OLCAR Exercise 2

Classical Reinforcement Learning

Tim Sandy

Assigned: 21.4.2015

Due: 6.5.2015

General Info

- Files and code: http://www.adrl.ethz.ch/doku.php/adrl:education:lecture:fs2015
- Submit solutions by **Weds 6.5.2015** at Midnight
  - *_Design.m files
  - Answers to questions in .pdf format – Max. 3 sentences per question
- Interviews on Fri 8.5.2015
  - Sign-up will be available through the course website soon
  - 10 minute interview for each group
  - Graded pass/fail
  - Bonus credit for a pass: Ex.1: +0.1, Ex. 2: +0.05, Ex 3. +0.1

Introduction

- Exercise 2a: Mountain Car – Model-based Reinforcement Learning
  - Building a discretized model of a system
  - Generalized Policy Iteration (GPI)
- Exercise 2b: Cliff World – Model-free Reinforcement Learning
  - On-Policy Monte-Carlo Algorithm
  - Q-Learning

Exercise 2a: Mountain Car

State: [Position (x) ; Velocity (v)]

Action: [Acceleration (a)]

$x \in [-1.2, 0.5]$

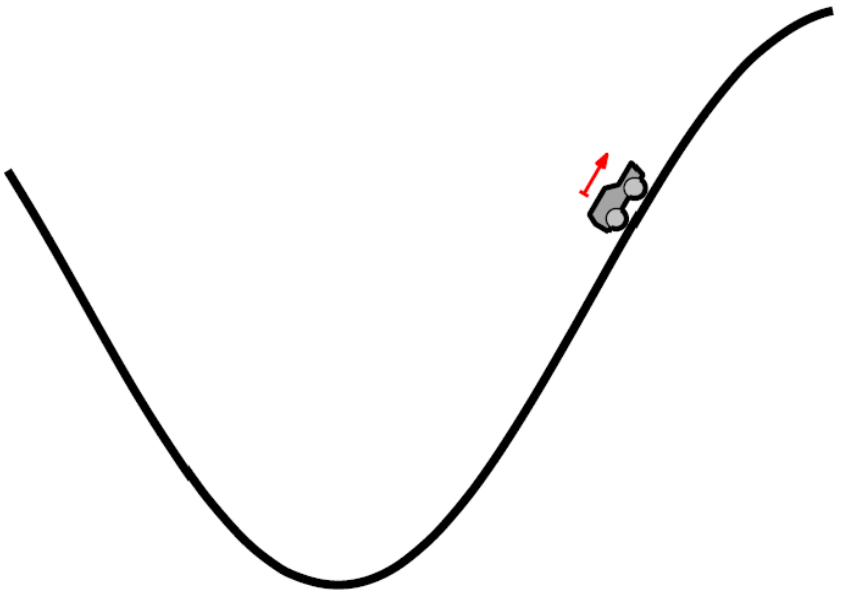
$v \in [-0.7, 0.7]$

$a \in [-1, 1]$

System Dynamics:

$$v(t+1) = v(t) + 0.001a(t) - 0.0025\cos(3x(t))$$

$$x(t+1) = x(t) + v(t+1)$$

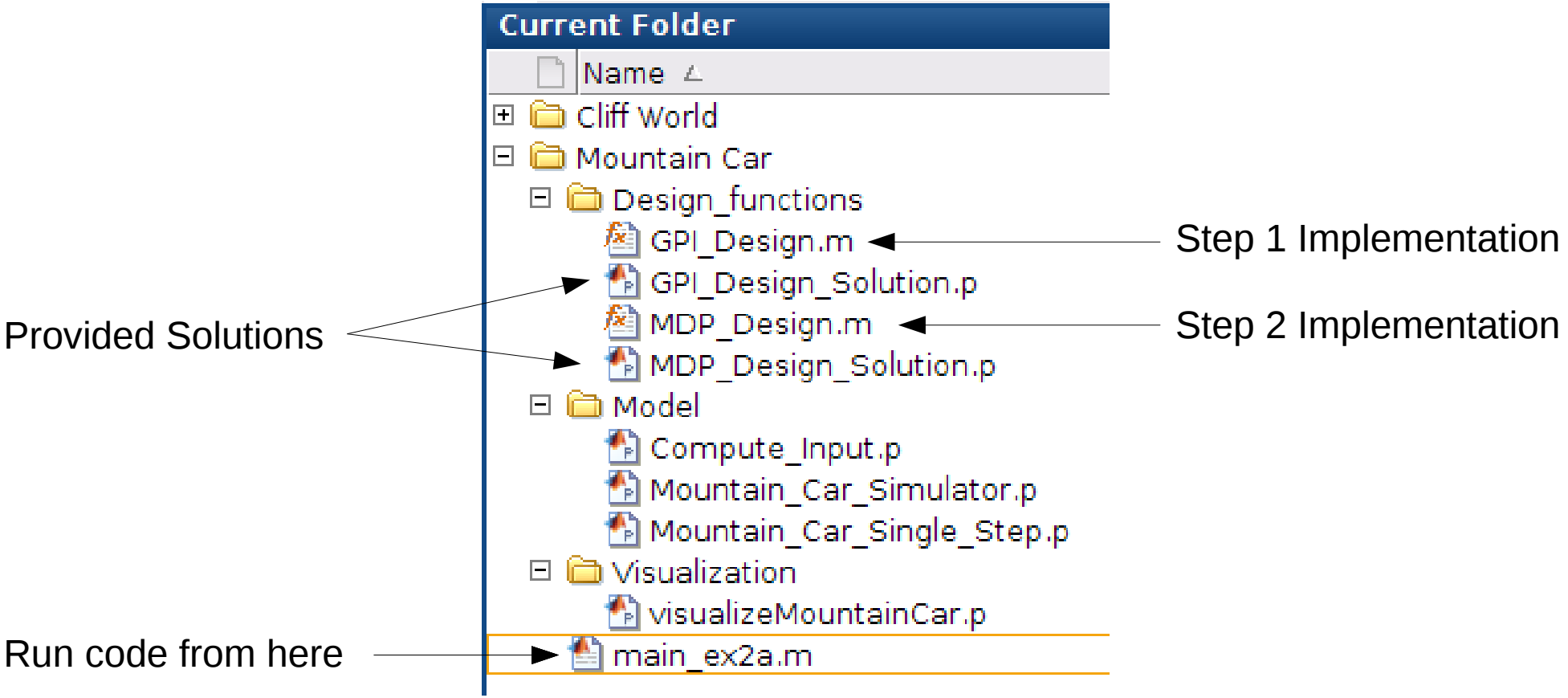


- Discrete time, continuous state, continuous actions
- If the car reaches the end-points, it is held there forever
- +10 reward when car reaches x = 0.5 , -1 reward otherwise
- Want to get the car to the top of the hill in the shortest time

Exercise 2a: Tasks

- Step 1: Create a MDP model of the system
  - Discretize the state and action spaces
  - Build a **probabilistic** model to capture the dynamics of the actuator
- Step 2: Implement the GPI algorithm
  - Find the optimal solution using Policy Iteration and Value Iteration

Exercise 2a: Software



Exercise 2a: main_ex2a.m

Information passed in & out of design functions in structures

Solution code also uses the parameter structure to allow you to tune the solutions

```
13    %% Step 1: Create a discretized model of the system
14
15    %Specify the parameters to be used for MDP modeling. These are the
16    %parameters required to run the solution code, set with values that give a
17    %reasonable solution. Please modify the parameters as you see fit.
18 -  MDP_Params = struct;
19 -  MDP_Params.pos_N = 20; %Number of bins to use for positions discretization
20 -  MDP_Params.vel_N = 20; %Number of bins to use for velocity discretization
21 -  MDP_Params.u_N = 7; %Number of bins to use for action discretization
22 -  MDP_Params.modeling_iter = 50; %Number of modeling iterations for each
23                                    %state & action
24                                    %Set = 1 to build a deterministic model
25
26
27 -  [Task,Controller] = MDP_Design(Task,MDP_Params);
28
29    %To run the provided solution use:
30    %[Task,Controller] = MDP_Design_Solution(Task,MDP_Params);
31
32
33    %Optionally save the model so it doesn't have to be recomputed every time
34 -  save('Discrete_Model.mat','Task','Controller');
35
```

System modeling computation takes a lot of time (~5-10 mins)

- Can do early tests with smaller 'pos_N', 'vel_N', and 'modeling_iter' to make debugging easier
- You can save a model to reuse it during Ex. 2b testing

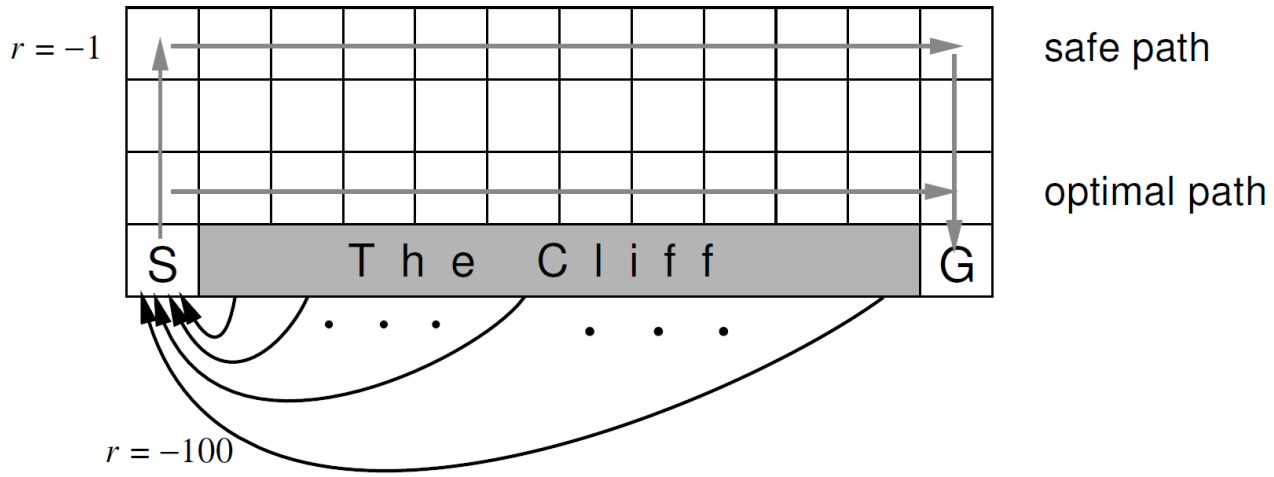
Exercise 2b: Cliff World

State: Grid Position $[x, y]$

$x \in \{1:4\}$

$y \in \{1:12\}$

Action: $\{Up, Down, Left, Right\}$

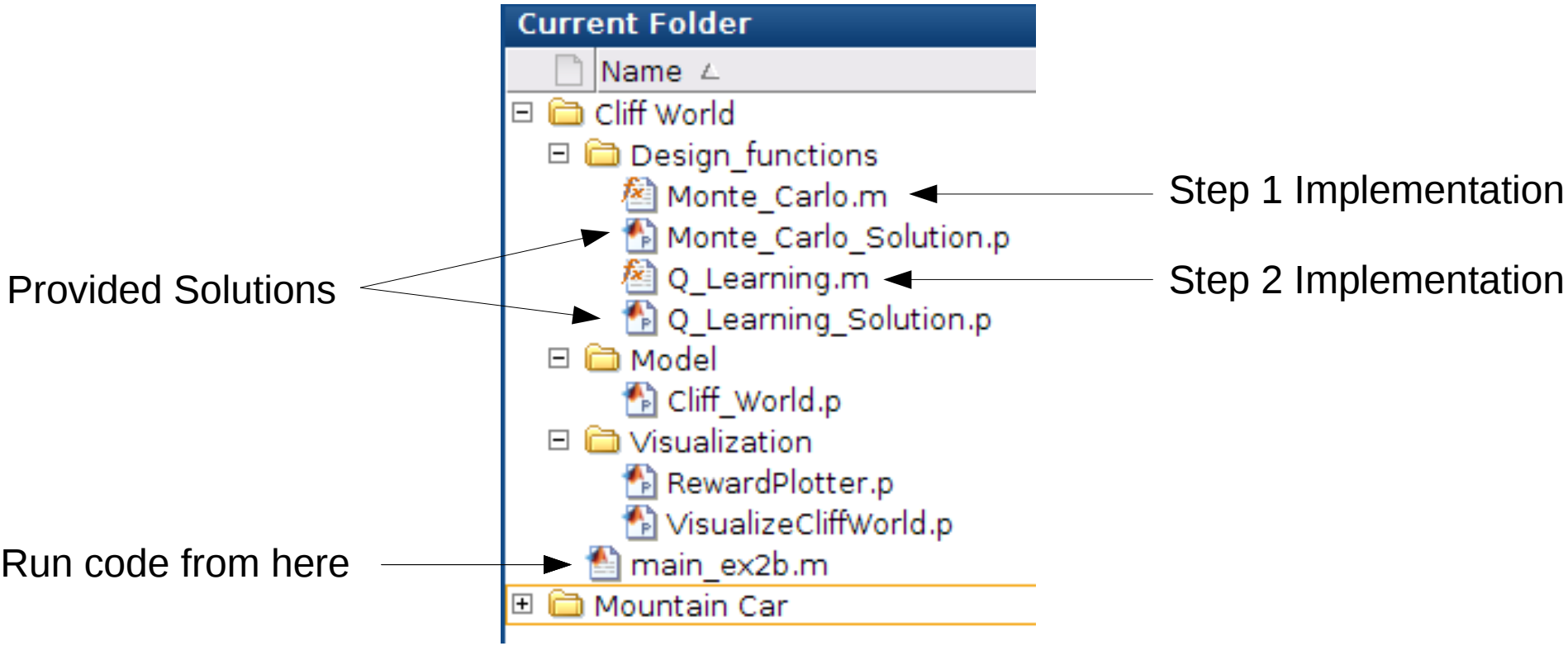


- Agent wants to get from S to G with the maximum reward
- Agent has perfect actuation, but no model knowledge
- Reward is -1 for all transitions, unless the agent enters a region marked 'Cliff', then it gets reward -100 and is moved back to S
- Episode is terminated once agent reaches G

Exercise 2b: Tasks

- Step 1: On-Policy Monte Carlo Algorithm
  - Evaluate learning episodes where the agent follows an epsilon-greedy policy
  - Implement Model-Free Policy Improvement
- Step 2: Q-Learning
  - Implement and test Q-Learning with decreasing exploration during learning

Exercise 2b: Software



Good Luck!