Algorithm 1 Learning Algorithm

```
Randomly initialize critic network $V_\pi^\phi(s_{it}, S_t)$ and actor network $a^\theta(s_{it}, S_t, q_t)$ with
parameters $\phi$ and $\theta$
Initialize state $S_t, \{s_{it}\}_{i \in I}$ according to user-supplied values/distributions
if PRETRAIN_CHOICE

    Using user-provided initial values for choices as labels, train the policy
    network to select those actions given initial state
    # This is just an attempt to initialize the policy network better than randomly
end
# Burn-in: Approach the stochastic steady state given the initial policy.
for t = 1:T_burn

    Guess values for equilibrium variables $\boldsymbol{q}_t$ (e.g. prices)
    while error < $\delta$
        Sample actions $a_{it} =$
        $a^\theta(\boldsymbol{s}_{it}, \boldsymbol{S}_t, \boldsymbol{q}_t)$ for all agents $i$
        Compute error $\leftarrow$ equilibrium_conditions$\left(\{\boldsymbol{a}_{it}, \boldsymbol{s}_{it}\}_{i \in I}, \boldsymbol{S}_t, \boldsymbol{q}_t\right)$
        (e.g. market clearing conditions)
    end
    Draw shocks $\{\varepsilon_{it}\}_{i \in I}, \varepsilon_t$
    Compute next state $\left(\{\boldsymbol{s}_{it+1}\}_{i \in I}, \boldsymbol{S}_{t+1}\right) =$
    $T\left(\{\boldsymbol{a}_{it}, \varepsilon_{it}, \boldsymbol{s}_{it}\}_{i \in I}, \boldsymbol{S}_t, \boldsymbol{q}_t, \varepsilon_t\right)$
end
# Estimate the value function given the initial policy
for t = 1:T_trainvalue
    Solve for equilibrium as above
    for j = 1:M
        Draw shocks $\{\varepsilon_{it}\}_{i \in I}, \varepsilon_t$
        Compute next state $\left(\left\{\boldsymbol{s}_{it+1}^j\right\}_{i \in I}, \boldsymbol{S}_{t+1}^j\right) =$
        $T\left(\{\boldsymbol{a}_{it}, \varepsilon_{it}, \boldsymbol{s}_{it}\}_{i \in I}, \boldsymbol{S}_t, \boldsymbol{q}_t, \varepsilon_t\right)$
        Compute for each $i \in I$ the TD target $y_{it}^j =$
        $r_i + \gamma V_\pi^\phi\left(s_{it+1}^j, S_{t+1}^j\right)$
    end
    Estimate the expected average TD error gradient $\nabla_\phi \delta_t^\phi =$
    $\nabla_\phi \frac{1}{N} \sum_{i=1}^N \left(\frac{1}{M} \sum_{j=1}^M y_{it}^j - V_\pi^\phi(s_{it}, S_t)\right)^2$
    Update $\phi$ according to the estimated gradient $\nabla_\phi \delta_t^\phi$
end
```

Algorithm 2

```
# Train the policy while moving toward the equilibrium steady state
and updating the value function
for episode = 1:M

    for t = 1:T_trainpolicy

        solve for equilibrium as above
        for j = 1:M

            Draw

        Estimate ∇_θ E[r_t + V(s_{t+1}, S_{t+1}) | S_t] ≈
        (1/N) Σ_{i∈I} [∇_θ (r_it + V_π^φ(s_{it+1}, S_t))]

            where ∇_θ (r_it + V_π^φ(s_{it+1}, S_t)) is computed directly using autodiff.
        Add this gradient to the policy gradient buffer.
        For discrete actions, update θ
        Compute for each i ∈ I the TD target y_it =
        r_i + γV_π^φ(s_{it+1}, S_{t+1})
        Estimate the expected TD error δ_t =
        E[δ_it | S_t] ≈ (1/N) Σ_{i∈I} (y_it − V_π^φ(s_it, S_t))^2
        Compute the gradient ∇_φ (1/N) Σ_{i∈I} (y_it − V_π^φ(s_it, S_t))^2 and add it to
        the TD error gradient buffer

    end
    Update θ according to the policy gradient buffer
    Update φ according to the TD error gradient buffer
    Update state so that the first state of the next episode is
    the last state from the previous episode

end
```