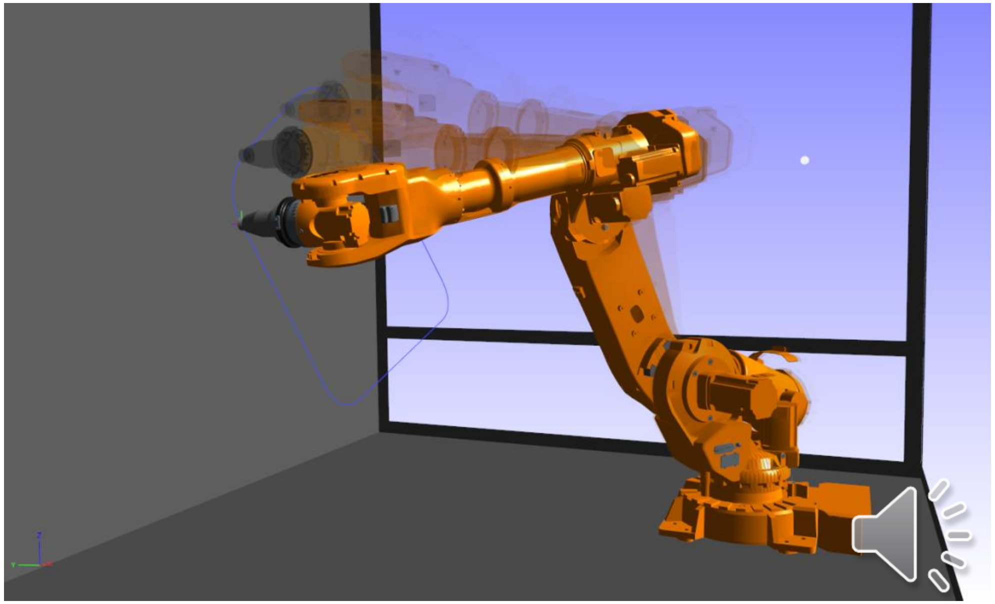
Optimal Control Theory

Robotics – COMS4045

Benjamin Rosman

Fitting things together…

- You've learned how to:
  - determine where the robot is (forward kinematics)
  - determine where you'd like it to be (inverse kinematics)
  - determine how it may move (dynamics)
- Control:
  - Open loop, closed loop
  - Model based
  - PID
- But now:
  - The **best** controller?

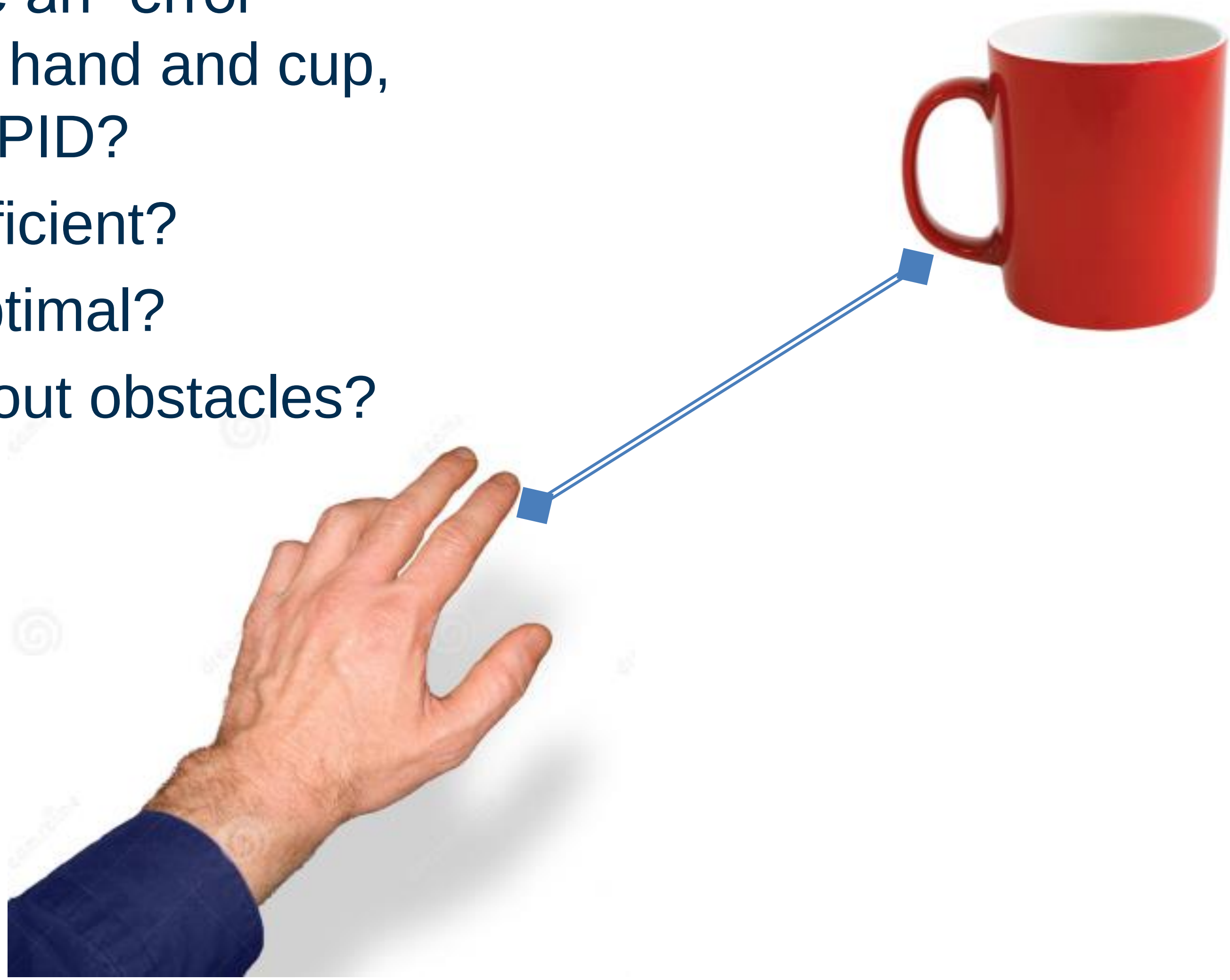


Reaching a cup

- How to reach a cup?
- (Ignore the fact that we're working in joint angles)

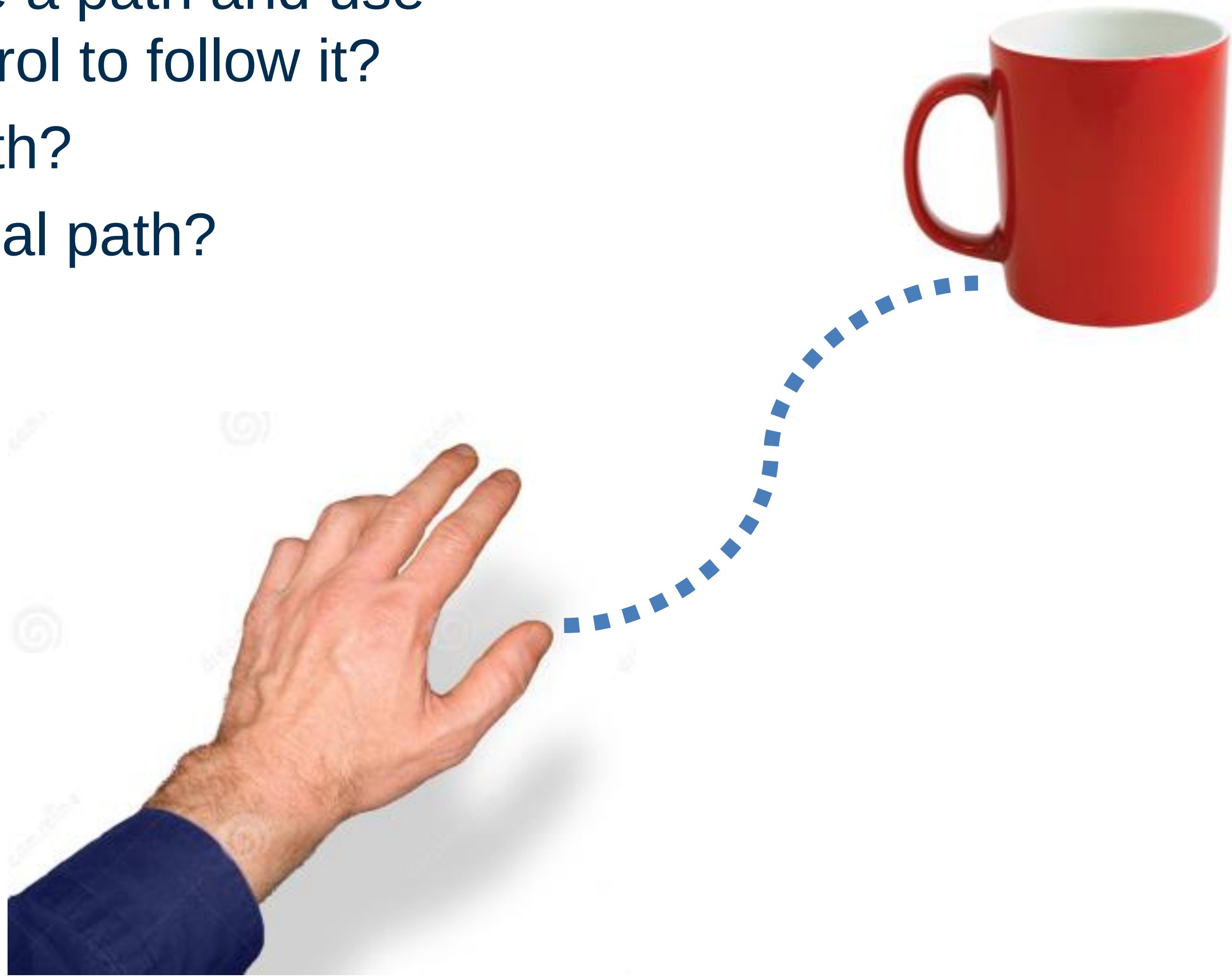
Reaching a cup

- Compute an “error” between hand and cup, and use PID?
- Is this efficient?
- Is this optimal?
- What about obstacles?



Reaching a cup

- Compute a path and use PID control to follow it?
- What path?
- An optimal path?



Reaching a cup

- What is optimality?
- Given some desired criteria, can we find an optimal control strategy?

Optimality

- An optimal controller $u^*(x,t)$ is one such that there does not exist any better $u(x,t)$
- Controlling the system: $\dot{x}(t) = f[x(t), u(t)]$
- What is “better”?
  - Achieve some goal?
    - E.g. reach a point, follow a trajectory, …
  - Achieve goal in some desired manner?
    - E.g. energy efficiently, as fast as possible, safely, …
- Wanting the best controller, implies optimisation
  - Minimise some cost

Expressing a goal

- Scalar cost function J of terminal and continuous costs
- $J = \phi[x(t_f)] + \int_{t_0}^{t_f} L[x(t), u(t)]\, dt$
- Terminal cost: how good is my final state?
  - Typically only a function of the final state (and final time)
  - E.g. $\phi[x(t_f)] = \|x(t_f) - x_{goal}\|^2$
- Integral/continuous cost: how good was my trajectory?
  - A function over the entire duration of the trajectory
  - Typically of state and control
  - E.g. $L[x(t), u(t)] = a\|x(t)\|^2 + b\|u(t)\|^2$
- Now we can compare trajectories (controllers)

The optimal control formulation

- Minimise scalar function J of terminal and continuous costs
- With respect to control $u(t)$ over time interval (t_0, t_f)
- Subject to dynamic constraints of the system

- Mathematically:

$$\min_{u(t)} J = \min_{u(t)} \left\{ \phi[x(t_f)] + \int_{t_0}^{t_f} L[x(t), u(t)]\, dt \right\}$$

- Subject to $\dot{x}(t) = f[x(t), u(t)]$, $x(t_0)$ given

Reformulating

- Augment cost function with dynamic constraints using Lagrange multiplier $\lambda(t)$ (same dimension as constraint)
- Constraint = 0 if satisfied

$$J = \phi\left[x(t_f)\right] + \int_{t_0}^{t_f} \{L[x(t), u(t)] + \lambda^T(t)[f[x(t), u(t)] - \dot{x}(t)]\}\, dt$$

- Define Hamiltonian:

$$H(x, u, \lambda) = L(x, u) + \lambda^T(t) f(x, u)$$

- Substitute:

$$J = \phi\left[x(t_f)\right] + \int_{t_0}^{t_f} \{H[x(t), u(t), \lambda(t)] - \lambda^T(t)\dot{x}(t)\}\, dt$$

Necessary conditions for optimality

- Optimal cost J^*:
$$\min_{u(t)} J = J^* = \phi\big[x\big(u(t_f)\big), t_f)\big] + \int_{t_0}^{t_f} \{H[x(u(t),t), u(t), \lambda(t)] - \lambda^T(t)\dot{x}(u(t),t)\}\, dt$$
- Consider perturbations of policy Δu giving rise to trajectories Δx
- For minimum: $\Delta J^* = 0$
- $\Delta J^* = \left\{\frac{\delta\phi}{\delta x} - \lambda^T\right\} \Delta x(\Delta u)|_{t=t_f} + [\lambda^T \Delta x(\Delta u)]|_{t=t_0} + \int_{t_0}^{t_f} \left\{\frac{\delta H}{\delta u}\Delta u + \left[\frac{\delta H}{\delta x} + \dot{\lambda}^T\right] \Delta x(\Delta u)\right\} dt = 0$
- Verify this using the chain rule and integration by parts!

Necessary conditions for optimality

- Individual terms stay zero for arbitrary variations in $\Delta x(t)$ and $\Delta u(t)$
- So:

1. $\left[\frac{\delta \phi}{\delta x} - \lambda^T\right]|_{t=t_f} = 0$
2. $\left[\frac{\delta H}{\delta x} + \dot{\lambda}^T\right] = 0$
3. $\frac{\delta H}{\delta u} = 0$

- Three necessary conditions for optimality
- Defines analytic solution as a BVP

Example

$$H(x,u,\lambda) = L(x,u) + \lambda^T(t) f(x,u)$$

1. $\left[\frac{\delta\phi}{\delta x} - \lambda^T\right]|_{t=t_f} = 0$
2. $\left[\frac{\delta H}{\delta x} + \dot{\lambda}^T\right] = 0$
3. $\frac{\delta H}{\delta u} = 0$

- Find u^* to minimise $J = (x_1(2) - 0.2)^2 + x_2^2(2) + \int_0^2 u^2\, dt$
- S.t. $\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u$, and $x_1(0) = 0, x_2(0) = 0$

Example

$$H(x,u,\lambda) = L(x,u) + \lambda^T(t) f(x,u)$$

1. $\left[\frac{\delta \phi}{\delta x} - \lambda^T\right]\Big|_{t=t_f} = 0$
2. $\left[\frac{\delta H}{\delta x} + \dot{\lambda}^T\right] = 0$
3. $\frac{\delta H}{\delta u} = 0$

- Find u^* to minimise $J = (x_1(2) - 0.2)^2 + x_2^2(2) + \int_0^2 u^2\, dt$
- S.t. $\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u$, and $x_1(0) = 0, x_2(0) = 0$
- $\phi(x) = (x_1(2) - 0.2)^2 + x_2^2(2)$
- $H = L + \lambda^T f = u^2 + (\lambda_1 \quad \lambda_2)\left[\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u\right]$

  $= u^2 + x_2\lambda_1 + u\lambda_2$

1. $\left[\frac{\delta \phi}{\delta x} - \lambda^T\right]\Big|_{t=2} = 0 = (2(x_1 - 0.2) - \lambda_1 \quad 2x_2 - \lambda_2)|_{t=2}$
2. $\left[\frac{\delta H}{\delta x} + \dot{\lambda}^T\right] = 0 = \begin{pmatrix} 0 \\ \lambda_1 \end{pmatrix} + \begin{pmatrix} \dot{\lambda}_1 \\ \dot{\lambda}_2 \end{pmatrix}$
3. $\frac{\delta H}{\delta u} = 0 = 2u + \lambda_2$

Example

- $\dot{x}_1 = x_2, \quad \dot{x}_2 = u$
- $2(x_1(2) - 0.2) - \lambda_1(2) = 0$
- $2x_2(2) - \lambda_2(2) = 0$
- $\dot{\lambda}_1 = 0$
- $\dot{\lambda}_2 + \lambda_1 = 0$
- $2u + \lambda_2 = 0$
- $x_1(0) = 0, x_2(0) = 0$

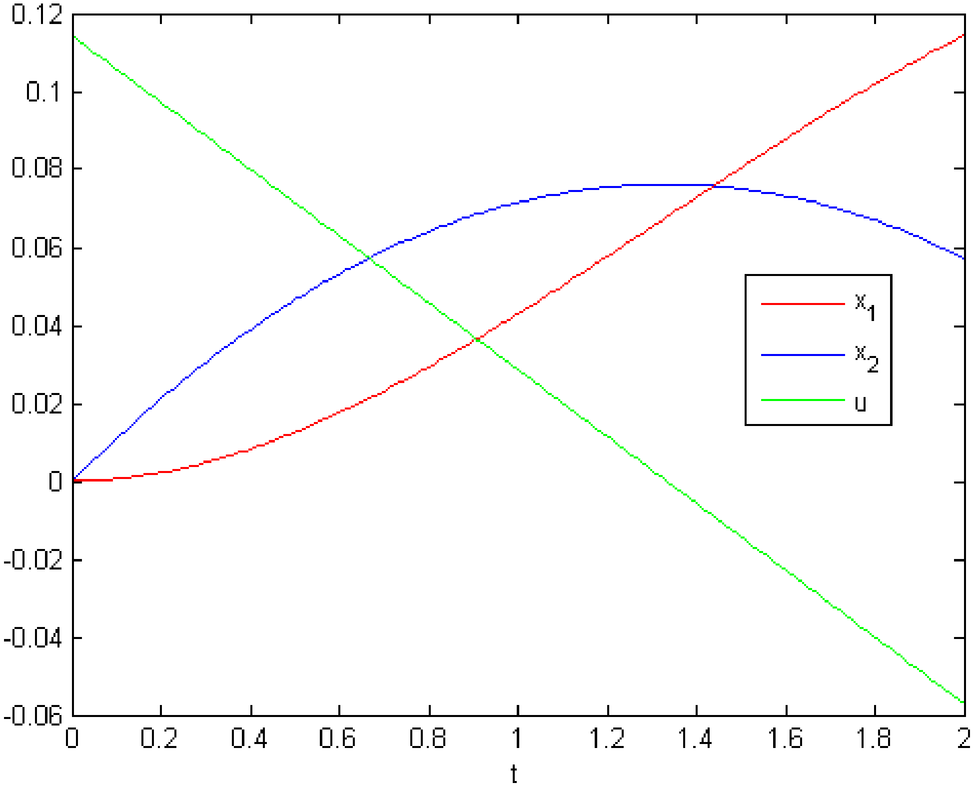
So:

- $\lambda_1 = c$
- $\lambda_2 = -ct + d$
- $u = \frac{ct-d}{2}$
- $\dot{x}_2 = \frac{ct-d}{2}$
- And so $x_2 = \frac{c}{4}t^2 - \frac{dt}{2} + e$
- $x_1 = \frac{c}{12}t^3 - \frac{d}{4}t^2 + et + f$

Example

Solving (using boundary conditions) gives:

$$c = -0.1714, d = -0.2286, e = 0, f = 0$$



Importance of weighting, e.g.,

$$J = A\phi[x(t_f)] + B\int_{t_0}^{t_f} L[x(t), u(t)]\,dt$$

And ratio $\frac{A}{B}$

Numerical solutions

- One approach:
- Assume a functional form for u:
  - E.g. $u = a_0 + a_1 t + a_2 t^2$
- Solve for parameters a_i to optimise J
  - Could use a grid search
  - Rather use a built-in optimisation routine

Numerical solutions

- Define cost_function(u)
  - Take an input u (or parameter vector a)
  - Solve ODE using numerical solver to return trajectory x(t)
    - E.g. In Matlab use *ode45()*
    - E.g. In Python use *ode()* in *scipy.integrate*
  - Compute J using u, x, t
  - Return scalar J
- Optimise cost_function using some initial a
  - E.g. In Matlab use *fminunc()*
  - E.g. In Python use *minimize()* in *scipy.optimize*

Linear Quadratic Regulator

- Special case
  - Objective function quadratic in x and u
    $$J = \frac{1}{2} x(t_f)^T S_f\, x(t_f) + \frac{1}{2}\int_{t_0}^{t_f} \left(x(t)^T Q\, x(t) + u(t)^T R\, u(t)\right) dt$$
  - Dynamics are linear
    $$\dot{x}(t) = Ax(t) + Bu(t), \qquad x(t_0) = x_0$$
- Feedback law: linear quadratic regular (LQR)
- Also useful for non-LQ systems in neighbourhood of desired state x^*
  - Linearise dynamics around x^*
  - 2nd order approx of J around x^*
  - Use LQR as local controller

LQR

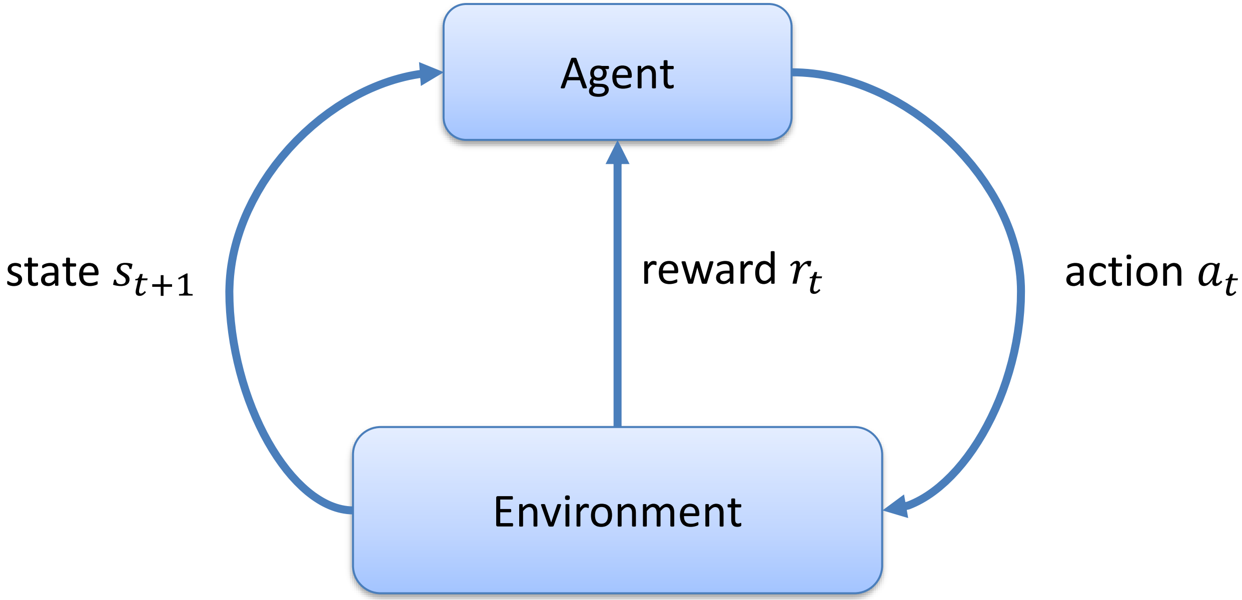
- In this case, optimal control law is linear state feedback:
  - $u(t) = -K(t)x(t)$
- State feedback (gain matrix K) given by:
  - $K(t) = R^{-1}B^T S(t)$
- $S(t)$ is the solution to the differential Ricatti equation:
  - $-\dot{S} = A^T S + SA - SBR^{-1}B^T S + Q$
  - $S(t_f) = S_f$
  - Usually solved numerically

Model learning

- What if I don't know my system dynamics or cost function?
- Reinforcement learning
  - f = transition function
  - J = -reward function
- Exploration vs exploitation
  - The multi-armed bandit problem
  - Explore: obtain new samples from f and J
  - Exploit: use approximate optimal controller, so as to improve costs (sample at a later point)

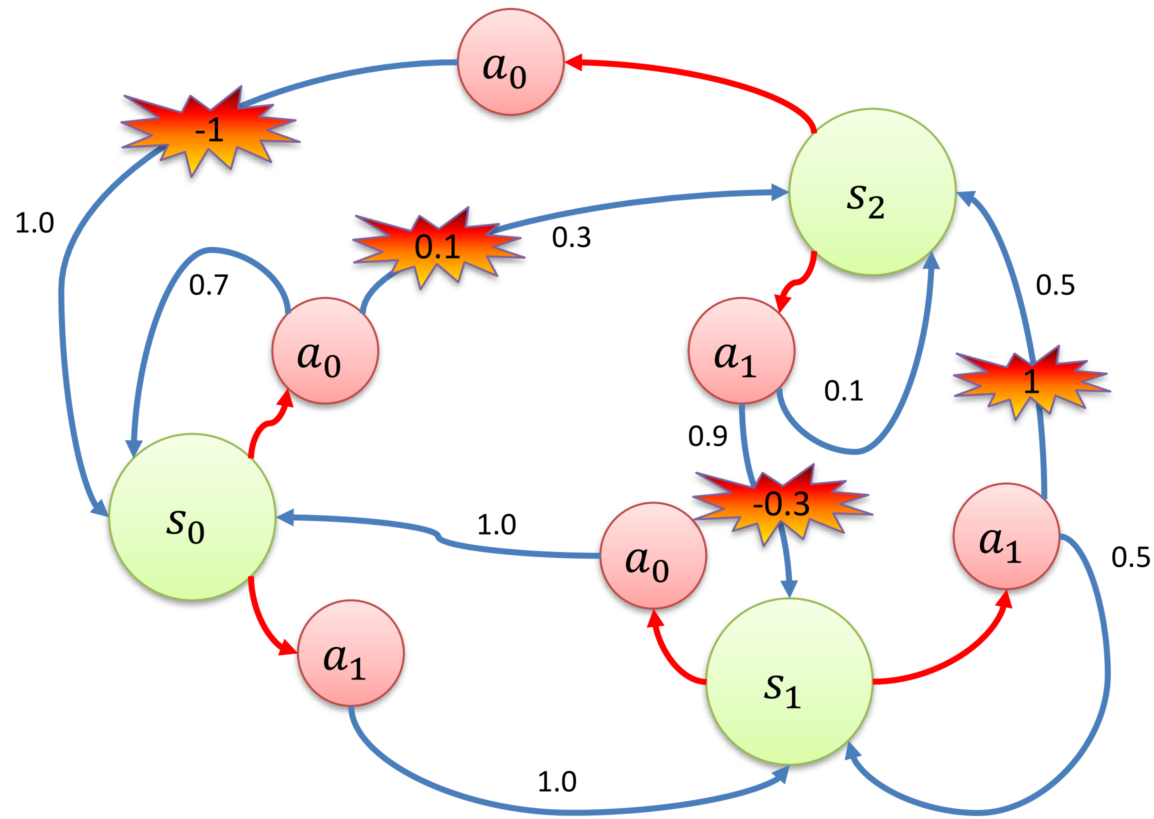
Reinforcement Learning

- Learn to **map situations to actions** so as to **maximise numerical reward** (which may be delayed)
- Difference is that the models are **unknown**. Either:
  - Estimate the models, and then solve (model-based RL)
  - Just learn the best action in each state (model-free RL)

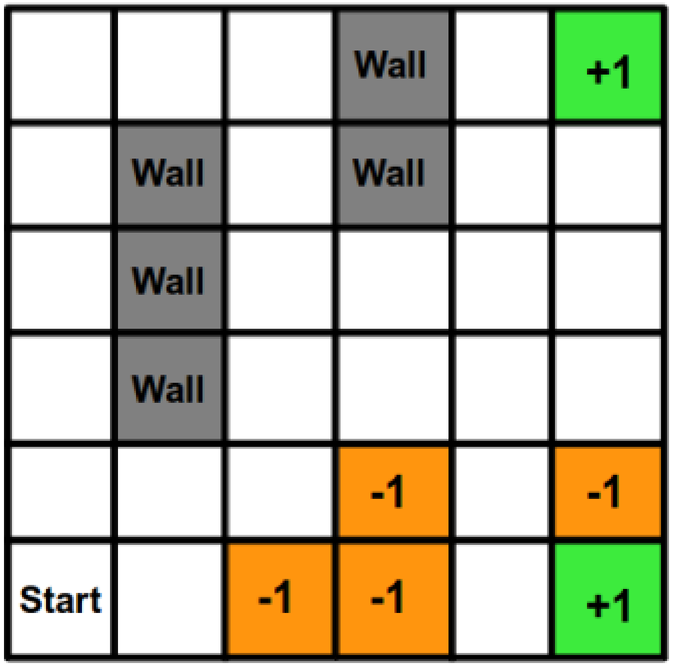


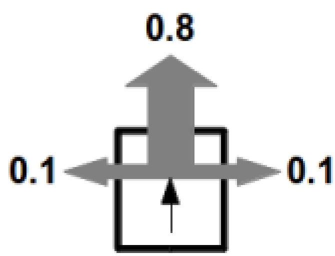
Markov Decision Processes (MDPs)

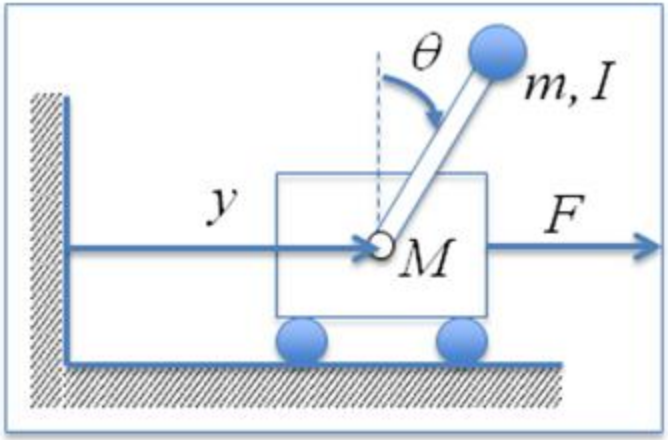
- Model a decision problem
- Usually discrete system
- $M = \langle S, A, T, R, \gamma \rangle$
- Observable
- Markov
- Policy π



Examples







Multi-agent systems

- What if the system is out to get me (an adversarial agent or environment)?

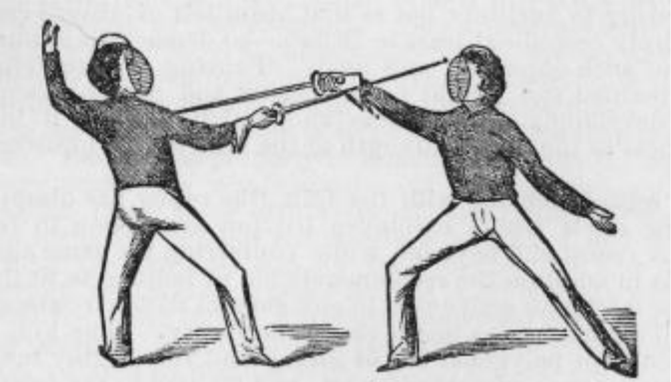
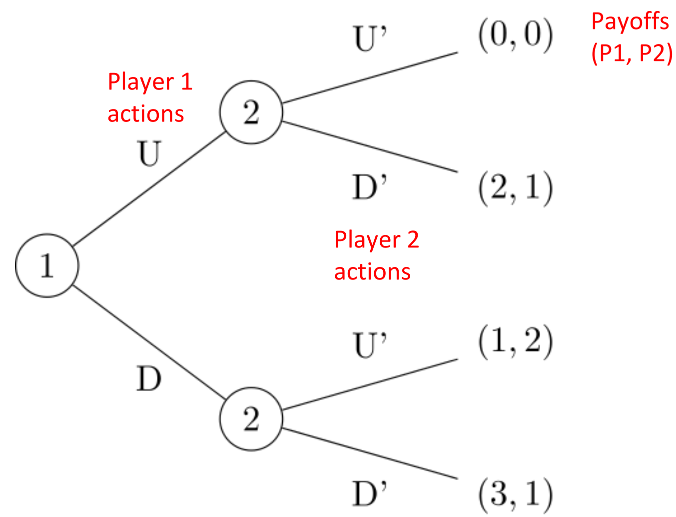


Fig. 1.—Carte.

- Game theory
  - Study of strategic decision making
  - Models of conflict and cooperation between rational decision makers

Game Theory – Basics

- Discrete games
- Consider two players
  - Player 1 maximises
  - Player 2 minimises
- Types of games:
  - Extensive form
  - Normal form
- Can be:
  - Cooperative
  - Adversarial
  - Zero-sum
  - …



	Player 2 chooses *Left*	Player 2 chooses *Right*
Player 1 chooses *Up*	4,3	-1,-1
Player 1 chooses *Down*	0,0	3,4

Payoffs

Player1 chooses Rows
Player2 chooses Columns

Solution Ideas

- Nash equilibrium: no player can benefit by unilaterally deviating
- E.g. assume column player is maximising:
- $$\begin{pmatrix} 1 & 3 & -1 & 2 \\ -3 & -2 & 2 & 1 \\ 0 & 2 & -2 & 1 \end{pmatrix}$$

This is a zero-sum game: what one player loses the other gains. So, a payoff of 1 implies (1, -1)

Solution Ideas

- Nash equilibrium: no player can benefit by unilaterally deviating
- E.g. assume column player is maximising:
- $\begin{pmatrix} 1 & 3 & -1 & 2 \\ -3 & -2 & 2 & 1 \\ 0 & 2 & -2 & 1 \end{pmatrix}$

Dominance

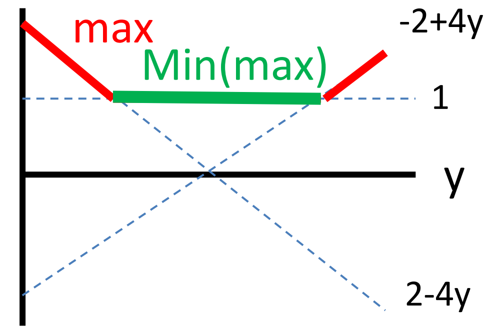
Solution Ideas

- Nash equilibrium: no player can benefit by unilaterally deviating
- E.g. assume column player is maximising:
- $$\begin{pmatrix} 1 & 3 & -1 & 2 \\ -3 & -2 & 2 & 1 \\ 0 & 2 & -2 & 1 \end{pmatrix}$$

Dominance

Solution Ideas

- $\begin{pmatrix} -2 & 2 & 1 \\ 2 & -2 & 1 \end{pmatrix}$
- Assume P1 picks row 1 w.p. y
- Payoffs to P2:
  – Column 1: -2y + 2(1-y) = 2 – 4y
  – Column 2: 2y - 2(1-y) = -2 + 4y
  – Column 3: y+(1-y) = 1



- P2 will choose to max this, so P1 will choose to min this max
- i.e. P1 will minimise max{2-4y, -2+4y, 1}
- This gives the min of the max is 1, when $y \in [\frac{1}{4}, \frac{3}{4}]$
  – And P2 chooses the last column

Differential Games

- Can have continuous games, with dynamics as differential equations
  - Differential games
  - Model dynamic adversarial situations
- E.g.
  - The homicidal chauffeur problem