies. However, if this was the only policy used, the resulting optimal policy would have an ϵ-greedy mechanism embedded within it. This would be suboptimal as a portion of actions would be incorrect due to the random actions selected. Additionally, the rate of convergence of these algorithms is very slow, especially for long trajectories, as there are more and more possible paths leading to a high variance and fewer samples available. To overcome this the algorithm could use smarter exploration strategies such as UCB which would favor less explored areas.

To overcome the embedded ϵ-greedy mechanism, the algorithm makes use of two policies. An on-policy which is deterministic and is used after training and learning has been completed, and an off-policy which is what the algorithm uses to learn the target policy. Over many episodes, the off-policy can be used to estimate the on-policy as it will eventually converge.

IDK IF WE SHOULD INCLUDE CODE HERE