Q-Learning for World Grid Navigation

EE5904/ME5404 Part II: Project 2

Report due on **April 23, 2021**

GA: Wong Pooi Mun

wong.pooimun@u.nus.edu

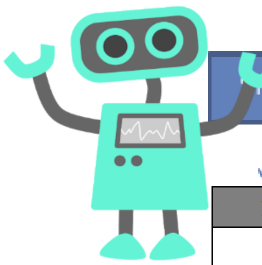
Outline

Project Description

- Task
- State Transition
- Reward Function
- Learning

Project Description: Task

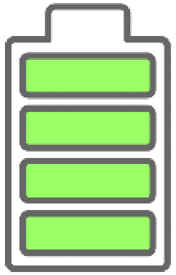
Using only **Q-learning** with ϵ-greedy exploration, the robot is to move from the initial state (s = 1) to the goal state (s = 100) with the maximum total reward of the trip.



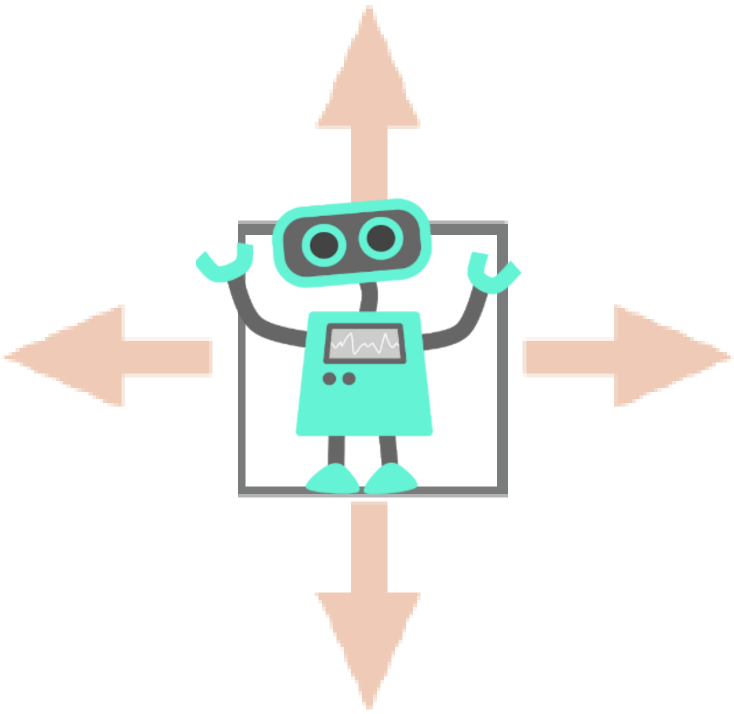
START

1	11	...	...	...	...	...	...	...	91
2	12	...	...	...	...	...	...	...	92
3	13							.	93
4	.							.	94
5	.							.	95
6	.							.	96
7	.							.	97
8	.							.	98
9	.							89	99
10	...	...	...	...	...	...	...	90	100

STOP



Project Description: State Transition



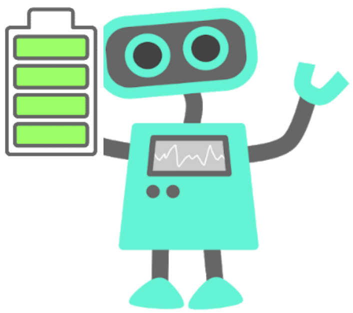
Deterministic Model

You can actually use dynamic programming to find the optimal policy

Project Description: Reward Function

Task 1 → “reward” in “task1.mat”

Task 2 → “qevalreward” in “~~qeval.mat~~”



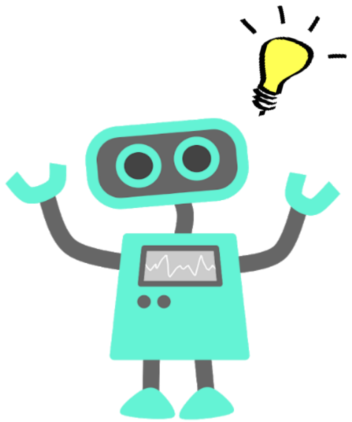
Reward Matrix: 100 x 4

$$\begin{bmatrix} r_{1\,1} & \cdots & r_{1\,4} \\ \vdots & \ddots & \vdots \\ r_{100\,1} & \cdots & r_{100\,4} \end{bmatrix}$$

Actions

States

Project Description: Learning



- The robot learns in 1 run.
- One run consists of the *N* trials.
- Each run starts with a set of initial values of the Q-function (100 x 4 matrix).
- Each trial starts when the robot moves from state 1.
- Each trial ends when the robot reaches state 100.
- The Q values are passed to the next trial.
- Each run ends when the Q values converge to the optimal values.

To skip Recap, go to **05:17** >>

Recap

- Reward
- Q Function
- Optimal Policy
- Model Free Value Iteration
- Greedy Exploration

Recap: Reward

Total reward for a state transition is given by:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

where,

- R_t determines present value of future rewards

Rewards received k steps in the future is discounted by factor γ^{k-1}.

Small γ → Focus more on intermediate rewards for next the few steps.

Large γ → Take into account future rewards more strongly.

Recap: Q Function

'Worth' of actions at different states is given by:

$$Q^{\pi}: S \times A \rightarrow \boxed{\mathcal{R}}$$

$$Q^{\pi}(s,a) = E^{\pi}[R_t | s_t = s] \rightarrow R_t | s_t = s$$

Deterministic Transition

Expected return from taking action a at sate s at time step t by following action π

Recap: Optimal Policy

Optimal policy is the state transitions that maximize the Q-values.

Slide 164-166

$$Q^\pi(s,a) = E^\pi\left[r_{t+1}\right] + E^\pi\left[\gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_t = s\right]$$

Values of Q-function are optimal if they are greater or equal to that of all other policies for all (s,a) pairs, i.e.,

$$Q^*(s,a) = \max_\pi Q^\pi(s,a)$$

Greedy policy

At each s, select a that yields the largest value for the Q-function. When multiple choices are available, such a can be picked randomly

Optimal policy: $\pi^*(s) \in \arg\max_a Q^*(s,a)$

Recap: Model-Free Value Iteration

When state transition model is **unknown**, the Q-function can be estimated via iterative update rule by using the reward received from observed state transitions.

$$Q_{k+1}(s_k, a_k) = Q_k(s_k, a_k) + \alpha_k \left(\underbrace{\overset{\text{Reward of action } a \text{ at state } s}{\boxed{r_{k+1}}} + \gamma \max_{a'} Q_k\left(s_{k+1}, a'\right)}_{\text{Estimate of } Q^*(s_k, a_k)} - Q_k(s_k, a_k) \right)$$

Exploitation:

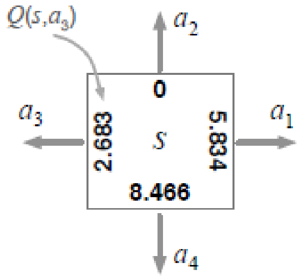
Use greedy policy to select currently known best action

$$a_{k+1} = \max_{a'} Q_k(s_{k+1}, a')$$

Exploration:

Try action other than current known best action

$$a_{k+1} \neq \max_{a'} Q_k(s_{k+1}, a')$$



Exploitation: Take a_4

Exploration: Take a_1, a_2, a_3

Recap: ϵ-greedy exploration

Initialize parameters

Input: Discount factor γ; exploration probability ϵ_k; learning rate α_k

- Initialize Q-function, e.g., $Q_0 \leftarrow 0$
- Determine the initial state s_0
- For time step k, select action a_k according to:

Select Action

$$a_k = \begin{cases} a \in \arg\max_{\hat{a}} Q_k(s_k, \hat{a}) & \text{with probability } 1 - \epsilon_k \\ \text{an action uniformly randomly selected from all other actions available at state } s_k & \text{with probability } \epsilon_k \end{cases}$$

Exploitation with probability $1 - \epsilon_k$

Exploration with probability ϵ_k

Apply Action

- Apply action a_k, receive reward r_{k+1}, then observe next state s_{k+1}

Update Q-value

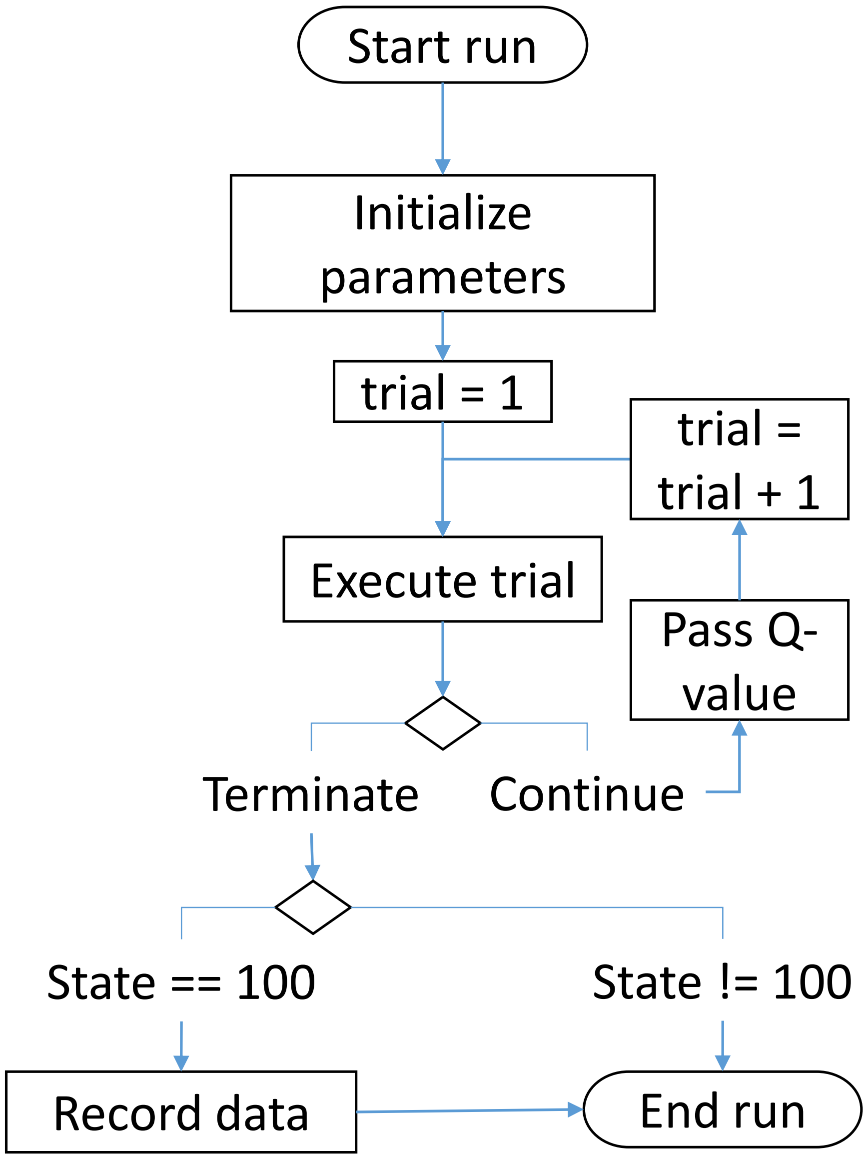
- Update Q-function with:

$$Q_{k+1}(s_k, a_k) = Q_k(s_k, a_k) + \alpha_k \left(r_{k+1} + \gamma \max_{a'} Q_k(s_{k+1}, a') - Q_k(s_k, a_k) \right)$$

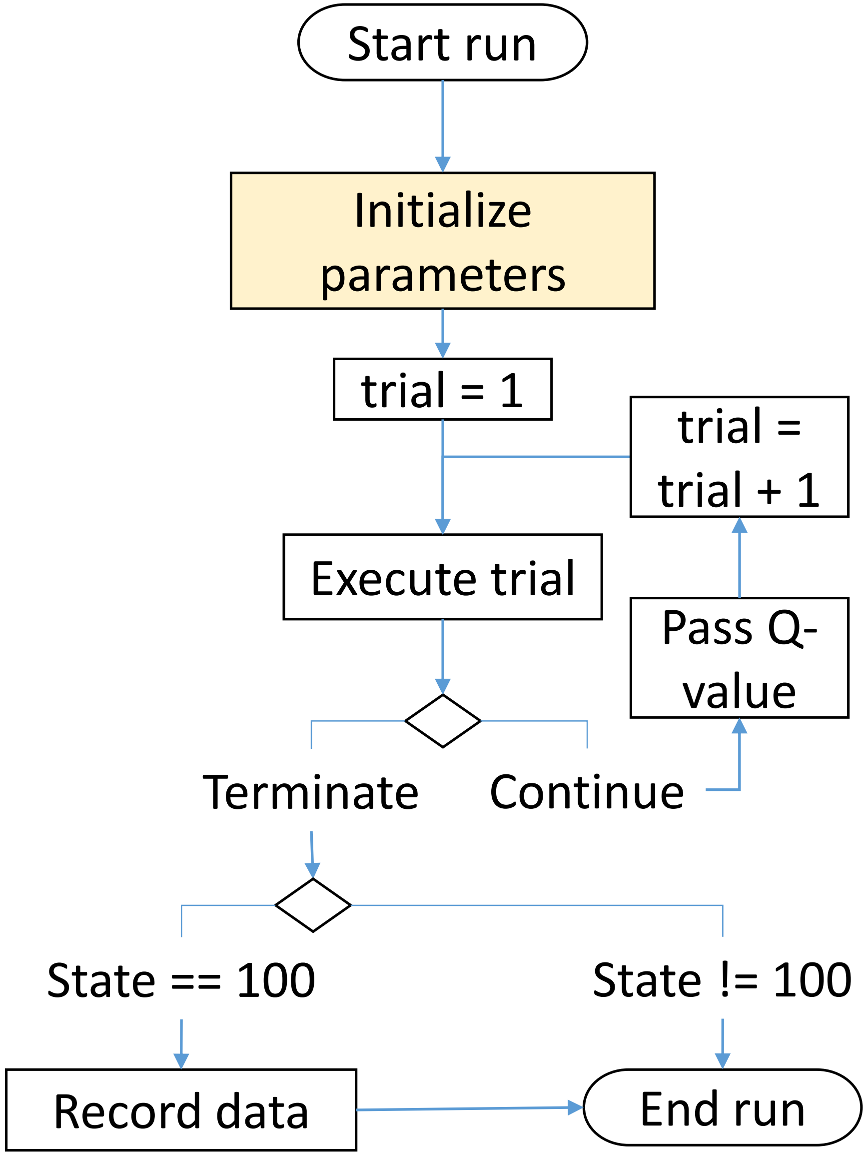
- Set $k = k + 1$ and repeat for-loop for the next time step

Implementation

Implementation



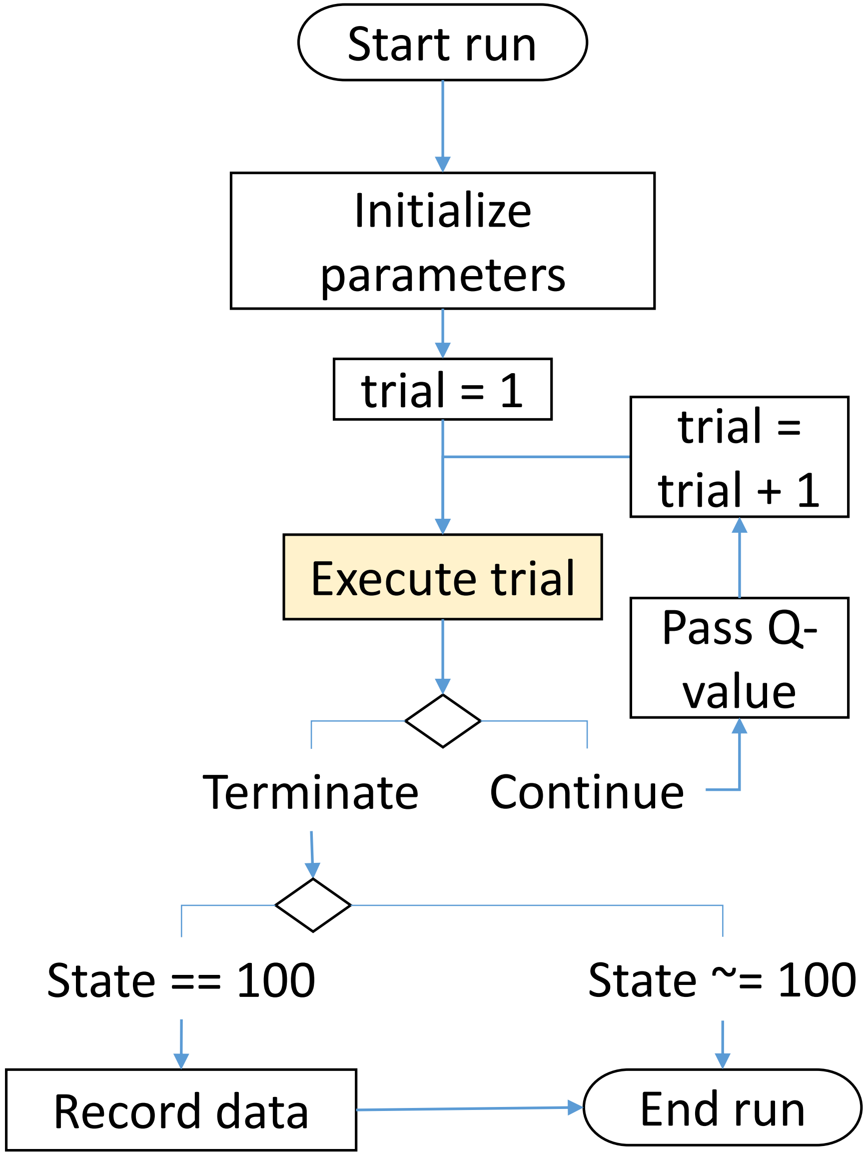
Implementation

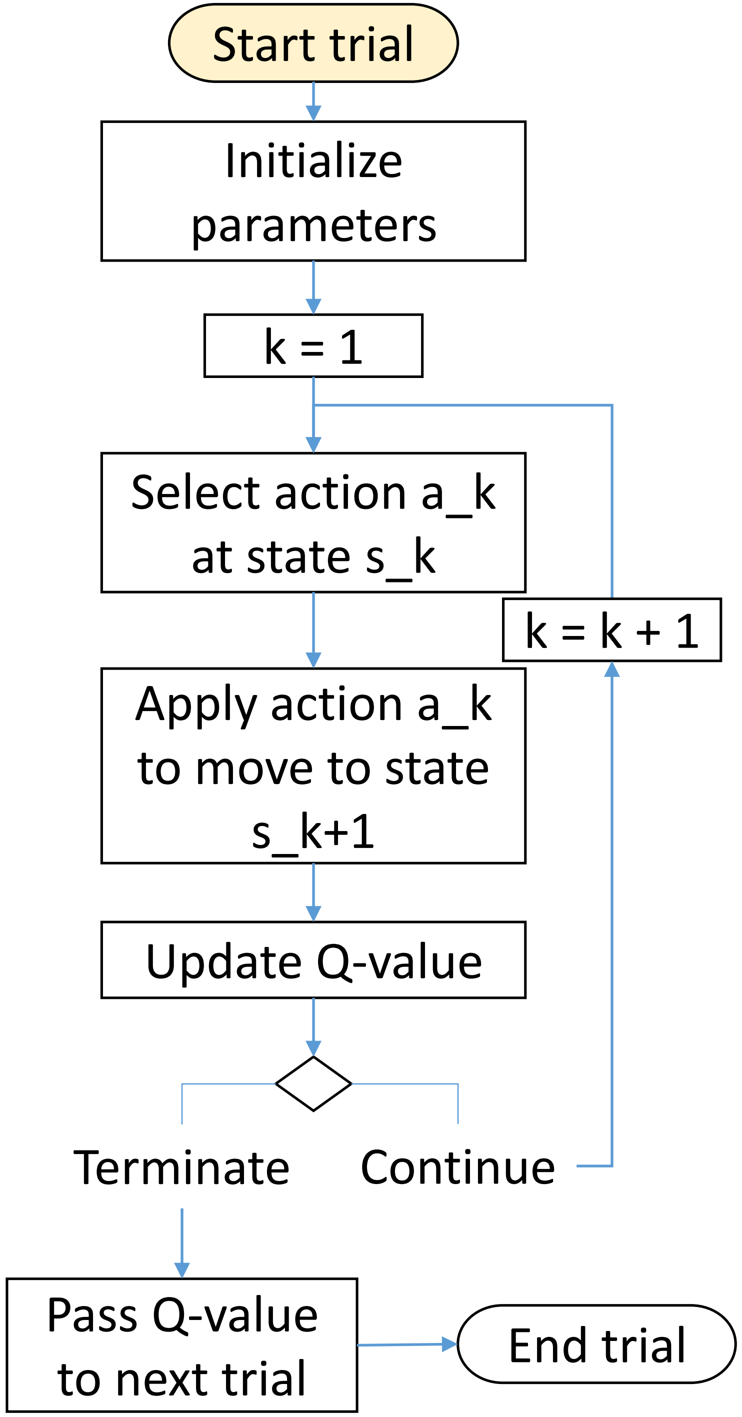


Parameters:

- Initial Q-function $Q_1 \leftarrow 0$ >> Optional
- Trial $\text{trial} = 1$
- threshold of error of Q-values between trials

Implementation

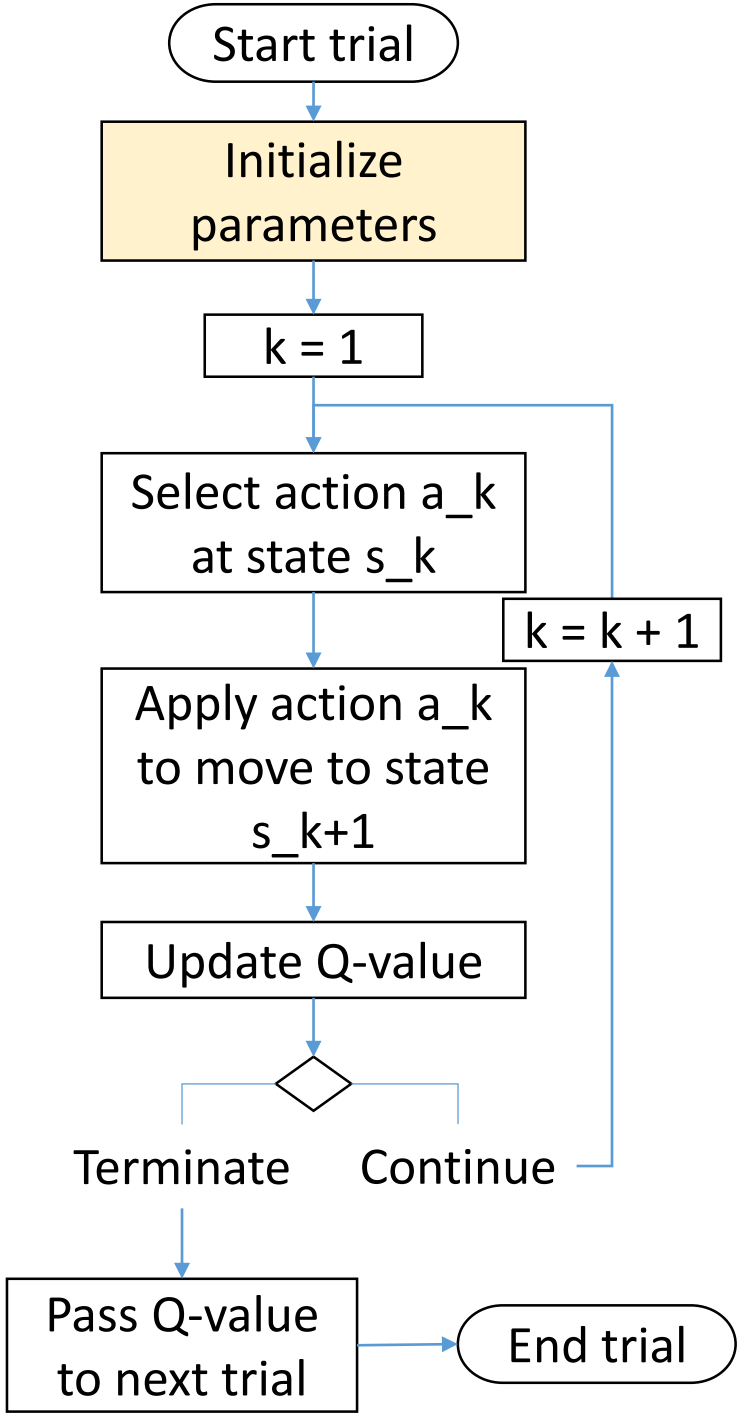




Implementation

Parameters:

- Discount factor γ
- Exploration probability ϵ_k
- Initial Q-function Q_1 from previous trial (if any)
- Learning rate $\alpha_k = \epsilon_k$
- Initial state $s_1 = 1$
- Time step $k = 1$



Implementation

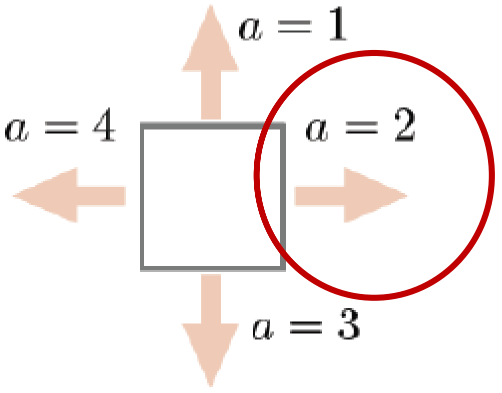
Example:

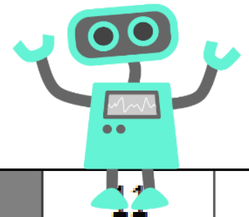
For $k = 3$,

- $\epsilon = 1 - \frac{1}{k}$
- $s_3 = 11$
- Current best action is 2.
- Exploitation: $a_3 = 2$ each has $1 - \epsilon = \frac{1}{3}$ probability to be selected
- Exploration: $a_3 = 3, 4$ each has $\epsilon = \frac{2}{(2)3}$ probability to be selected
- $\mathrm{a}_3 = 1$ cannot be selected due to the boundary

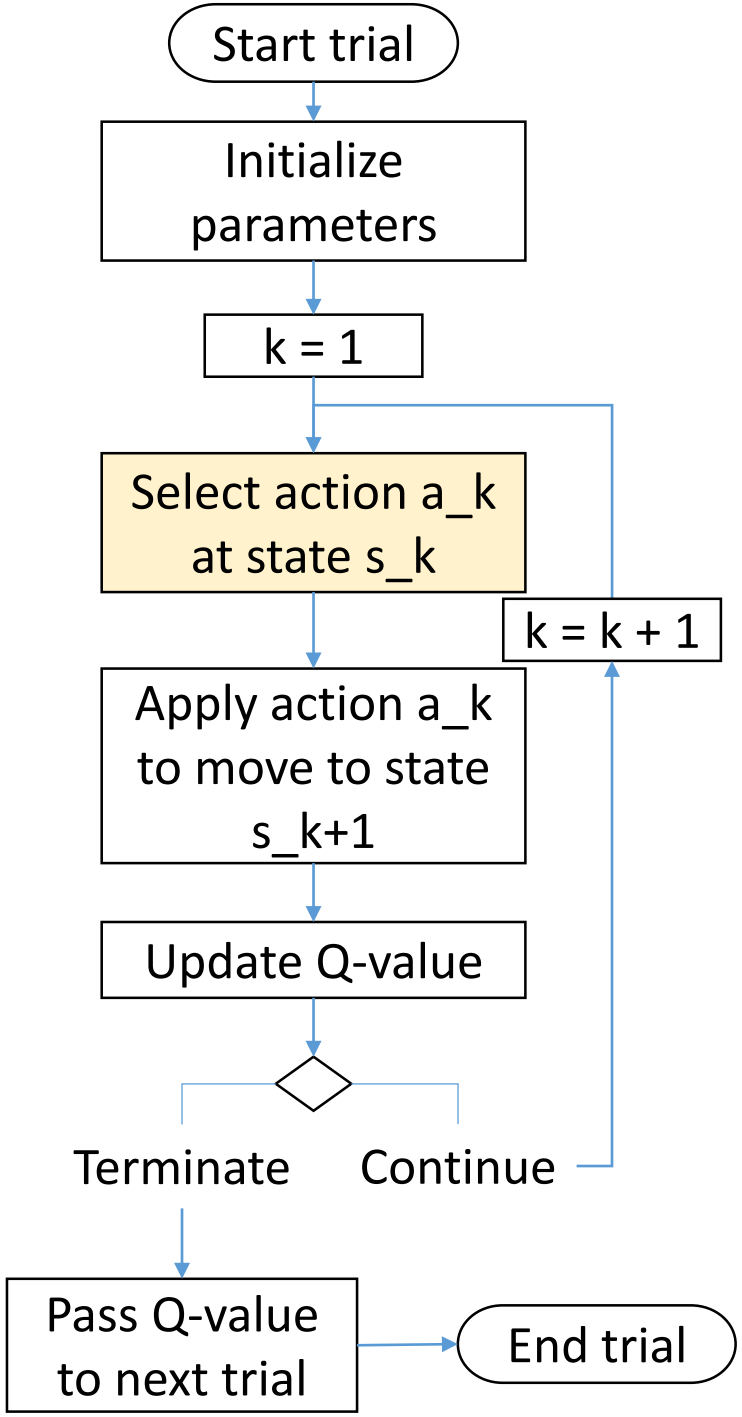
$$a_k = \begin{cases} a \in \arg\max_{\hat{a}} Q_k(s_k, \hat{a}) & \text{with probability } 1 - \epsilon_k \\ \text{an action uniformly randomly selected from all other actions available at state } s_k & \text{with probability } \epsilon_k \end{cases}$$

$k = 3$





1	11	...
2	12	...
3	13	
4	.	



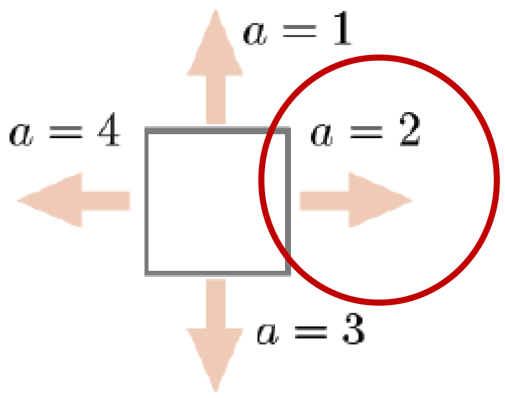
Implementation

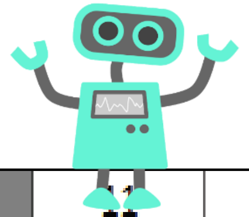
Example (continue):

For $k = 3$,

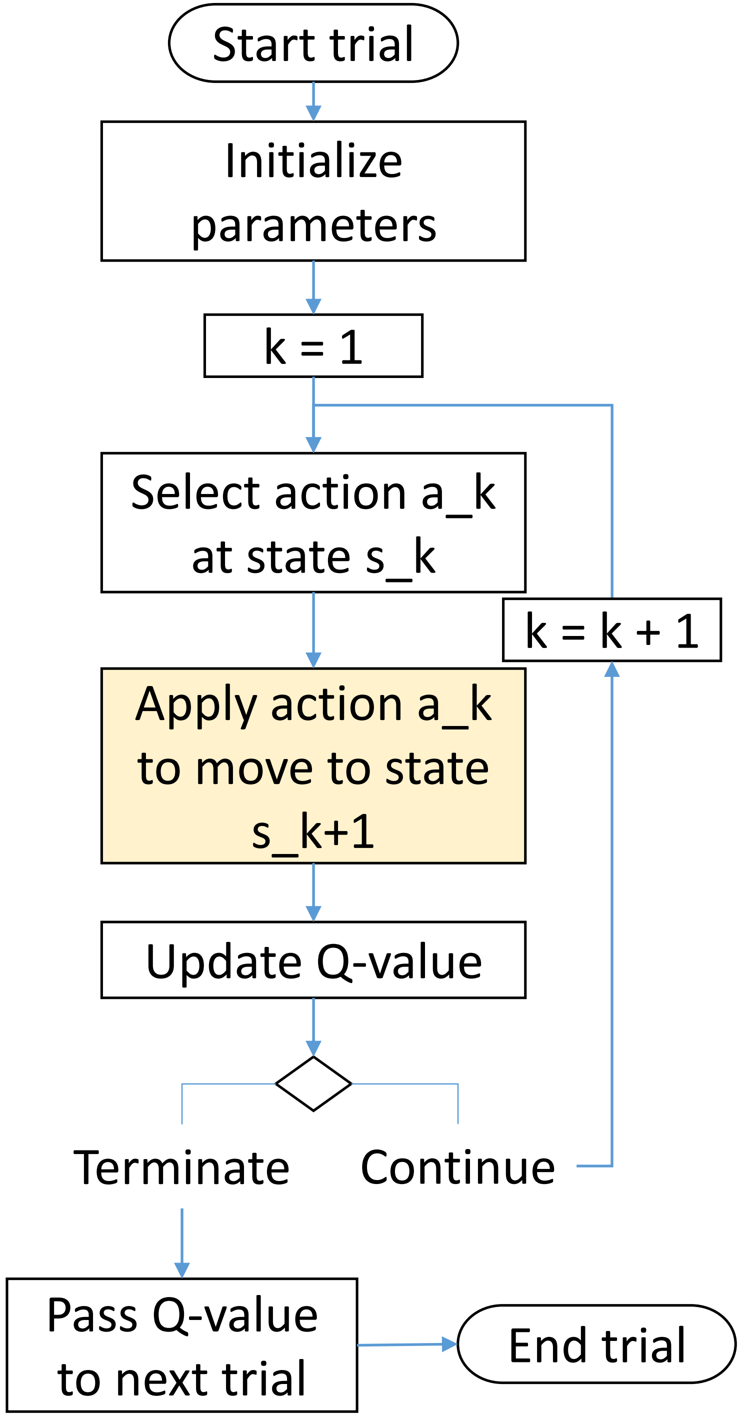
- $s_3 = 11$
- Selected action is $a_3 = 2$.
- $s_4 = 21$

$k = 3$





1	11	...
2	12	...
3	13	
4	.	

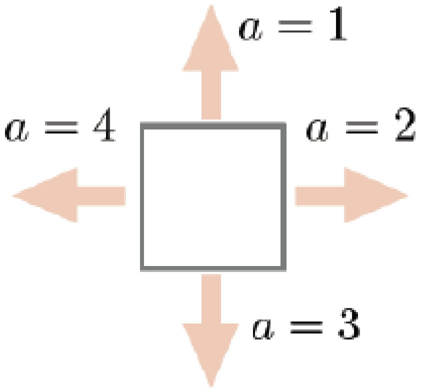


Implementation

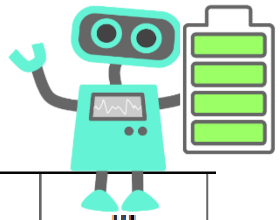
Example (continue):

For $k = 3$,

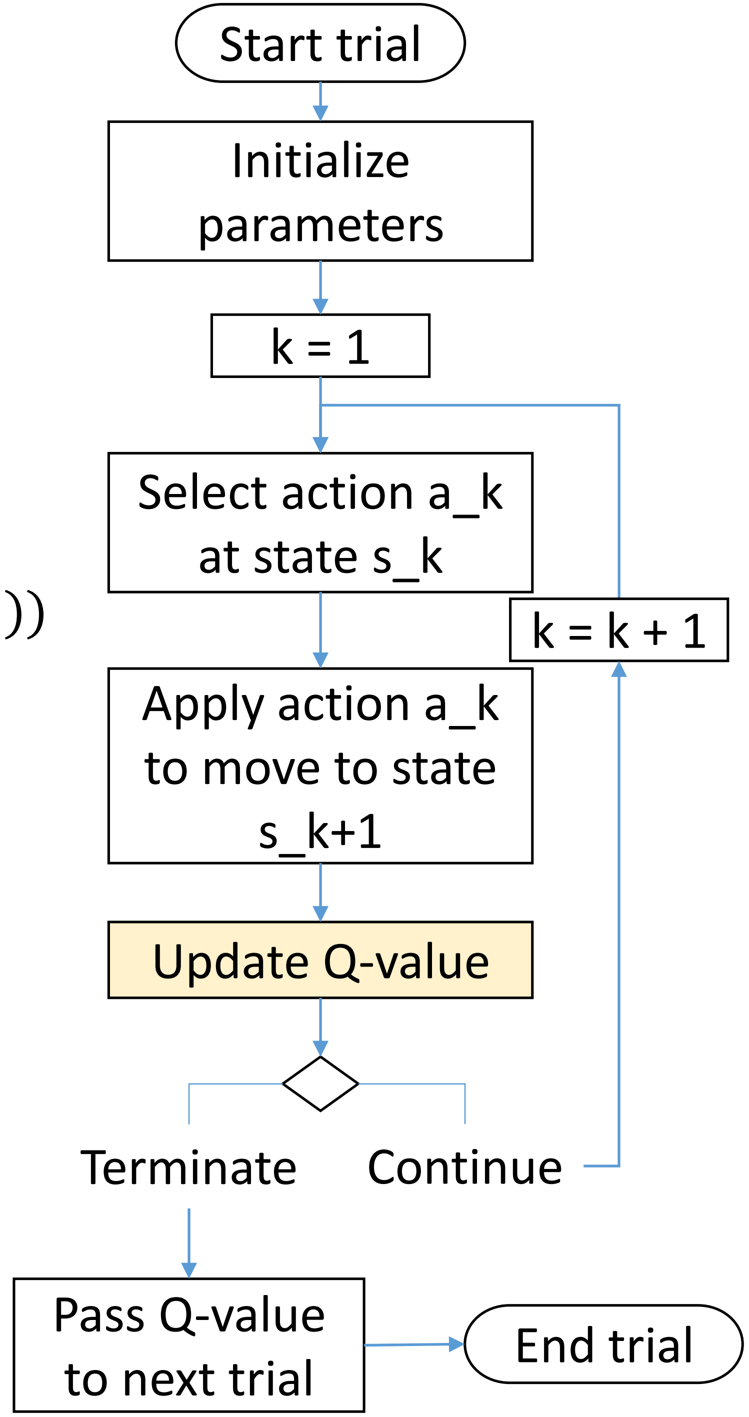
- Receive reward $r_{11\ 2}$
- $Q_4(11,2)$
  $= Q_3(11,2) + \alpha_3(\boldsymbol{reward}(11,2) + \gamma * max\big(Q_3(21,:)\big) - Q_3(11,2))$



$k = 4$



1	11	...
2	12	...
3	13	
4	.	



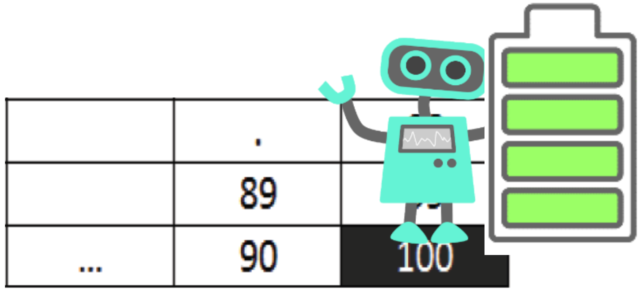
Implementation

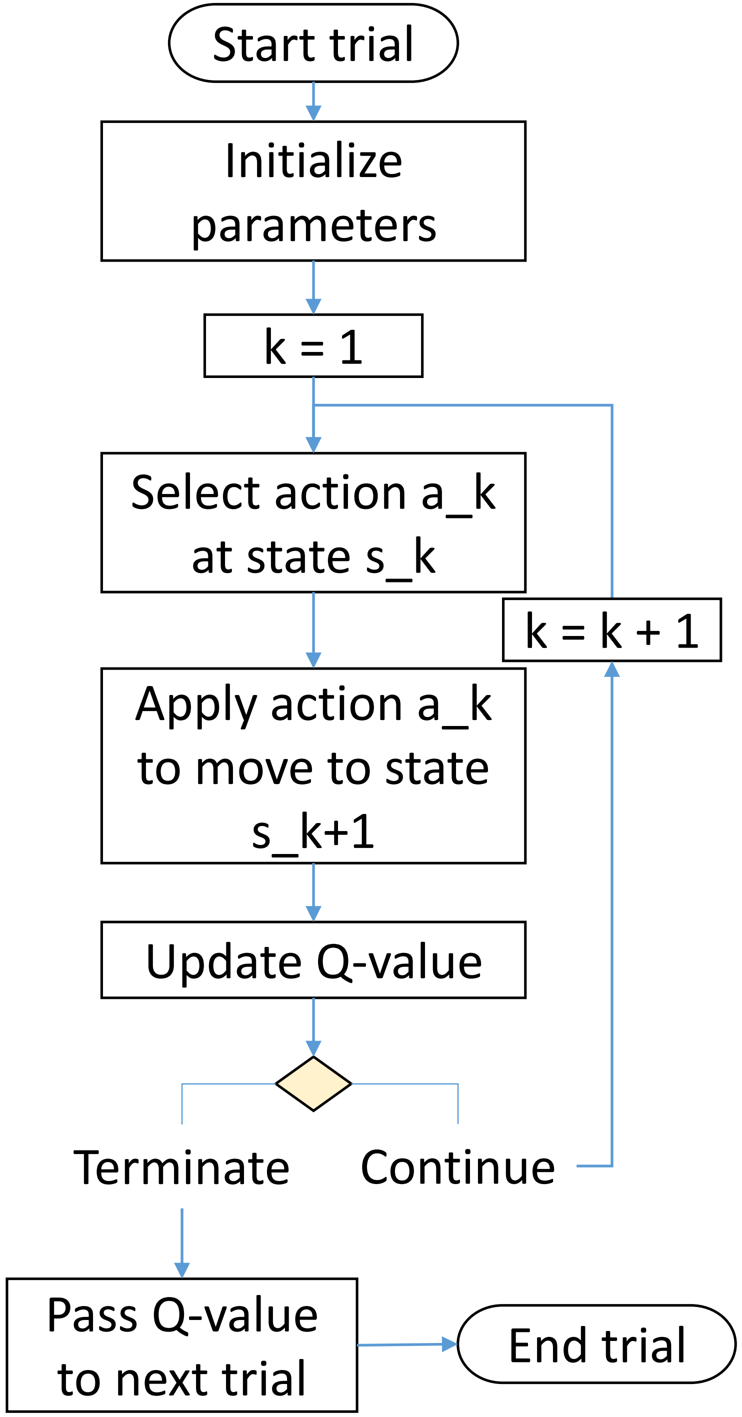
Termination condition for each trial:

- Robot reaches goal state $s_k = 100$ >> Ideal Case
- $\alpha_k < 0.005$ >> Optional
- Maximum number of time step $k_{\max}$ is reached

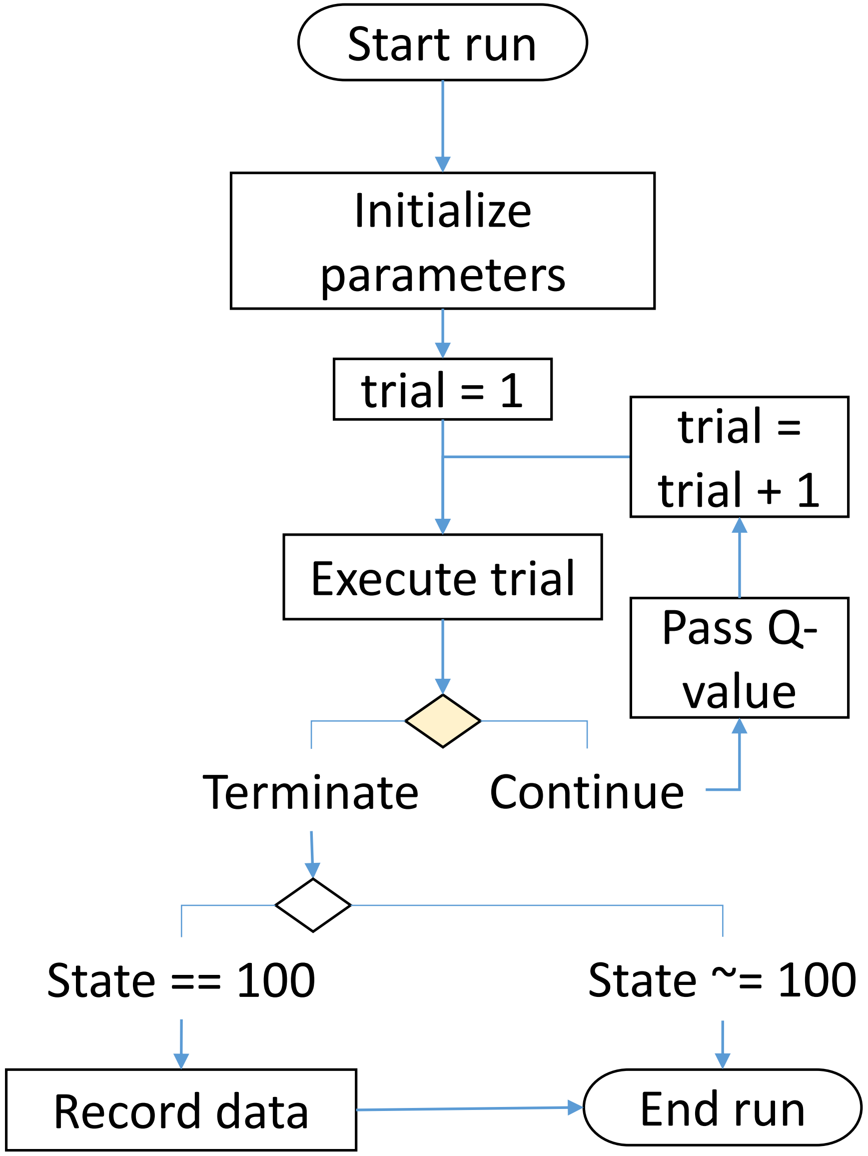
Continuation condition for each trial:

- Otherwise





Implementation



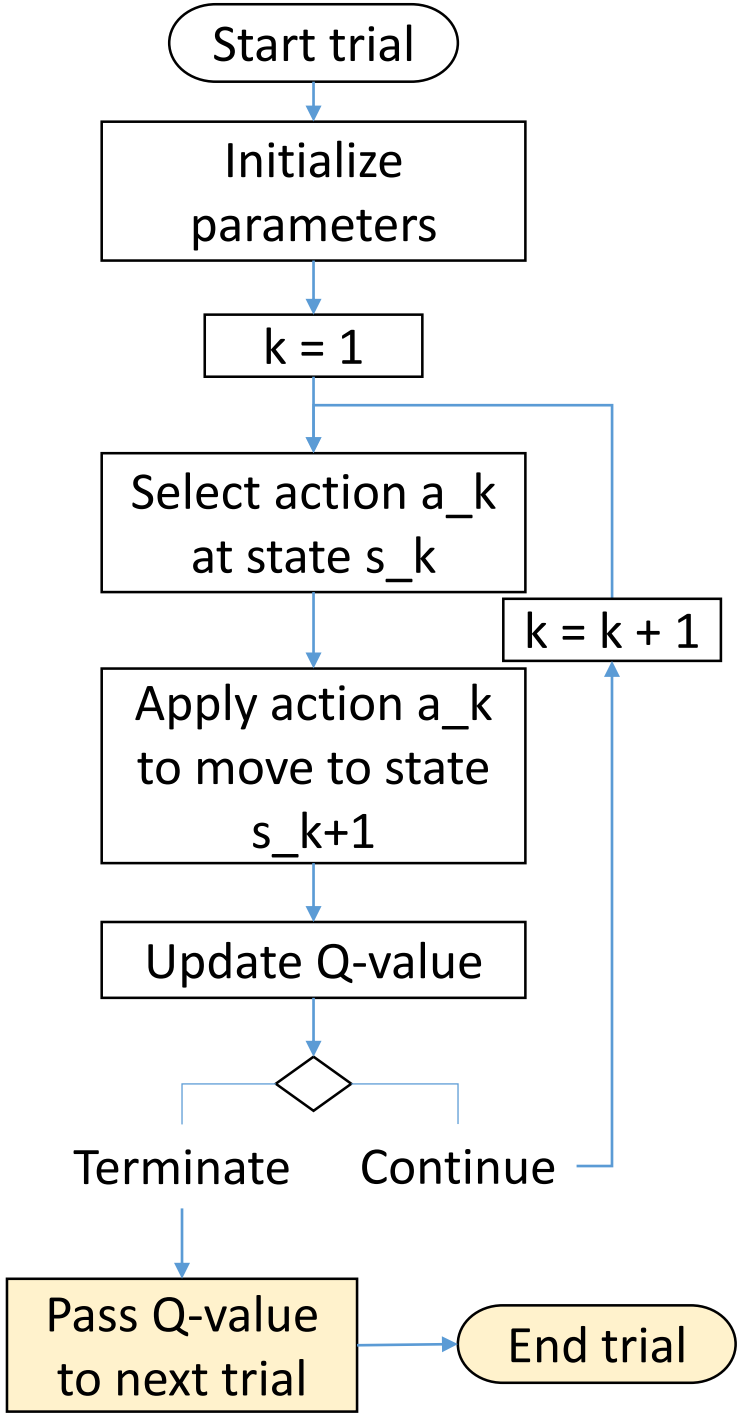
Initial Q-values of next trial is the optimal policy of this trial.

Termination condition for each run:

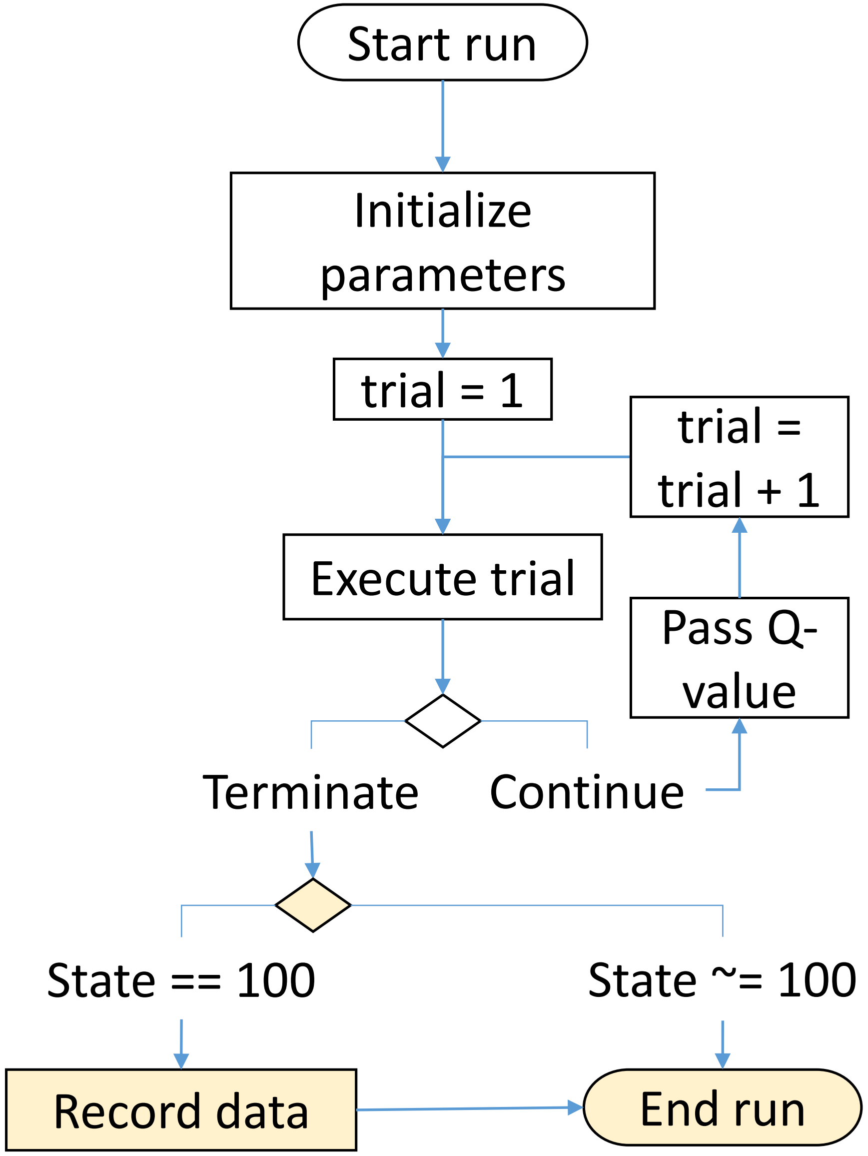
- Q-function converged to the optimal values >> Ideal Case
- Maximum number of trials $trial_{max}$ is reached

Continuation condition for each trial:

- Otherwise



Implementation



Record:

- Q values
- Execution time

Task 1

- What to do?
- Table 1
- Program Output
- Report
- Assessment

Task 1: What to do?

1. Implement Q-learning algorithm in MATLAB.
2. Reward function is in task1.mat.
3. Discount factor γ and exploration probability ϵ_k are given in Table 1.
4. For each set of parameter values,
   - Run 10 runs ($trial_{max} = 3000$ each).
5. Complete report.

TABLE I

PARAMETER VALUES AND PERFORMANCE OF Q-LEARNING

ϵ_k, α_k	No. of goal-reached runs		Execution time (sec.)	
	$\gamma = 0.5$	$\gamma = 0.9$	$\gamma = 0.5$	$\gamma = 0.9$
$\frac{1}{k}$	?	?	?	?
$\frac{100}{100+k}$	?	?	?	?
$\frac{1+log(k)}{k}$	?	?	?	?
$\frac{1+5log(k)}{k}$	?	?	?	?

Task 1: Table 1

For each parameter set

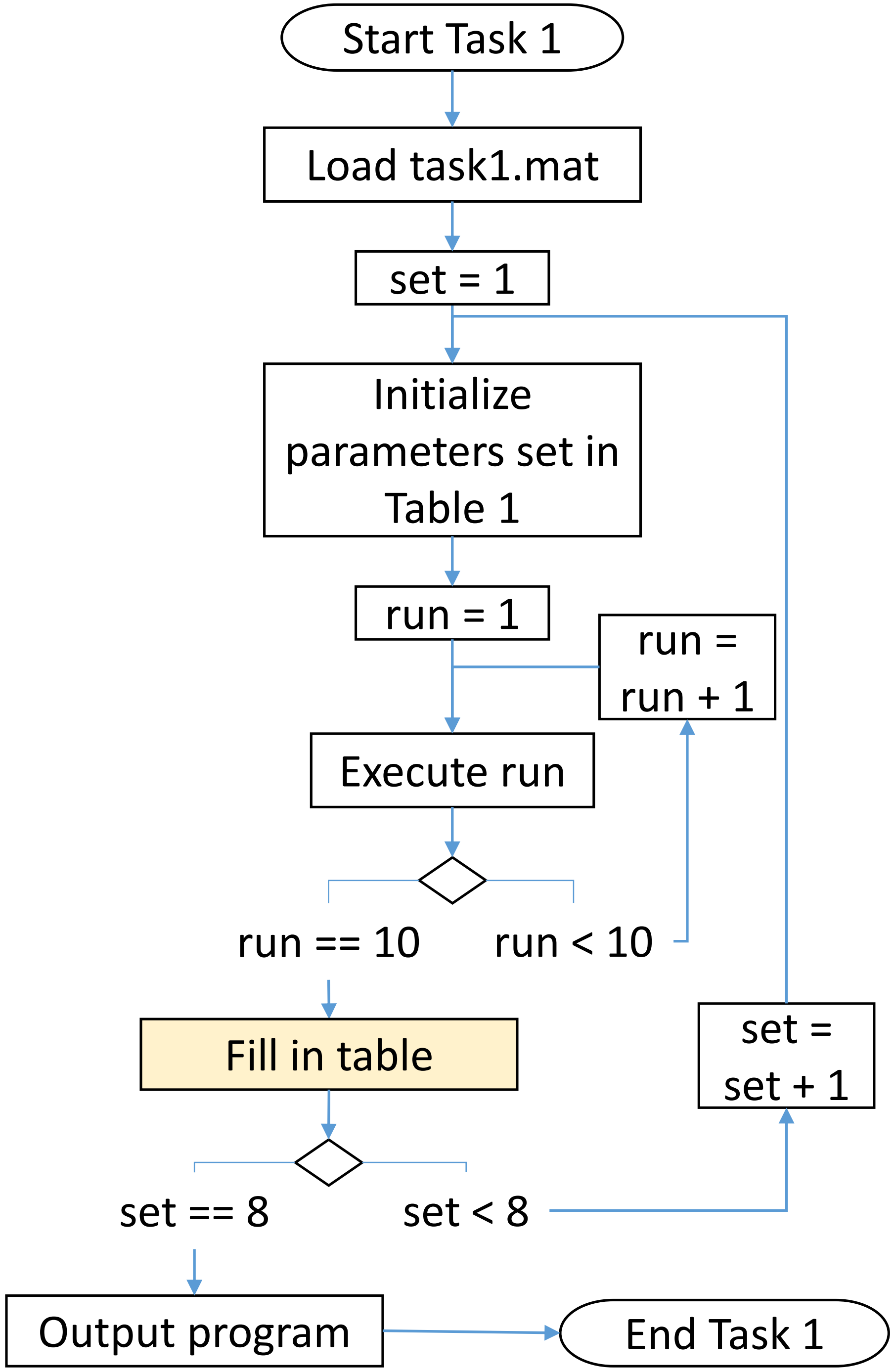
Number of goal reaching runs:
Out of 10 runs, count the runs that the robot ends at state 100.

Execution time:
Average the recorded execution time for those goal reaching runs.

TABLE I

PARAMETER VALUES AND PERFORMANCE OF Q-LEARNING

ϵ_k, α_k	No. of goal-reached runs		Execution time (sec.)	
	$\gamma = 0.5$	$\gamma = 0.9$	$\gamma = 0.5$	$\gamma = 0.9$
$\frac{1}{k}$	?	?	?	?
$\frac{100}{100+k}$	?	?	?	?
$\frac{1+log(k)}{k}$	?	?	?	?
$\frac{1+5log(k)}{k}$	?	?	?	?



Task 1: Program Output

For all parameter sets

Optimal policy:

Use the Q_{final} to extract the optimal path with greedy policy:

$$\pi^*(s) \in \arg\max_a Q^*(s,a)$$

Output the state transition in a single column matrix.

Total reward:

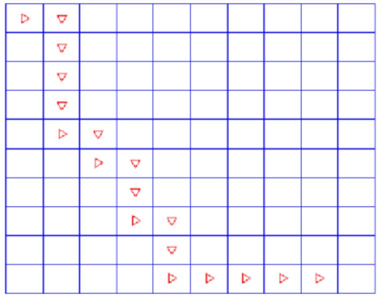
Substitute the *r* with the rewards from task1.mat that corresponds to the optimal policy.

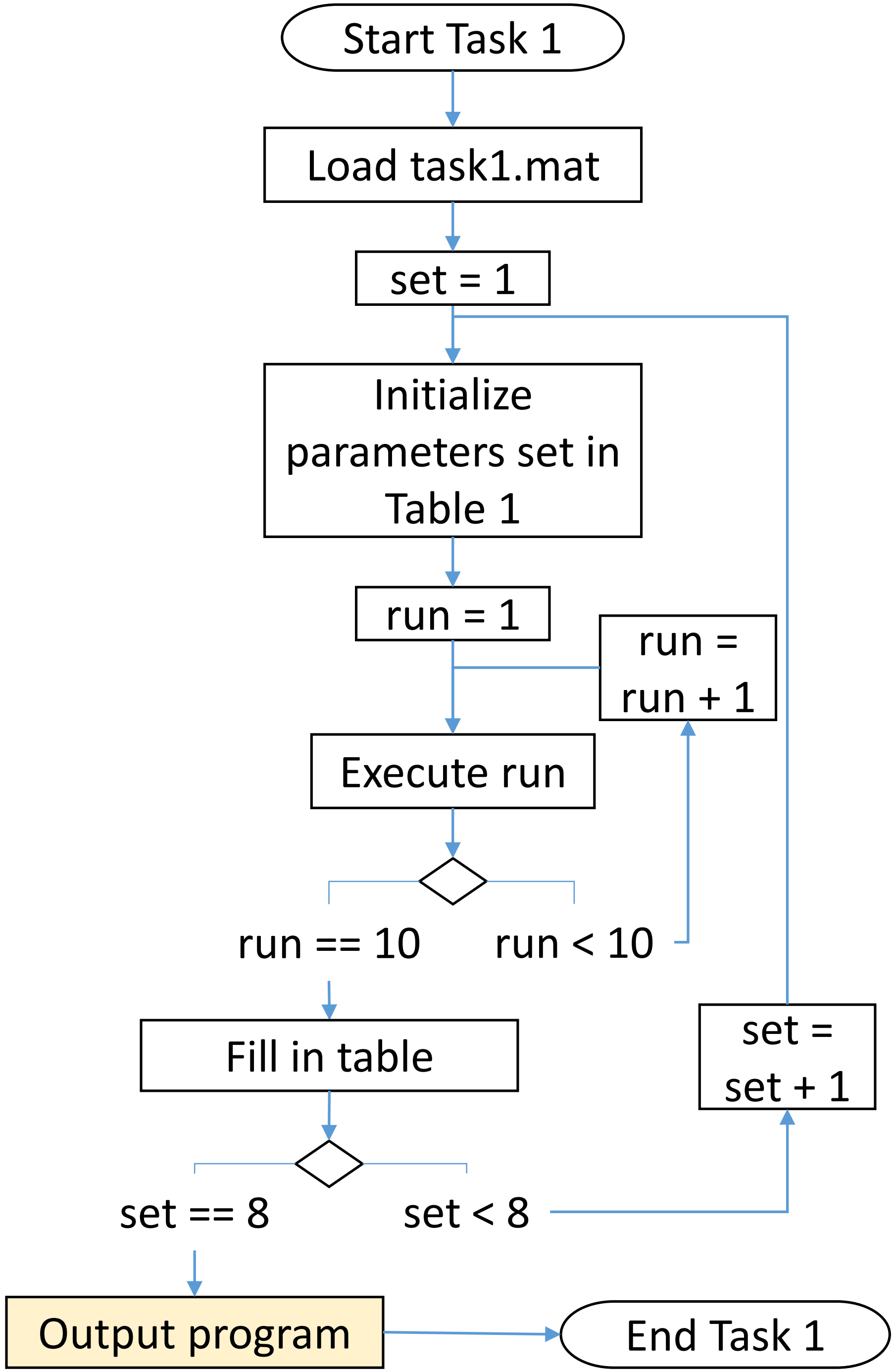
$$R_t = \sum_{k=1}^{\infty} \gamma^k r_{t+k+1}$$

Trajectory plot:

Plot the optimal policy using arrows.

Show the grid world of the 100 states.





Task 1: Report

- Table 1
- Output of the program
- Comments on the results (with proof)

TABLE I

PARAMETER VALUES AND PERFORMANCE OF Q-LEARNING

ϵ_k, α_k	No. of goal-reached runs		Execution time (sec.)	
	$\gamma = 0.5$	$\gamma = 0.9$	$\gamma = 0.5$	$\gamma = 0.9$
$\frac{1}{k}$	?	?	?	?
$\frac{100}{100+k}$	?	?	?	?
$\frac{1+log(k)}{k}$	?	?	?	?
$\frac{1+5log(k)}{k}$	?	?	?	?

Task 1: Assessment

Report

Task 2

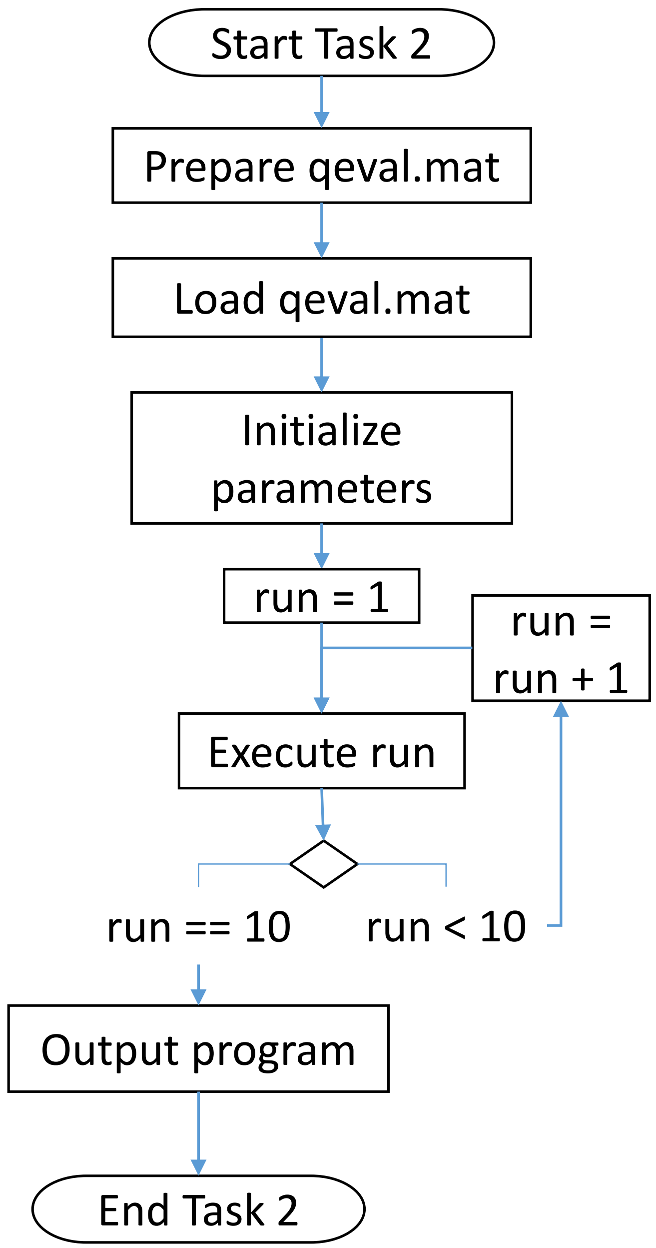
- What to do?
- Program Output
- Report
- Assessment

Task 2: What to do?

What to do?

1. Implement Q-learning algorithm in MATLAB.
2. Come up with your own reward function.
    - 100 x 4 matrix
    - Name the matrix as **qevalreward**
    - Save/load it as **qeval.mat**
3. Decide on your discount factor γ and exploration probability ϵ_k.
4. Complete report.

Need to deal with unknown rewards



Task 2: Program Output

Optimal policy:

Use the Q_{final} to extract the optimal path with greedy policy:

$$\pi^*(s) \in \arg\max_a Q^*(s,a)$$

Output the state transition in a single column matrix.

Total reward:

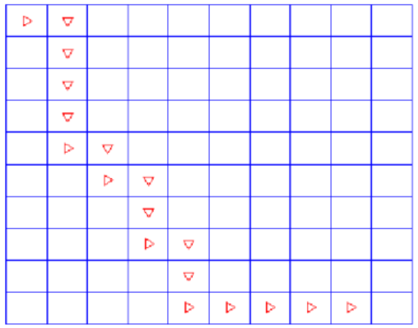
Substitute the *r* with the rewards from qeval.mat that corresponds to the optimal policy.

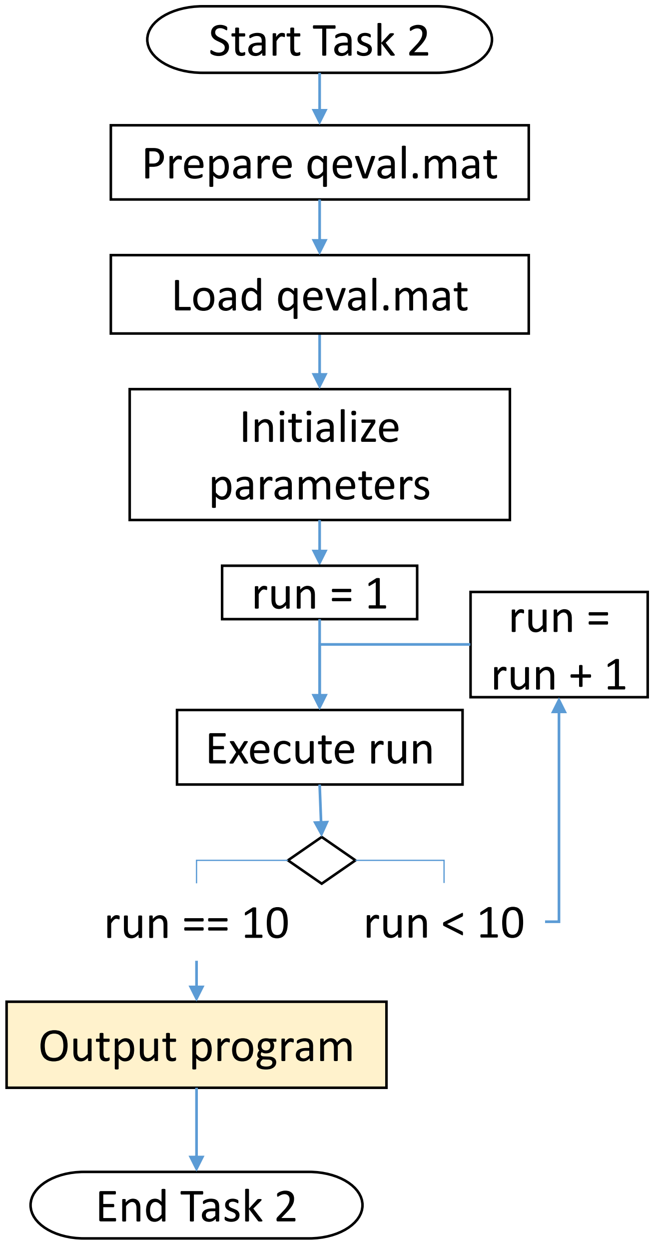
$$R_t = \sum_{k=1}^{\infty} \gamma^k r_{t+k+1}$$

Trajectory plot:

Plot the optimal policy using arrows.

Show the grid world of the 100 states.





Task 2: Report

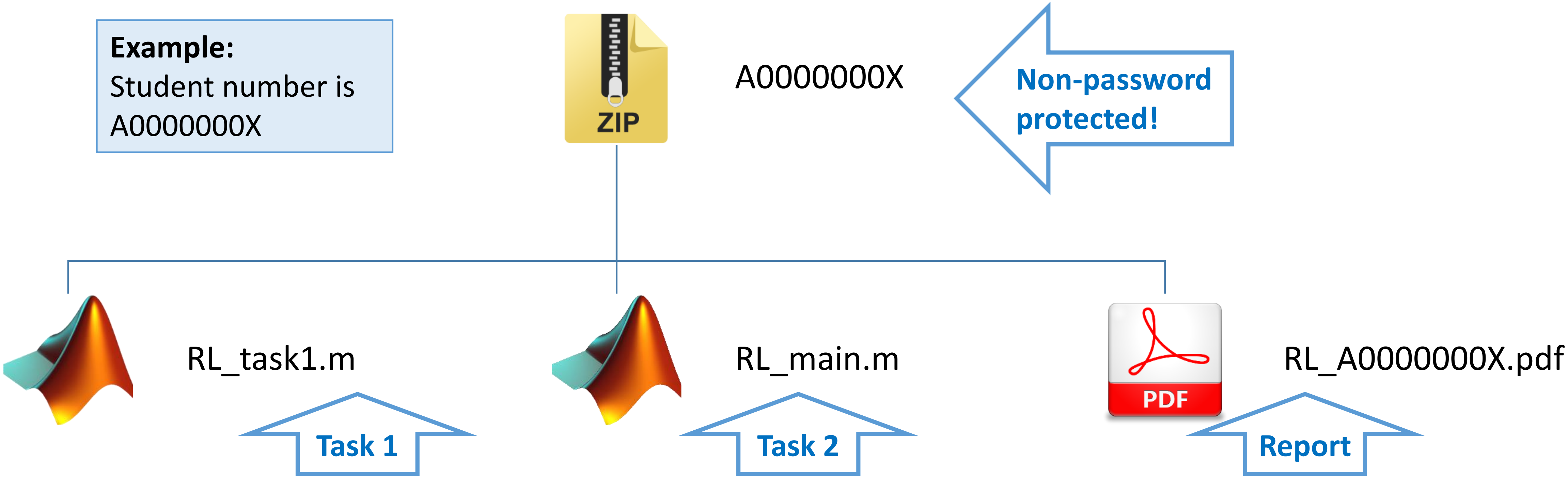
- Justify the selected parameters (with proof)

Task 2: Assessment

Report
Code execution
Execution time
Output from code

Submission Details

Submission Details



MATLAB Crash Course

- Reward
- Matrix
- Data
- Plot

MATLAB Crash Course

MATRIX	
Create 2 x 3 matrix	[1 2 3; 4 5 6]
Create 4 x 3 matrix of zeros	zeros(4, 3)
Find number of rows and columns of matrix A	size(A)
Get element at 1st row and 1st column of matrix A	A(1, 1)

REWARD	
Load ‘task1.mat’	load task1.mat
Create ‘qeval.mat’	save(‘qeval.mat’, ‘qevalreward’)

DATA	
Find the time taken of a block of code	tic % block of code toc
Display string with variable	disp([‘This is ‘ variable ‘variable.’])

MATLAB Crash Course

TRAJECTORY PLOT	
Plot coordinate x and y with arrows	• plot(x, y, ’^’); % action 1 • plot(x, y, ’>’); % action 2 • plot(x, y, ’v’); %, action 3 • plot(x, y, ’<’); % action 4
Set axis min and max	axis([0 10 0 10])
Format title	title(['Execution of optimal policy with associated reward = ‘ total_reward])
Show grid	grid on
Start grid from top left corner	set(gca,'YDir','reverse')

The End