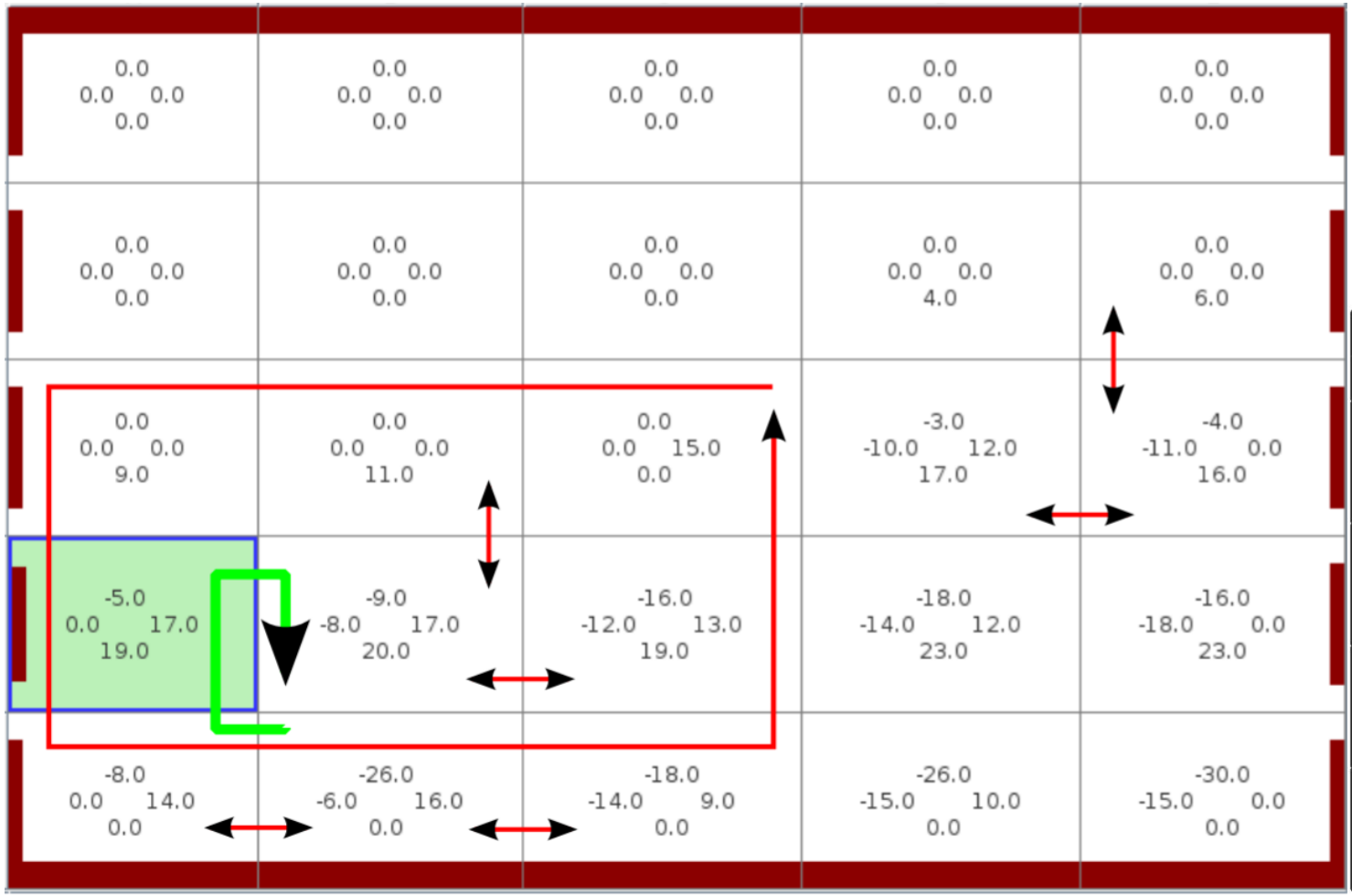
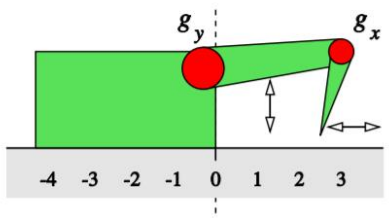
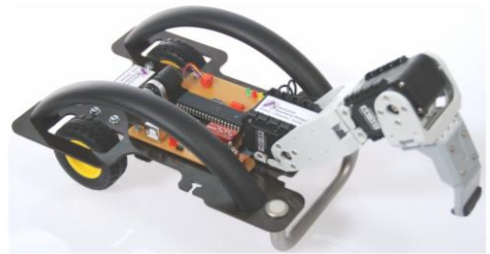
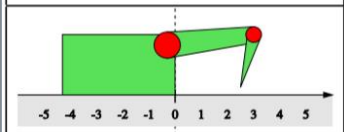
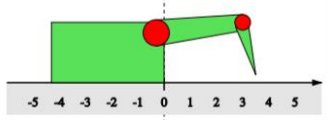
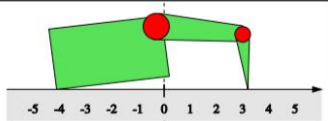
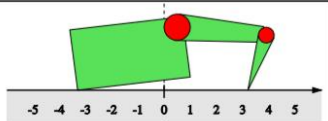




(a)



(b)

robot	time t	state g_y	state g_x	reward x	action a_t
	0	up	left	0	right
	1	up	right	0	down
	2	down	right	0	left
	3	down	left	1	up

Übung 2 – Dynamic Programming für das Lernen von Laufbewegungen eines Roboters

Implementieren Sie als Python-Funktion die Value-Iteration Lernregel aus http://www.incompleteideas.net/book/first/ebook/node44.html zum Erlernen einer optimalen Policy eines Laufroboters.

Hinweise / Vorgehensweise:

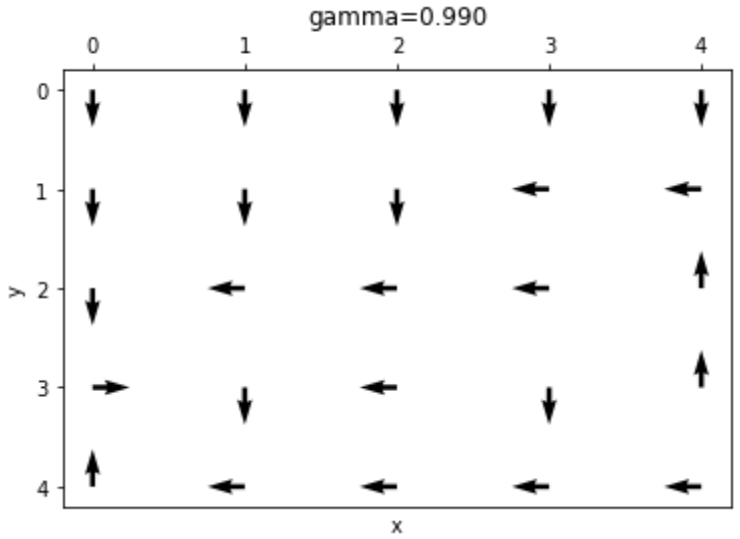
1) Implementieren Sie nur eine vereinfachte Version, bei der Sie davon ausgehen, dass jede Aktion in genau einen Nachfolgezustand s‘ führt. Die Zustandsübergangswahrscheinlichkeiten sind demnach stets: $\mathcal{P}^a_{ss'} = 1$

2) Parametrierbares Abbruchkriterium für Value-Iteration:

- Entweder maximale Anzahl von Iterationen erreicht (z.B. 10000 Iterationen),
- oder maximaler Value-Fehler kleiner als gewählte Schwelle (z.B. $\theta = 0.00001$)

3) Vorgehensweise zur Implementierung:

- Lernen Sie die optimale (grüne) Policy von Folie 1. Verifizieren Sie die Ergebnisse Ihrer Implementierung gegen die Referenz für $\gamma = 0.99$



- Untersuchen Sie verschiedene Belegungen für den Discounting-Parameter:

$$\gamma \in \{0, 0.1, 0.3, 0.5, 0.7, 0.8, 0.85, 0.9, 0.99\}$$

3.1) Erstellen Sie einen Line-Plot mit:

- X-Achse: γ
- Y-Achse: Anzahl Iterationen bis zur Konvergenz bei $\theta = 0.00001$

→ Was stellen Sie fest?

3.2) Erstellen Sie einen Line-Plot mit:

- X-Achse: γ
- Y-Achse: kumulierter Reward für 20 Schritte wenn dieser Policy auf dem Reward-Modell gefolgt wird bei Start in Zustand oben rechts: s_init=(0, 4).

→ Was stellen Sie fest?

3.3) Welches untersuchte γ ist optimal hinsichtlich Rechenzeit und Qualität des Ergebnisses?

4) Verwenden Sie die Value-Iteration Implementierung der MDP-Toolbox für Python und ersetzen die von Hand implementierte Variante durch die der Toolbox:

- pip install mdptoolbox
- https://pymdptoolbox.readthedocs.io/en/latest/api/mdp.html#mdptoolbox.mdp.ValueIteration
- Was müssen Sie zusätzlich angeben?

5) Lernen Sie eine Policy mit Value-Iteration an der Laufroboter-Simulation

- OpenAI-Gym Environment: https://github.com/micheltokic/crawlingrobot
- Interaktionsbeispiel unter: https://github.com/micheltokic/crawlingrobot/blob/master/example/CrawlingRobot.ipynb
- Aktualisieren Sie das Reward-Modell nach jeder Aktion und führen nach dieser genau einen Value-Iteration-Sweep durch (analog zum Hardware-Laufroboter).
- Leiten Sie die Policy entsprechend von der Werte-Funktion ab.
- Verwenden Sie eine geeignete Explorations-Strategie, z.B. E-Greedy oder Softmax.
- Starten Sie jede Episode ohne Vorwissen (imitiert Einschaltens des Roboters), d.h. initialisieren Sie die Werte- und Reward-Funktion stets neu mit 0.
- Optimieren Sie den kumulierten Reward über 2000 Schritte (ohne Vorwissen). Welchen Return und Reward/Step können sie erreichen? Optimieren Sie anhand dessen Ihren Explorationsparameter.