

# Convex Q-Learning

NREL Workshop on Autonomous Energy Systems


Aug. 19–20, 2020




Sean Meyn



Based on ongoing research with Prashant Mehta @UIUC

Department of Electrical and Computer Engineering  University of Florida

Inria International Chair  Inria, Paris

Thanks to to many [Gators at UF](#), and to our sponsors: NSF and ARO

# Convex Q-Learning

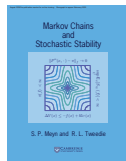
## Outline

- 1 Advertising
- 2 What Is Q?
- 3 Convex Q-Learning
- 4 Conclusions
- 5 References

# Resources

## Simons Institute Programs

- 2018 tutorial on RL  
<https://www.youtube.com/watch?v=dhEF5pfYmvc>
- Theory of Reinforcement Learning, Aug. 19–Dec. 18, 2020  
<https://simons.berkeley.edu/programs/r120>



### This talk:

[4] P. G. Mehta and S. P. Meyn. **Convex Q-learning**. *ArXiv e-prints:2008.03559*, 2020.

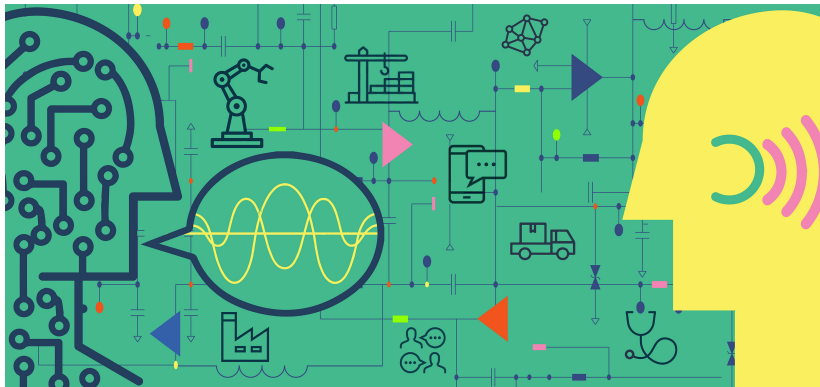
### Background:

- [3] P. G. Mehta and S. P. Meyn. **Q-learning and Pontryagin's minimum principle**. CDC, 2009.
- [5] A. Bernstein, Y. Chen, M. Colombino, E. Dall'Anese, P. Mehta, and S. Meyn. **Optimal rate of convergence for quasi-stochastic approximation**. *arXiv:1903.07228*, 2019.
- [6] A. Bernstein, Y. Chen, M. Colombino, E. Dall'Anese, P. G. Mehta, and S. Meyn. **Quasi-stochastic approximation and off-policy reinforcement learning**. CDC, 2019.
- [7] A. M. Devraj, A. Bušić, and S. Meyn. **Fundamental design principles for reinforcement learning algorithms**. In *Handbook on Reinforcement Learning and Control*. Springer, 2020.

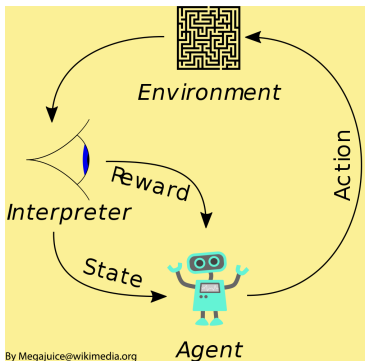
# Reinforcement Learning & Control



Theory of Reinforcement Learning  
*August 19 – December 18, 2020*

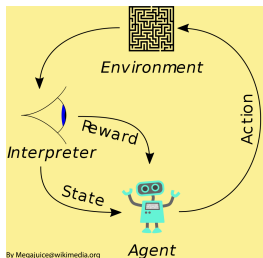


Tutorials and surveys available in real time



## Q Crash Course

# Reinforcement Learning

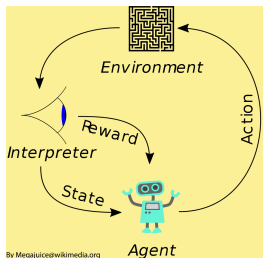


Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward –Wikipedia

Examples relevant to me and NREL

- Optimizing windfarms
- Smart Grids
- Smart Buildings

# Reinforcement Learning



Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward –Wikipedia

Examples relevant to me and NREL

- Optimizing windfarms
- Smart Grids
- Smart Buildings

RL is an emerging science, evolving alongside decision and control theory: “...as RL algorithms are increasingly and more aggressively deployed in safety critical settings, control theorists must be part of the conversation” [22]

# Dynamic Programming and RL

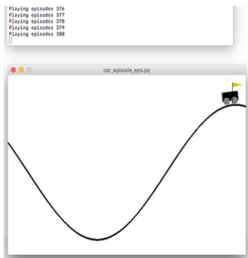
$$X_{k+1} = F(X_k, U_k)$$

Example from `gym.openai.com`: get up the hill efficiently

State:  $X_k$  denotes position and velocity

(why?)

Input (or *action*):  $U_k$  is force





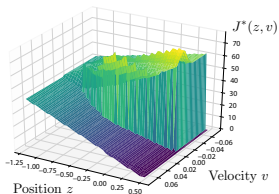
# Dynamic Programming and RL

$$X_{k+1} = F(X_k, U_k)$$

Example from `gym.openai.com`: get up the hill efficiently

State:  $X_k$  denotes position and velocity

Input (or *action*):  $U_k$  is force



Value function: 
$$J^*(x) = \min_{\text{actions}} \sum_{k=0}^{\tau} c(X_k, U_k)$$

$c$  : cost function

$\tau$  : time to reach the hill top

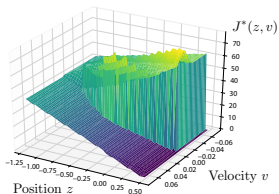
## Dynamic Programming and RL

$$X_{k+1} = F(X_k, U_k)$$

Example from `gym.openai.com`: get up the hill efficiently

State:  $X_k$  denotes position and velocity

Input (or *action*):  $U_k$  is force



Value function: 
$$J^*(x) = \min_{\text{actions}} \sum_{k=0}^{\tau} c(X_k, U_k)$$

$c$  : cost function

$\tau$  : time to reach the hill top

DP eqn: 
$$J^*(X_0) = \min_{U_0} \{c(X_0, U_0) + J^*(X_1)\}$$

Recall Bellman or Bellman-Ford

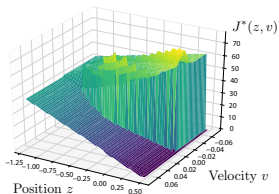
## Dynamic Programming and RL

$$X_{k+1} = F(X_k, U_k)$$

Example from `gym.openai.com`: get up the hill efficiently

State:  $X_k$  denotes position and velocity

Input (or *action*):  $U_k$  is force



Value function: 
$$J^*(x) = \min_{\text{actions}} \sum_{k=0}^{\tau} c(X_k, U_k)$$

$c$  : cost function

$\tau$  : time to reach the hill top

DP eqn: 
$$J^*(X_0) = \min_{U_0} \underbrace{c(X_0, U_0) + J^*(X_1)}_{Q^*(X_0, U_0)}$$

Q-learning is all about approximating  $Q^*$

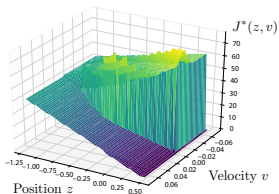
## Dynamic Programming and RL

$$X_{k+1} = F(X_k, U_k)$$

Example from `gym.openai.com`: get up the hill efficiently

State:  $X_k$  denotes position and velocity

Input (or *action*):  $U_k$  is force



Value function: 
$$J^*(x) = \min_{\text{actions}} \sum_{k=0}^{\tau} c(X_k, U_k)$$

$c$  : cost function

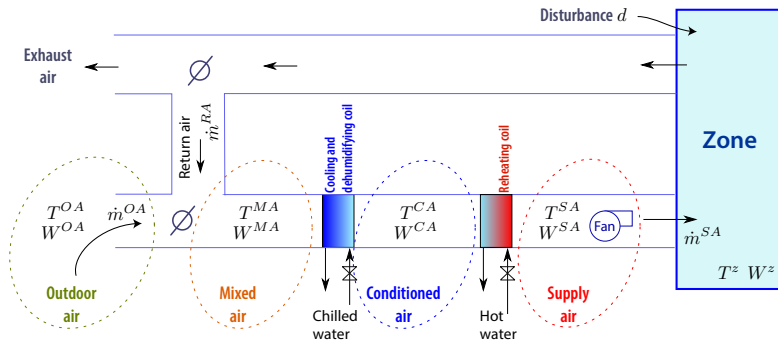
$\tau$  : time to reach the hill top

DP eqn: 
$$J^*(X_0) = \min_{U_0} \underbrace{c(X_0, U_0) + J^*(X_1)}_{Q^*(X_0, U_0)}$$

Q-learning is all about approximating  $Q^*$

If we know  $Q^*$ , we obtain  $U_k^* = \phi^*(X_k)$

# Control Design for Heating and Ventilation



Five dimensional state space and four dimensional input space

Joint work with N. S. Raman, P. Barooah @ UF MAE, A. Devraj @ Stanford

See final page of references, and bibliography of [17]

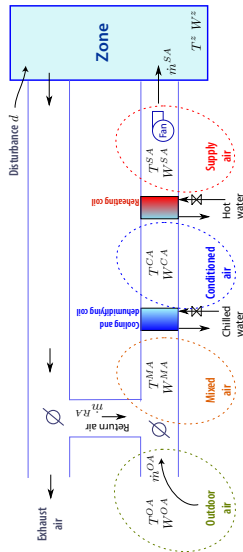
# Control Design for Heating and Ventilation

**Input:**  $U_k := [m_{sa}(k), r_{oa}(k), T_{ca}(k), T_{sa}(k)]^T$

- 1 Supply air flow rate ( $m_{sa}$ )
- 2 Outdoor air ratio ( $r_{oa}$ )
- 3 Conditioned air temperature ( $T_{ca}$ )
- 4 Supply air temperature ( $T_{sa}$ )

**State:**  $X_k := [T_z(k), W_z(k), T_{oa}(k), W_{oa}(k), U(k-1)]^T$

- 1 Zone air temperature ( $T_z$ )
- 2 Zone air humidity ratio ( $W_z$ )
- 3 Outdoor air temperature ( $T_{oa}$ )
- 4 Outdoor air humidity ratio ( $W_{oa}$ )
- 5 Control inputs from the previous time step
- 6 ... forecast of occupancy, weather, ...



# Control Design for Heating and Ventilation

**Input:**  $U_k := [m_{sa}(k), r_{oa}(k), T_{ca}(k), T_{sa}(k)]^T$

**State:**

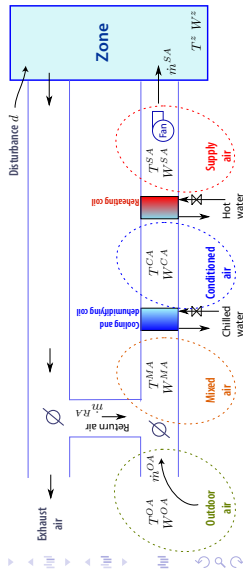
$X_k := [T_z(k), W_z(k), T_{oa}(k), W_{oa}(k), U(k-1)]^T$

Quadratic basis: + Zap Q-learning

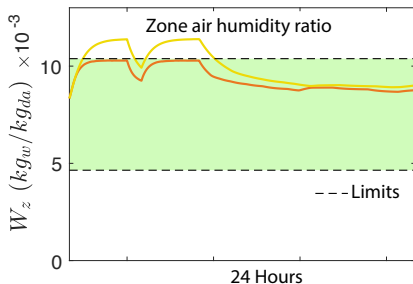
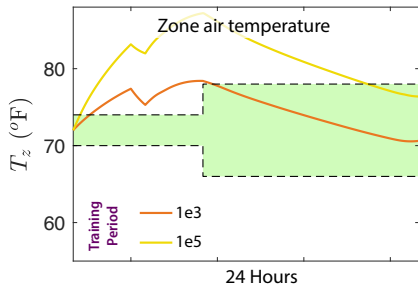
$$Q^\theta(x, u) = (x, u)^T M_\theta(x, u) + (x, u)^T L_\theta + k_\theta$$

$$= \sum_i \theta_i \psi_i(x, u)$$

Initial results are great ...



## Close Loop Response: Temperature and humidity evolution



- **Goal:** Maintain temperature / humidity, *and* minimize energy consumption
- **Inputs:** Air-flow rate, out-door air ratio, conditioned air temperature, supply air temperature
- **Approach:** Find  $\theta^*$  with quadratic basis:

$$Q^{\theta}(x, u) = (x, u)^T M_{\theta}(x, u) + (x, u)^T L_{\theta} + k_{\theta}$$



## From DP to Q-learning

$$X_{k+1} = F(X_k, U_k)$$

DP eqn:  $J^*(X_0) = \min_{U_0} \underbrace{\{c(X_0, U_0) + J^*(X_1)\}}_{Q^*(X_0, U_0)}$

## From DP to Q-learning

$$X_{k+1} = F(X_k, U_k)$$

$$\text{DP eqn: } J^*(X_0) = \min_{U_0} \underbrace{\{c(X_0, U_0) + J^*(X_1)\}}_{Q^*(X_0, U_0)}$$

**Magic:** Denote  $Q^*(x) = \min_u Q^*(x, u) = J^*(x)$

$\implies$  Fixed point equation for Q-function

$$Q^*(x, u) = c(x, u) + J^*(F(x, u))$$

## From DP to Q-learning

$$X_{k+1} = F(X_k, U_k)$$

$$\text{DP eqn: } J^*(X_0) = \min_{U_0} \underbrace{\{c(X_0, U_0) + J^*(X_1)\}}_{Q^*(X_0, U_0)}$$

Magic: Denote  $\underline{Q}^*(x) = \min_u Q^*(x, u) = J^*(x)$

$\implies$  Fixed point equation for Q-function

$$Q^*(x, u) = c(x, u) + J^*(F(x, u)) = c(x, u) + \underline{Q}^*(F(x, u))$$

## From DP to Q-learning

$$X_{k+1} = F(X_k, U_k)$$

$$\text{DP eqn: } J^*(X_0) = \min_{U_0} \underbrace{\{c(X_0, U_0) + J^*(X_1)\}}_{Q^*(X_0, U_0)}$$

Magic: Denote  $\underline{Q}^*(x) = \min_u Q^*(x, u) = J^*(x)$

$\implies$  Fixed point equation for Q-function

$$Q^*(x, u) = c(x, u) + J^*(F(x, u)) = c(x, u) + \underline{Q}^*(F(x, u))$$

Choose approximation among  $\{Q^\theta(x, u) : \theta \in \mathbb{R}^d\}$

$$\text{Bellman error: } \mathcal{E}^\theta(x, u) = -Q^\theta(x, u) + c(x, u) + \underline{Q}^\theta(F(x, u))$$

For example,  $\theta_i$  is a “weight” in a **neural network**, or

$$Q^\theta(x, u) = \sum_i \theta_i \psi_i(x, u)$$

## From DP to Q-learning

$$X_{k+1} = F(X_k, U_k)$$

$$\text{DP eqn: } J^*(X_0) = \min_{U_0} \underbrace{\{c(X_0, U_0) + J^*(X_1)\}}_{Q^*(X_0, U_0)}$$

Magic: Denote  $\underline{Q}^*(x) = \min_u Q^*(x, u) = J^*(x)$

$\implies$  Fixed point equation for Q-function

$$Q^*(x, u) = c(x, u) + J^*(F(x, u)) = c(x, u) + \underline{Q}^*(F(x, u))$$

Choose approximation among  $\{Q^\theta(x, u) : \theta \in \mathbb{R}^d\}$

$$\text{Bellman error: } \mathcal{E}^\theta(x, u) = -Q^\theta(x, u) + c(x, u) + \underline{Q}^\theta(F(x, u))$$

Model Free Error Representation:

$$\mathcal{E}^\theta(X_k, U_k) = -Q^\theta(X_k, U_k) + c(X_k, U_k) + \underline{Q}^\theta(X_{k+1})$$

# Q(0) Learning and Deep Q-Learning

A generalization of Watkins' algorithm [25, 7, 1]

Model Free Error Representation:

$$\mathcal{E}^\theta(X_k, U_k) = -Q^\theta(X_k, U_k) + c(X_k, U_k) + \underline{Q}^\theta(X_{k+1})$$

# Q(0) Learning and Deep Q-Learning

A generalization of Watkins' algorithm [25, 7, 1]

**Model Free Error Representation:**

$$\mathcal{E}^\theta(X_k, U_k) = -Q^\theta(X_k, U_k) + c(X_k, U_k) + \underline{Q}^\theta(X_{k+1})$$

**Goal:** Find roots of

$$\bar{f}(\theta) = \mathbb{E}_\infty[\zeta^\theta \mathcal{E}^\theta(X, U)] = \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{k=0}^{T-1} \zeta_k^\theta \mathcal{E}^\theta(X_k, U_k)$$

**Eligibility vector:**  $\zeta_k^\theta = \nabla_\theta Q^\theta(X_k, U_k)$       *Q(0)-learning*

## Q(0) Learning and Deep Q-Learning

A generalization of Watkins' algorithm [25, 7, 1]

Model Free Error Representation:

$$\mathcal{E}^\theta(X_k, U_k) = -Q^\theta(X_k, U_k) + c(X_k, U_k) + \underline{Q}^\theta(X_{k+1})$$

**Goal:** Find roots of

$$\bar{f}(\theta) = \mathbb{E}_\infty[\zeta^\theta \mathcal{E}^\theta(X, U)] = \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{k=0}^{T-1} \zeta_k^\theta \mathcal{E}^\theta(X_k, U_k)$$

Eligibility vector:  $\zeta_k^\theta = \nabla_\theta Q^\theta(X_k, U_k)$

**Design principle:**

Step 1: consider an ODE:  $\frac{d}{dt} \theta_t = -G_t \bar{f}(\theta_t)$  (matrix gain part of design)

Step 2: translate to a discrete time algorithm based on measurements.



# Q(0) Learning and Deep Q-Learning

A generalization of Watkins' algorithm [25, 7, 1]

Model Free Error Representation:

$$\mathcal{E}^\theta(X_k, U_k) = -Q^\theta(X_k, U_k) + c(X_k, U_k) + \underline{Q}^\theta(X_{k+1})$$

**Goal:** Find roots of  $\bar{f}(\theta^*) = 0$  *Why?*

$$\bar{f}(\theta) = \mathbb{E}_\infty[\zeta^\theta \mathcal{E}^\theta(X, U)] = \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{k=0}^{T-1} \zeta_k^\theta \mathcal{E}^\theta(X_k, U_k)$$

Eligibility vector:  $\zeta_k^\theta = \nabla_\theta Q^\theta(X_k, U_k)$

Design principle:

Step 1: consider an ODE:  $\frac{d}{dt}\theta_t = -G_t \bar{f}(\theta_t)$  (matrix gain part of design)

Step 2: translate to a discrete time algorithm based on measurements.

## Q(0) Learning and Deep Q-Learning

$$\bar{f}(\theta) = \lim \frac{1}{T} \sum_{k=0}^{T-1} \zeta_k^\theta \mathcal{E}^\theta(X_k, U_k) \quad \bar{f}(\theta^*) = 0 \quad \text{Why?}$$

Troubles with Q: Slow! Does a root exist? Does it have significance?

## Q(0) Learning and Deep Q-Learning

$$\bar{f}(\theta) = \lim \frac{1}{T} \sum_{k=0}^{T-1} \zeta_k^\theta \mathcal{E}^\theta(X_k, U_k) \quad \bar{f}(\theta^*) = 0 \quad \text{Why?}$$

Troubles with Q: Slow! Does a root exist? Does it have significance?

Batch algorithms to the rescue? [18, 19, 20, 21]

$$\mathbf{DQN} \quad \theta_{n+1} = \arg \min_{\theta} \left\{ \mathcal{E}_n(\theta) + \frac{1}{\alpha_{n+1}} \|\theta - \theta_n\|^2 \right\}$$

$$\mathcal{E}_n(\theta) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} \left[ -Q^\theta(X_k, U_k) + c(X_k, U_k) + \underline{Q}^{\theta_n}(X_{k+1}) \right]^2$$

## Q(0) Learning and Deep Q-Learning

$$\bar{f}(\theta) = \lim \frac{1}{T} \sum_{k=0}^{T-1} \zeta_k^\theta \mathcal{E}^\theta(X_k, U_k) \quad \bar{f}(\theta^*) = 0 \quad \text{Why?}$$

Troubles with Q: Slow! Does a root exist? Does it have significance?

Batch algorithms to the rescue? [18, 19, 20, 21]

$$\mathbf{DQN} \quad \theta_{n+1} = \arg \min_{\theta} \left\{ \mathcal{E}_n(\theta) + \frac{1}{\alpha_{n+1}} \|\theta - \theta_n\|^2 \right\}$$

$$\mathcal{E}_n(\theta) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} \left[ -Q^\theta(X_k, U_k) + c(X_k, U_k) + \underline{Q}^{\theta_n}(X_{k+1}) \right]^2$$

With a linear parameterization, this is a quadratic program!

## Q(0) Learning and Deep Q-Learning

$$\bar{f}(\theta) = \lim \frac{1}{T} \sum_{k=0}^{T-1} \zeta_k^\theta \mathcal{E}^\theta(X_k, U_k) \quad \bar{f}(\theta^*) = 0 \quad \text{Why?}$$

Troubles with Q: Slow! Does a root exist? Does it have significance?

Batch algorithms to the rescue? [18, 19, 20, 21]

$$\mathbf{DQN} \quad \theta_{n+1} = \arg \min_{\theta} \left\{ \mathcal{E}_n(\theta) + \frac{1}{\alpha_{n+1}} \|\theta - \theta_n\|^2 \right\}$$

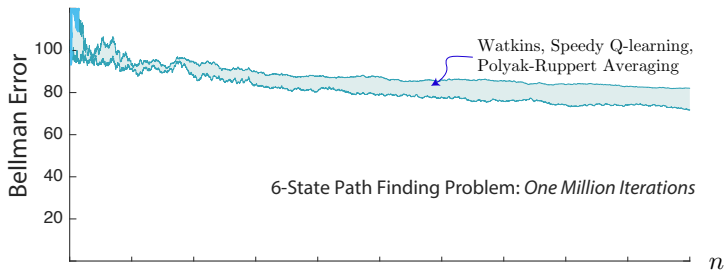
$$\mathcal{E}_n(\theta) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} \left[ -Q^\theta(X_k, U_k) + c(X_k, U_k) + \underline{Q}^{\theta_n}(X_{k+1}) \right]^2$$

With a linear parameterization, this is a quadratic program!

Sadly,

**ODE approximation for DQN  $\equiv$  Q(0) Learning**

*Even for neural network function approximation [M&M, 2020]*



## Convex Q-Learning

# Every DP is an LP

Every control student knows this, starting with [Manne, 1960]

**Proposition:** [Subject to mild assumptions]

$J^*$  solves the following LP:

$$\max_J \langle \mu, J \rangle$$

$$\text{s.t. } J(x) \leq c(x, u) + J(F(x, u)), \quad x \in X, u \in U(x)$$

$J$  is continuous, and  $J(x^e) = 0$ .

$\mu$  a probability measure on  $X$  (given)

- Applications to ADP in the thesis of de Farias (with BVR) [8, 9]
- One way to derive the SDP representation of LQR [Boyd et al]
- Applications in deterministic control every day

# Every DP is an LP

Every control student knows this, starting with [Manne, 1960]

**Proposition:** [Subject to mild assumptions]

The pair  $(J^*, Q^*)$  solve the following LP:

$$\max_{J, Q} \langle \mu, J \rangle$$

$$\text{s.t. } Q(x, u) \leq c(x, u) + J(F(x, u))$$

$$Q(x, u) \geq J(x), \quad x \in \mathbf{X}, u \in \mathbf{U}(x)$$

$J$  is continuous, and  $J(x^e) = 0$ .

$\mu$  a probability measure on  $\mathbf{X}$  (given)



# Every DP is an LP

Every control student knows this, starting with [Manne, 1960]

**Proposition:** [Subject to mild assumptions]

The pair  $(J^*, Q^*)$  solve the following LP:

$$\max_{J, Q} \langle \mu, J \rangle$$

$$\text{s.t. } Q(x, u) \leq c(x, u) + J(F(x, u))$$

$$Q(x, u) \geq J(x), \quad x \in X, u \in U(x)$$

$J$  is continuous, and  $J(x^e) = 0$ .

$\mu$  a probability measure on  $X$  (given)

Over-parameterization for RL more recent.

Motivation:  $Q(X_k, U_k) \leq c(X_k, U_k) + J(X_{k+1})$  (observed)

# Every DP is a QP

**Proposition:** [Subject to mild assumptions]

The pair  $(J^*, Q^*)$  solve the following QP:

$$\min_{J, Q} - \langle \mu, J \rangle + \kappa \langle \nu, \mathcal{E}^2 \rangle$$

$$\text{s.t. } 0 \leq \mathcal{E}(x, u) := -Q(x, u) + c(x, u) + J(F(x, u))$$

$$Q(x, u) \geq J(x), \quad x \in X, u \in U(x)$$

$J$  is continuous, and  $J(x^e) = 0$ .

$\nu$  a probability measure on  $X \times U$

# Every DP is a QP

**Proposition:** [Subject to mild assumptions]

The pair  $(J^*, Q^*)$  solve the following QP:

$$\min_{J, Q} - \langle \mu, J \rangle + \kappa \langle \nu, \mathcal{E}^2 \rangle$$

$$\text{s.t. } 0 \leq \mathcal{E}(x, u) := -Q(x, u) + c(x, u) + J(F(x, u))$$

$$Q(x, u) \geq J(x), \quad x \in X, u \in U(x)$$

$J$  is continuous, and  $J(x^e) = 0$ .

$\nu$  a probability measure on  $X \times U$

The objective and constraints can be observed, without a model

$\implies$  Long list of possible RL approximations

# Every DP is a QP $\implies$ Convex Q Learning

$$\min_{\theta} - \langle \mu, J^{\theta} \rangle + \kappa \langle \nu, \mathcal{E}(\theta)^2 \rangle$$

$$\text{s.t. } 0 \leq -Q^{\theta}(X_k, U_k) + c(X_k, U_k) + J^{\theta}(X_{k+1})$$

$$Q^{\theta}(x, u) \geq J^{\theta}(x) \quad \iff \text{Enforce through function architecture}$$

# Every DP is a QP $\implies$ Convex Q Learning

$$\min_{\theta} - \langle \mu, J^{\theta} \rangle + \kappa \langle \nu, \mathcal{E}(\theta)^2 \rangle$$

$$\text{s.t. } 0 \leq -Q^{\theta}(X_k, U_k) + c(X_k, U_k) + J^{\theta}(X_{k+1}) \implies z_n(\theta) \geq 0$$

$$z_n(\theta) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} [-Q^{\theta}(X_k, U_k) + c(X_k, U_k) + J^{\theta}(X_{k+1})] \zeta_k^+$$

$\zeta_k^+$  : vector with non-negative entries

# Every DP is a QP $\implies$ Convex Q Learning

$$\min_{\theta} - \langle \mu, J^{\theta} \rangle + \kappa \langle \nu, \mathcal{E}(\theta)^2 \rangle$$

$$\text{s.t. } 0 \leq -Q^{\theta}(X_k, U_k) + c(X_k, U_k) + J^{\theta}(X_{k+1}) \implies z_n(\theta) \geq 0$$

$$z_n(\theta) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} [-Q^{\theta}(X_k, U_k) + c(X_k, U_k) + J^{\theta}(X_{k+1})] \zeta_k^+$$

$$\bar{\mathcal{E}}_n^2(\theta) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} [-Q^{\theta}(X_k, U_k) + c(X_k, U_k) + J^{\theta}(X_{k+1})]^2$$

# Every DP is a QP $\implies$ Convex Q Learning

$$\min_{\theta} \quad -\langle \mu, J^{\theta} \rangle + \kappa \langle \nu, \mathcal{E}(\theta)^2 \rangle$$

$$\text{s.t.} \quad 0 \leq -Q^{\theta}(X_k, U_k) + c(X_k, U_k) + J^{\theta}(X_{k+1}) \implies z_n(\theta) \geq 0$$

$$z_n(\theta) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} [-Q^{\theta}(X_k, U_k) + c(X_k, U_k) + J^{\theta}(X_{k+1})] \zeta_k^+$$

$$\bar{\mathcal{E}}_n^2(\theta) = \frac{1}{r_n} \sum_{k=T_n}^{T_{n+1}-1} [-Q^{\theta}(X_k, U_k) + c(X_k, U_k) + J^{\theta}(X_{k+1})]^2$$

## Convex Q Version 1.0

$$\theta_{n+1} = \arg \min_{\theta \in \Theta} \left\{ -\langle \mu, J^{\theta} \rangle + \kappa \bar{\mathcal{E}}_n^2(\theta) - \lambda_n^T z_n(\theta) + \frac{1}{\alpha_{n+1}} \frac{1}{2} \|\theta - \theta_n\|^2 \right\}$$

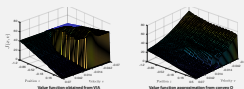
$$\lambda_{n+1} = [\lambda_n - \alpha_{n+1} z_n(\theta_n)]_+$$

# Conclusions

The LP and QP characterization of DP equations gives rise to RL algorithms that provably converge, and for which **we know what problem we are actually solving!**



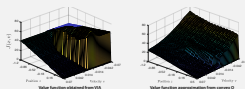
# Conclusions



The LP and QP characterization of DP equations gives rise to RL algorithms that provably converge, and for which we know what problem we are actually solving!

- Extensions to stochastic control – *not a big deal*
- Much more work is required to develop these algorithms for particular applications, and to improve efficiency

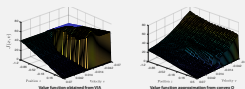
# Conclusions



The LP and QP characterization of DP equations gives rise to RL algorithms that provably converge, and for which we know what problem we are actually solving!

- Extensions to stochastic control – *not a big deal*
- Much more work is required to develop these algorithms for particular applications, and to improve efficiency
- Extensions to Convex Policy Gradient algorithms are expected soon!

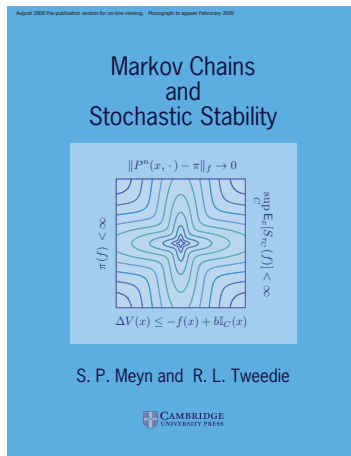
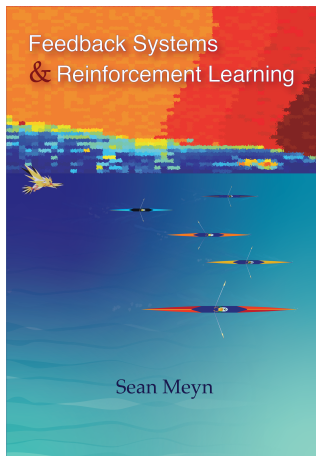
# Conclusions



The LP and QP characterization of DP equations gives rise to RL algorithms that provably converge, and for which we know what problem we are actually solving!

- Extensions to stochastic control – *not a big deal*
- Much more work is required to develop these algorithms for particular applications, and to improve efficiency
- Extensions to Convex Policy Gradient algorithms are expected soon!

Thank you!



## References

# Selected References I

- [1] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press. On-line edition at <http://www.cs.ualberta.ca/~sutton/book/the-book.html>, Cambridge, MA, 2nd edition, 2018.
- [2] S. P. Meyn. *Feedback systems and reinforcement learning*. In preparation, 2020.
- [3] P. G. Mehta and S. P. Meyn. *Q-learning and Pontryagin's minimum principle*. In *Proc. of the IEEE Conf. on Dec. and Control*, pages 3598–3605, Dec. 2009.
- [4] P. G. Mehta and S. P. Meyn. *Convex Q-learning, part 1: Deterministic optimal control*. *ArXiv e-prints:2008.03559*, 2020.
- [5] A. Bernstein, Y. Chen, M. Colombino, E. Dall'Anese, P. Mehta, and S. Meyn. *Optimal rate of convergence for quasi-stochastic approximation*. *arXiv:1903.07228*, 2019.
- [6] A. Bernstein, Y. Chen, M. Colombino, E. Dall'Anese, P. G. Mehta, and S. Meyn. *Quasi-stochastic approximation and off-policy reinforcement learning*. In *Proc. of the IEEE Conf. on Dec. and Control*, pages 5244–5251, Mar 2019.
- [7] A. M. Devraj, A. Bušić, and S. Meyn. *Fundamental design principles for reinforcement learning algorithms*. In *Handbook on Reinforcement Learning and Control*. Springer, 2020.
- [8] D. P. de Farias and B. Van Roy. *The linear programming approach to approximate dynamic programming*. *Operations Res.*, 51(6):850–865, 2003.

## Selected References II

- [9] D. P. de Farias and B. Van Roy. *A cost-shaping linear program for average-cost approximate dynamic programming with performance guarantees*. *Math. Oper. Res.*, 31(3):597–620, 2006.
- [10] A. M. Devraj. *Reinforcement Learning Design with Optimal Learning Rate*. PhD thesis, University of Florida, 2019.
- [11] D. Huang, W. Chen, P. Mehta, S. Meyn, and A. Surana. *Feature selection for neuro-dynamic programming*. In F. Lewis, editor, *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley, 2011.
- [12] A. M. Devraj, A. Bušić, and S. Meyn. *Fundamental design principles for reinforcement learning algorithms*. In *Handbook on Reinforcement Learning and Control*. Springer, 2020.
- [13] S. Chen, A. M. Devraj, A. Bušić, and S. Meyn. *Explicit Mean-Square Error Bounds for Monte-Carlo and Linear Stochastic Approximation*. *arXiv e-prints*, 2002.02584, Feb. 2020.
- [14] A. M. Devraj and S. P. Meyn. *Q-learning with Uniformly Bounded Variance: Large Discounting is Not a Barrier to Fast Learning*. *arXiv e-prints 2002.10301*, and to appear *AISTATS*, Feb. 2020.
- [15] A. M. Devraj, A. Bušić and S. P. Meyn. *Zap Meets Momentum: Stochastic Approximation Algorithms with Optimal Convergence Rate*. *ArXiv* , September 2018.

## Selected References III

- [16] P. G. Mehta and S. P. Meyn. *Q-learning and Pontryagin's minimum principle*. In *IEEE Conference on Decision and Control*, 3598–3605, Dec. 2009.
- [17] N. S. Raman, A. M. Devraj, P. Barooah, and S. P. Meyn. *Reinforcement learning for control of building HVAC systems*. In *American Control Conference*, July 2020.
- [18] M. Riedmiller. *Neural fitted Q iteration – first experiences with a data efficient neural reinforcement learning method*. In J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, and L. Torgo, editors, *Machine Learning: ECML 2005*, pages 317–328, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- [19] S. Lange, T. Gabel, and M. Riedmiller. *Batch reinforcement learning*. In *Reinforcement learning*, pages 45–73. Springer, 2012.
- [20] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. *Playing Atari with deep reinforcement learning*. *ArXiv*, abs/1312.5602, 2013.
- [21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. *Human-level control through deep reinforcement learning*. *Nature*, 518:529–533, 2015.

# Selected References IV

- [22] N. Matni, A. Proutiere, A. Rantzer, and S. Tu. *From self-tuning regulators to reinforcement learning and back again*. In *Proc. of the IEEE Conf. on Dec. and Control*, pages 3724–3740, 2019.
- [23] C. Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2010.
- [24] V. S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Hindustan Book Agency and Cambridge University Press, Delhi, India & Cambridge, UK, 2008.
- [25] C. J. C. H. Watkins and P. Dayan. *Q-learning*. *Machine Learning*, 8(3-4):279–292, 1992.
- [26] J. N. Tsitsiklis and B. Van Roy. *An analysis of temporal-difference learning with function approximation*. *IEEE Trans. Automat. Control*, 42(5):674–690, 1997.
- [27] S. P. Meyn and R. L. Tweedie. *Markov chains and stochastic stability*. Cambridge University Press, Cambridge, second edition, 2009. Published in the Cambridge Mathematical Library.
- [28] S. P. Meyn. *Control Techniques for Complex Networks*. Cambridge University Press, 2007. *See last chapter on simulation and average-cost TD learning*