```
%%%%% TRIAL INITIATION [time = 0] %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
observe(s₀)                    %binning firing rates from the rat’s brain
calculate_max(Qₖ(s₀))             % for all k - use eqn. 5

% impossible to train network because Qₖ(s₁) does not exist

calculate(Qₖ(s₀))            % for all k - use eqn. 5
select(a₀)                  % use eqn. 11 - robot will be moved using IKO
calculate(e₀(s₀)ₖ)           % for all k - use eqn. 9
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%% FIRST UPDATE [time = 1] %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
observe(s₁)                      % binning firing rates from the rat’s brain
calculate_max(Qₖ(s₁))            % for all k - use eqn. 5

% now the training signal can be calculated for time = 0 because Qk(s_1) exists
for n = [x' y' z']
    calculate(d(n)₁)        % use eqn. 4a
end                         %   this uses CURRENT robot position and static target position
calculate(dg)               % use eqn. 4
calculate(r₁)               % use eqn. 3
calculate(delta₀)           % use eqn. 6

update(Qk(s₀))              % for all k - use eqn. 12
                            %   apply CURRENT reward to PRIOR state
% end of training

calculate(Qₖ(s₁))            % for all k - use eqn. 5
select(a₁)                  % use eqn. 11 - robot will be moved using IKO
calculate(e₁(s₁)ₖ)         % for all k - use eqn. 9

update(e₁(s₀)ₖ)            % for all k - use eqn. 10
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%% SECOND UPDATE [time = 2] %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
observe(s₂)                    % binning firing rates from the rat’s brain
calculate_max(Qₖ(s₂))             % for all k - use eqn. 5

% now the training signal can be calculated for time = 1 because Qₖ(s₂) exists
for n = [x' y' z']
    calculate(d(n)₂)        % use eqn. 4a
end                         %   this uses CURRENT robot position and static target position
calculate(dg)               % use eqn. 4
calculate(r₂)                % use eqn. 3
calculate(delta₁)            % use eqn. 6

update(Qₖ(s₁))               % for all k - use eqn. 12
update(Qₖ(s₀))               % for all k - use eqn. 12
                            %   apply CURRENT reward to PRIOR states
% end of training

calculate(Qₖ(s₂))          % for all k - use eqn. 5
select(a₂)                 % use eqn. 11 - robot will be moved using IKO
calculate(e₂(s₂)ₖ)         % for all k - use eqn. 9

update(e₂(s₁)ₖ)            % for all k - use eqn. 10
update(e₂(s₀)ₖ)            % for all k - use eqn. 10
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

$$r_t = -0.01 + \exp\left(-r_s \cdot \left(d_{thres} - dg\right)\right) \qquad (3)$$

$$dg = \exp\left(-\tfrac{1}{2}\left(\frac{d(x')^2}{0.001} + \frac{d(y')^2}{0.003} + \frac{d(z')^2}{0.0177}\right)\right) \qquad (4)$$

$$d(n)_t = \sqrt{\left(lever(n) - robot(n)_t\right)^2} \qquad (4a)$$

$$\begin{aligned} Q_k(s_t) &= \sum_j \left(\tanh\left(\sum_i s_{i,t} w_{ij}\right)\right) w_{jk} \\ &= \sum_j net_j\left(s_t\right) \cdot w_{jk} \end{aligned} \qquad (5)$$

$$\delta_t = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \qquad (6)$$

$$e_t(s_t)_k = \begin{cases} 1 & a_t = k \\ 0 & else \end{cases} \qquad (9)$$

$$e_{t+n}(s_t)_k = \begin{cases} (\gamma\lambda)^n e(s_t)_k & a_{t-n-1} = \arg\max_k Q_k(s_{t-n-1}) \\ 0 & else \end{cases} . \qquad (10)$$

$$a_t = \begin{cases} \arg\max_k \left\{Q_k(s_t)\right\} & p(1-\varepsilon) \\ rand \neq \arg\max & p(\varepsilon) \end{cases} \qquad (11)$$

$$\partial J(t) / \partial Q_k(s_t) = -\sum_{n=0}^{T} e_{t+n}(s_t)_k \cdot \delta_{t+n} \qquad (12)$$