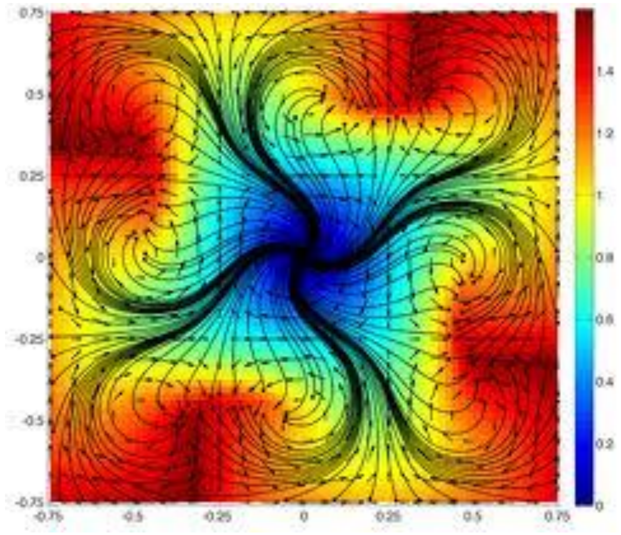
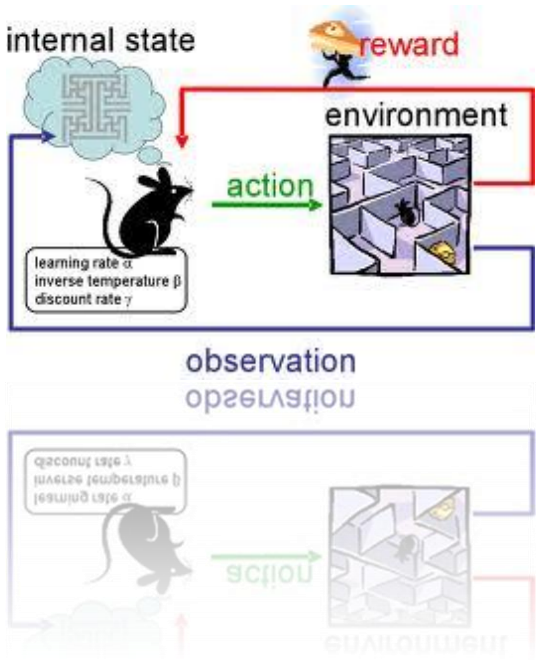
Combining Markov Decision Processes with Linear Optimal Controllers



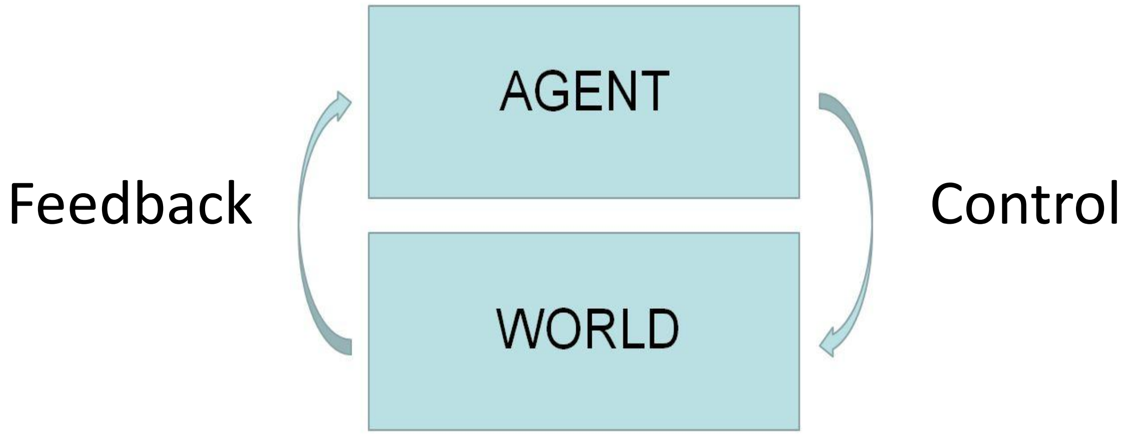


ICCSW 2011

Ekaterina Abramova, Daniel Kuhn, Aldo Faisal
Imperial College London
Department of Computing

30 Sep 2011

Optimal Control (OC)



- Aims to find a control law which would manipulate a dynamical system in an optimal way, while minimizing a cost associated with that system

- Non-linear OC most difficult area of control theory

Motivation

- Currently non-linear OC problems are approximated using iterative methods. These are computationally costly and we need better methods

- Combine robust power of OC with adaptive properties of RL (requirement: ability to formally state/approximate system dynamics and costs)

- Optimal Control:
  - continuous state
  - discrete time
  - solves linear systems
  - approximates non-linear systems

- Reinforcement Learning:
  - discrete state

    => curse of dimensionality
  - discrete time
  - does not have regard for system dynamics

Linear Optimal Control (Linear Quadratic Regulator)

- Deterministic linear Optimal Control

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k + \dot{\mathbf{x}}_k \mathrm{D}t \\ &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k\end{aligned}$$

- Minimize total cost of finite horizon problem

$$J = \frac{1}{2}\mathbf{x}_n{}^T \mathbf{Q}^f \mathbf{x}_n + \sum_{k=0}^{n-1}\left(\frac{1}{2}\mathbf{u}_k{}^T \mathbf{R}\mathbf{u}_k + \frac{1}{2}\mathbf{x}_k{}^T \mathbf{Q}\mathbf{x}_k\right)$$

- Control update rule

$$\mathbf{u}_k = -\mathbf{L}\mathbf{x}_k$$

Non-Linear Optimal Control

- Deterministic non-linear Optimal Control

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \dot{\mathbf{x}}_k \Delta t$$
$$= \mathbf{A}(\mathbf{x}_k)\mathbf{x}_k + \mathbf{B}(\mathbf{x}_k)\mathbf{u}_k$$

- Minimize total cost of finite horizon problem

$$J = \frac{1}{2}\mathbf{x}_n{}^T \mathbf{Q}^f \mathbf{x}_n + \sum_{k=0}^{n-1} \left(\frac{1}{2}\mathbf{u}_k{}^T \mathbf{R} \mathbf{u}_k + \frac{1}{2}\mathbf{x}_k{}^T \mathbf{Q} \mathbf{x}_k\right)$$

- Control update rule

$$\mathbf{u}_k = \mathbf{L}(\mathbf{x}_k)\mathbf{x}_k$$

Main Methods for Non-Linear OC

- Dynamic Programming – relies on Bellman Optimality Principle in discrete domain [Bellman, 1957]

- Control Lyapunov Functions (CLFs) – if such can be found for the problem dynamics it guarantees the existence of a stable control [Sontag, 1983]

- Receding Horizon Control (RHC), aka Model Predictive Control (MPC) – obtains iterative solution on a finite horizon prediction scale (i.e. localization in time) [Kwon, 1983]

- Differential Dynamic Programming (DDP) – maintains representation of a single trajectory and improves it locally based on dynamic programming (local DP rather than global DP) [Jacobson, 1968]

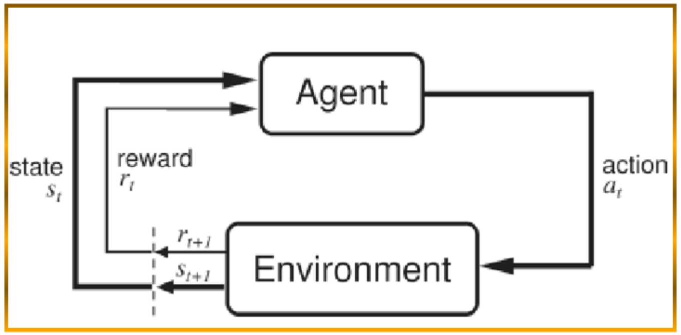
- Iterative Linear Quadratic Gaussian (iLQG) - obtains iterative linearizations around nominal trajectory (with noise) [Todorov, 2005]

Current Methods' Disadvantages

- **DP:** Curse of dimensionality, especially for continuous state spaces
- **CLFs**: No systematic techniques for finding CLFs; only work well on certain systems. Lack stability (diverge away from equilibrium) unless CLF fits closely with the system's value function
- **RHC/MPC**: May perform suboptimally in practice and can be difficult to implement
- **DDP and iLQG**: involve iterative calculations of the system dynamics and feedback gain, therefore making them computationally costly

Reinforcement Learning (RL)

- Computational approach to improving agent's behaviour through interactions with the environment (explores the world through actions and receives corresponding rewards)
- One of the three pillars of Machine Learning: Supervised Learning, Unsupervised Learning and Reinforcement Learning (main methods: DP, MC, TD)



Reinforcement Learning

- Markov Decision Process is a framework used to describe a class of control problems

- Reinforcement learning problems that satisfy the Markov property are MDPs:

$$p(s_{t+1} = s^{'}, r_{t+1} = r \mid s_t, a_{t,} r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0) = p(s_{t+1} = s^{'}, r_{t+1} = r \mid s_t, a_t)$$

- MDP consists of: set of states (S), set of actions (A), transition function (t), reward function (r) and a policy (π)

Curse of dimensionality

Rapid growth of difficulty of problems (number of states *N*) with the number of dimensions evaluated (d). $N = s^d$

- RL: prosthetic arm (2 joints, 2 velocities)

  d = 4 dimensions

  s = 0.1 x 180 = 1800 equal segments

  $N = 1800^4 = 10^{13}$
- RLOC approach:

  d = 4 dimensions

  s = 6 (i.e. 180/30) equal segments

  $N = 6^4 = 2 \times 10^3$

- RL: portfolio (80 stocks over 5 year period)

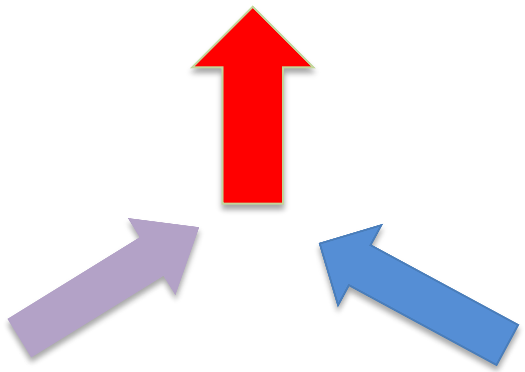
  d = 80 *x* 5 *x* 250 = 100,000 dimensions

  s = 1,000 equal segments

  $N = 1000^{100000} = (10^3)^{(10^5)} = 10^{30000}$

  number of states to be evaluated

Merge RL with OC to give RLOC



- Both methods have advantages and disadvantages. Reinforcement Learning suffers curse of dimensionality and non-linear Optimal Control methods are computationally costly.
- Therefore will merge OC and RL to benefit from efficiency of OC and adaptive qualities of RL

Application: Human Arm and Prosthetic Control

- Motivation: evidence of OC used by the brain for motor tasks and of RL used by the brain for learning
- Forward arm dynamics

$$\ddot{\theta} = M(\theta)^{-1}(\tau - C(\theta,\dot{\theta}) - B\dot{\theta})$$

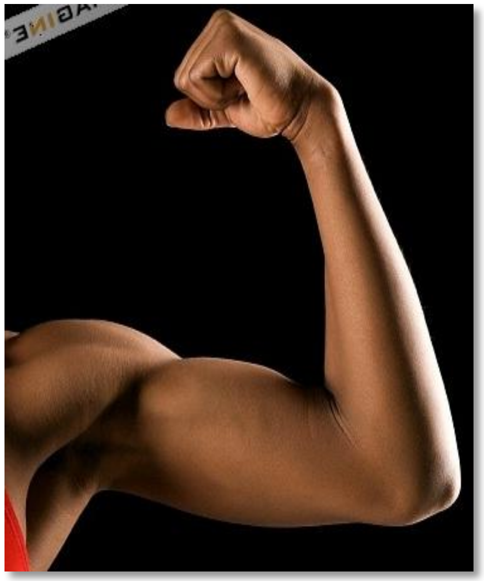
- State vector

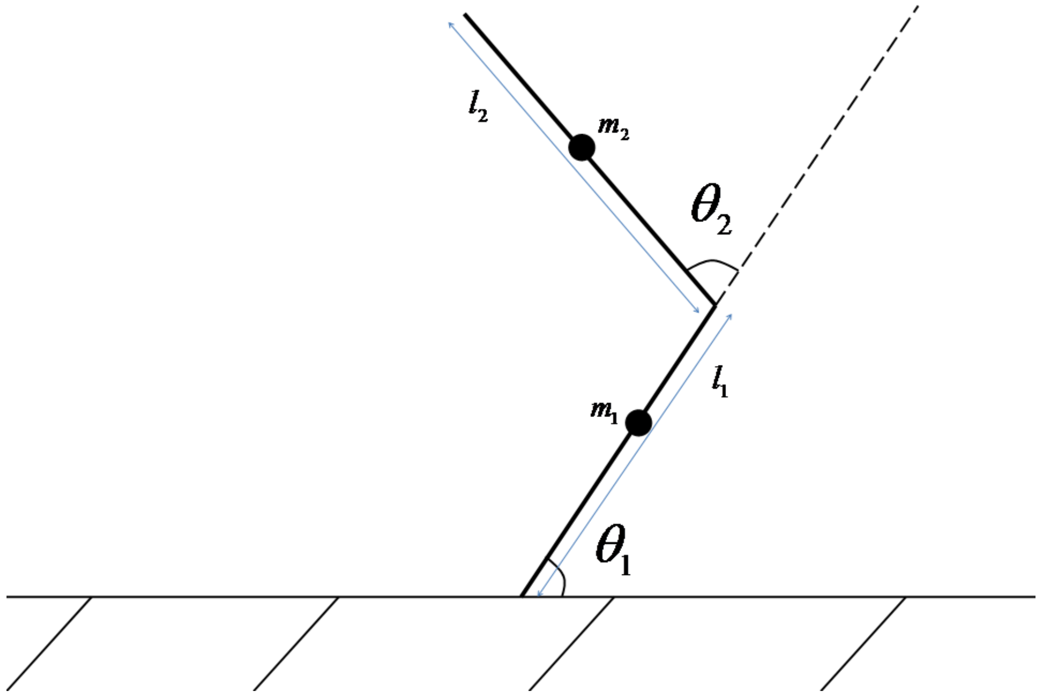
$$\mathbf{x} = (\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)^{\mathbf{T}}$$

- Arm dynamics

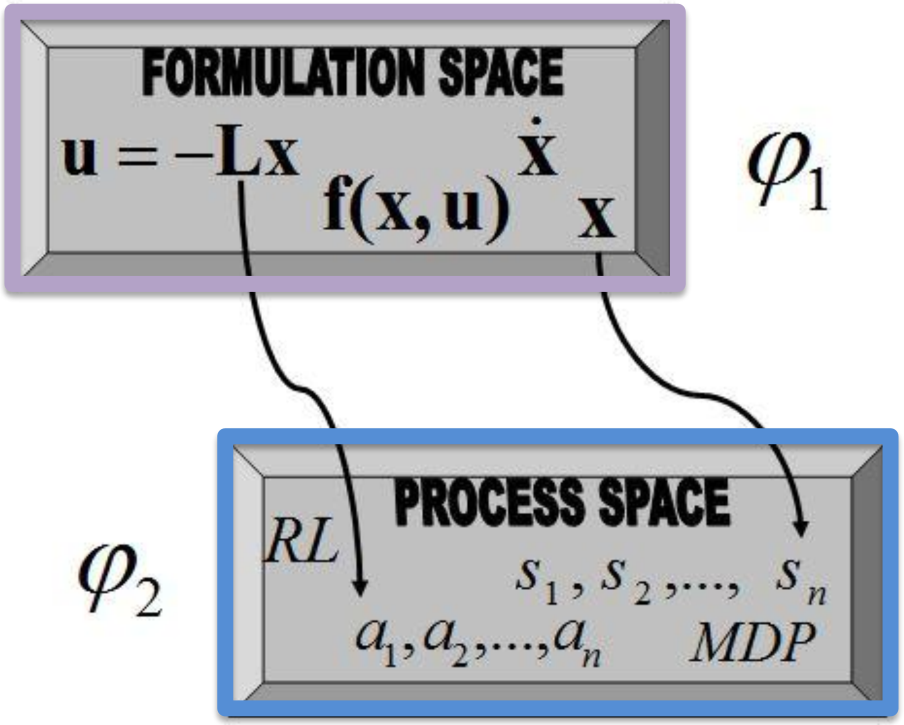
$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u}) = (\dot{\theta}_1, \dot{\theta}_2, \ddot{\theta}_1, \ddot{\theta}_2)^{\mathbf{T}}$$

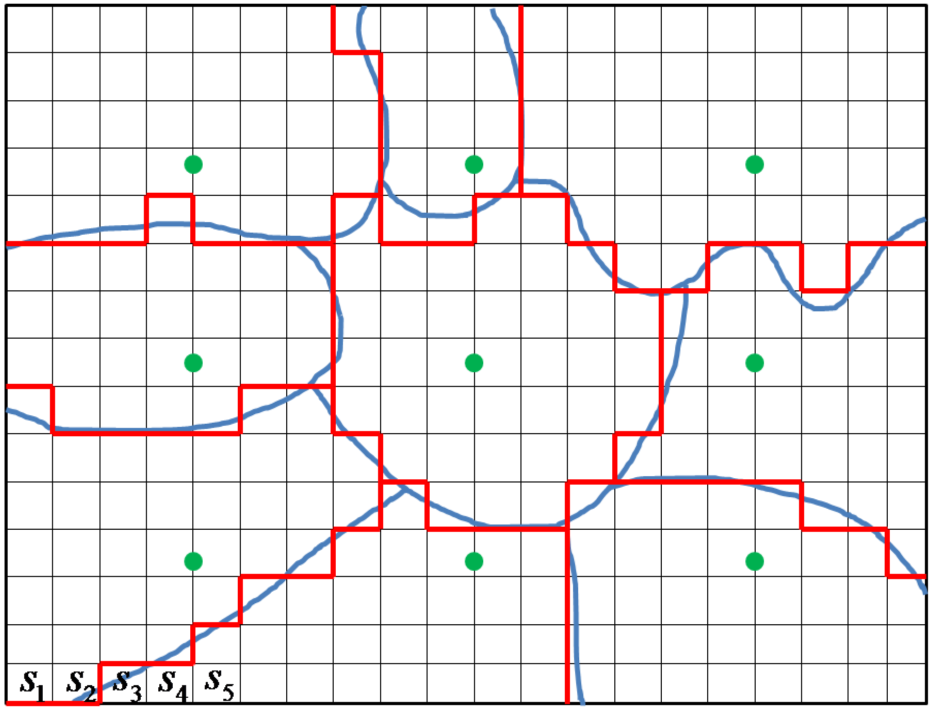
Arm Representation





Definition of dual spaces





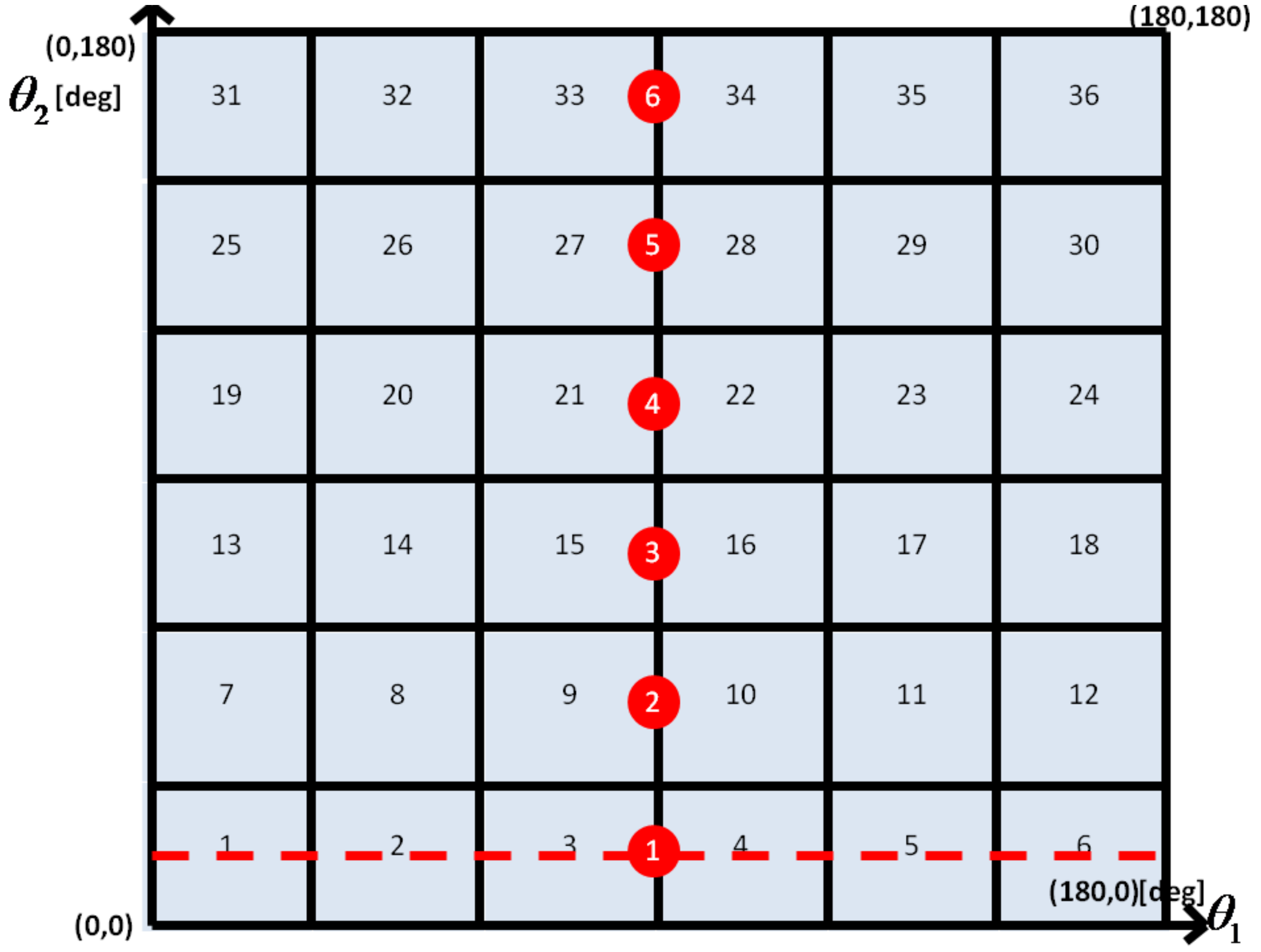
Non-Linear OC problem is linked to finite MDP:

1. Discretizing continuous Formulation Space (i.e. OC variable x) into finite number of equal Process States $s_1, s_2, \ldots, s_n$ (RL states)

2. Using Linear Quadratic Regulator (LQR) to obtain small set of localized linear optimal controllers (feedback gain matrices)

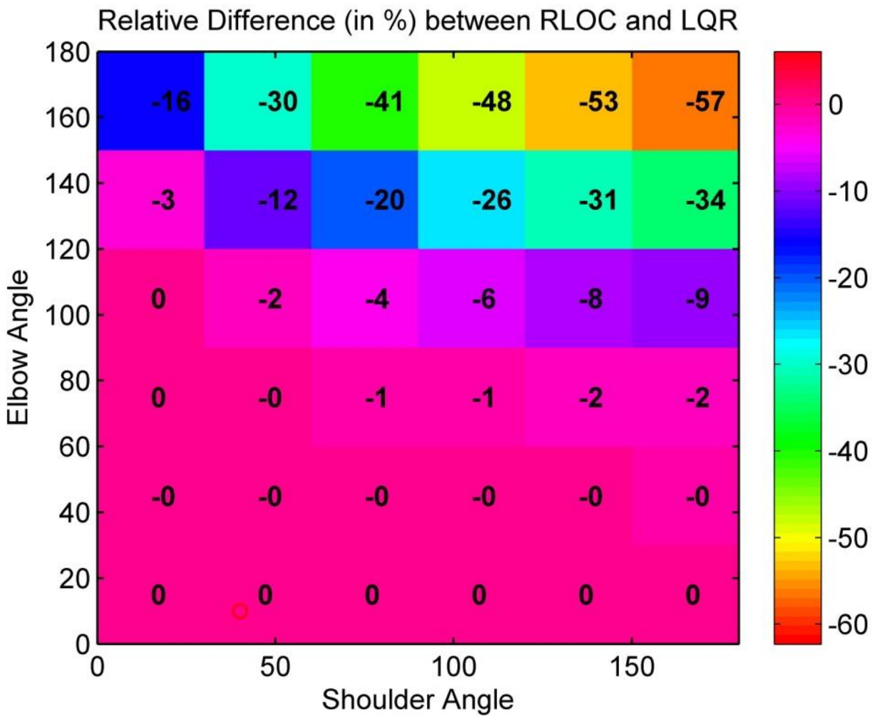
3. Using on-policy epsilon-greedy MC algorithm to learn the mapping from the Process States to controllers (where MC rewards are OC costs and MC actions are OC feedback gain matrices)

Formulation Space mapped into Process Space

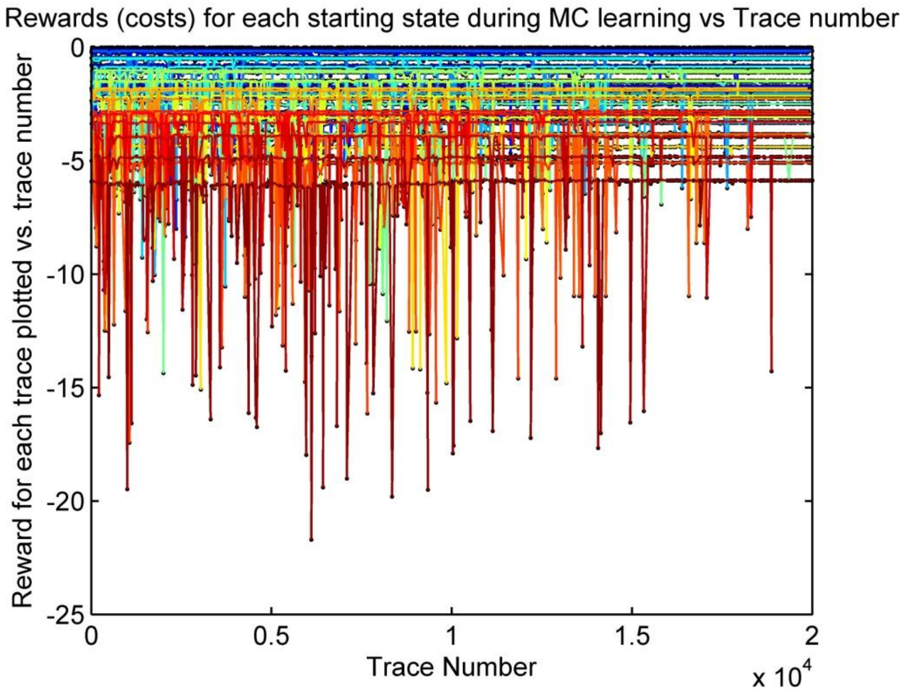


Results $m_1 = 1.4kg$ ___ $m_2 = 1.1kg$

20000 traces (300 steps per trace)



-16	-30	-41	-48	-53	-57
-3	-12	-20	-26	-31	-34
0	-2	-4	-6	-8	-9
0	-0	-1	-1	-2	-2
-0	-0	-0	-0	-0	-0
0	0	0	0	0	0



Future Work

- Improve the Reinforcement Learning part algorithm (MC iterative Q updates, Function Approximation method: can discretize velocities too)

- Apply RLOC to a more complex task: Double Inverted Pendulum

Conclusion

- RLOC algorithm performs better than LQR for an important class of non-linear control problems
- The algorithm is adaptive and can control the system from the start of the simulation, while progressively improving its performance

Thank you!