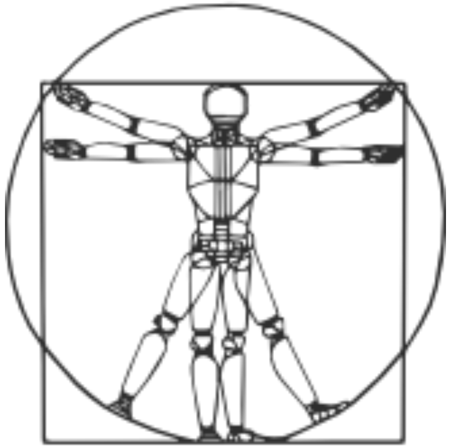
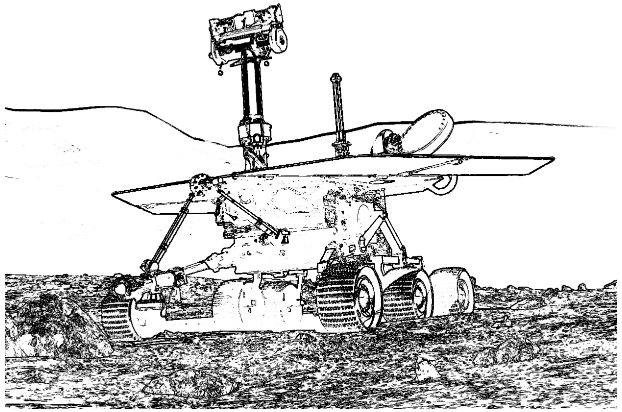




COMP417 - Intro to Robotics

Optimal Control

Fall 2019

David Meger

McGill

MRL Mobile Robotics Lab at McGill University

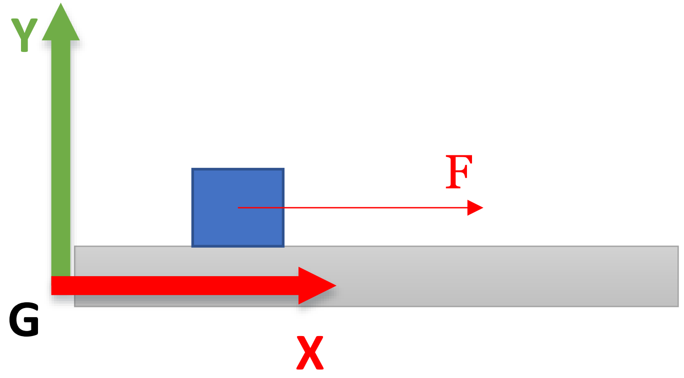
Intro to Optimal Control

- Rather than having a reference trajectory, suppose we have a cost function that tells us just how bad each “wrong” action is
  - Achieving goal can still be the best (least cost)
  - Now we can prioritize different mistakes
- This allows us to ***think ahead in time*** and solve for trajectories that minimize the sum of costs as we formulate their matching control

The state of a double integrator (block on ice)

$$\mathbf{x} = \left[{}^{G}p_x \right]$$

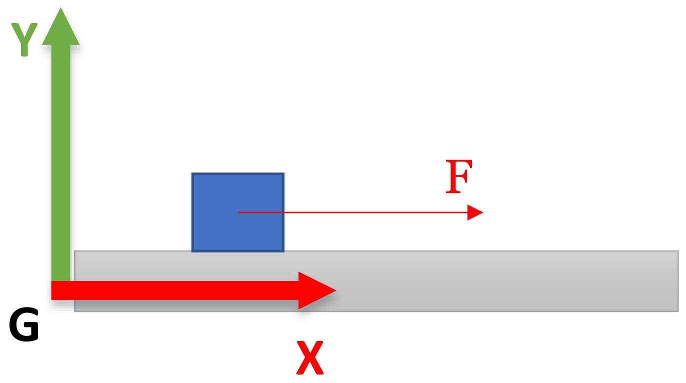
State = [Position along x-axis]



Controls of a double integrator

$$\mathbf{u} = \left[{}^{G}u_x\right]$$

Controls = [Force along x-axis]

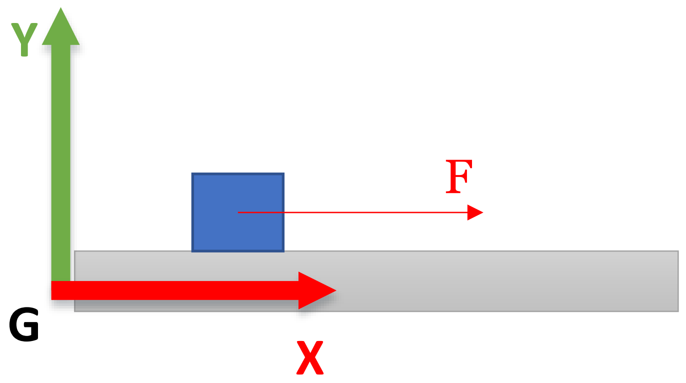


Dynamics of a double integrator

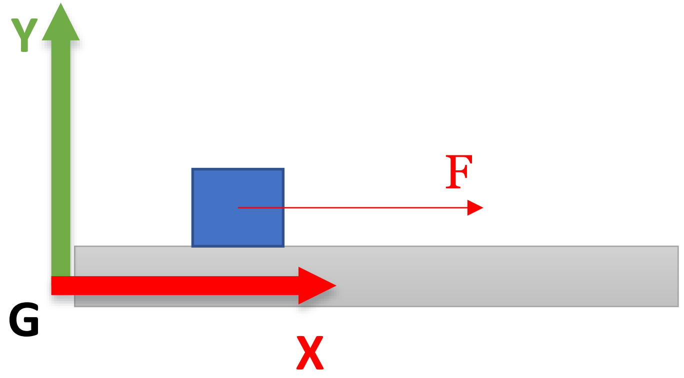
Passive dynamics correspond to taxi sliding on the roads of Montreal (dirty ice). Where is the car going to end up? Similar to curling.

$$F = ma$$

$$-k\dot{x} = \mathrm{m}\ddot{x}$$



Dynamics of a double integrator



Passive dynamics correspond to taxi sliding on the roads of Montreal (dirty ice). Where is the car going to end up? Similar to curling.

$$F = ma$$

$$-k\dot{x} = \mathrm{m}\ddot{x}$$

Controlled dynamics:

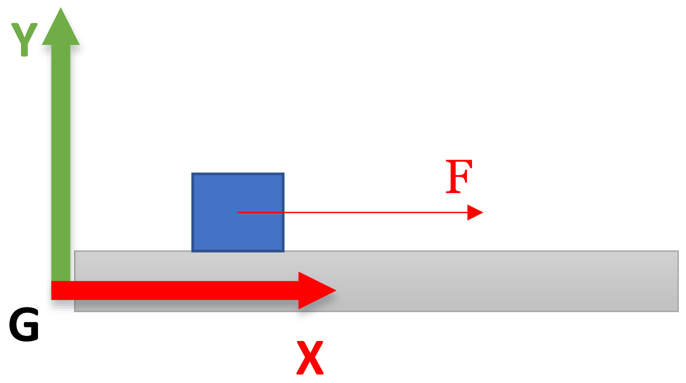
$$\mathrm{m}\ddot{x} = \mathrm{u} - k\dot{x}$$

What controller achieves $\ddot{x}_{ref}$?

$$u_{opt} = \frac{\ddot{x}_{ref} + k\dot{x}}{m}$$

If we assume a known reference acceleration, this is enough. Let's add some challenge!

Dynamics of a double integrator



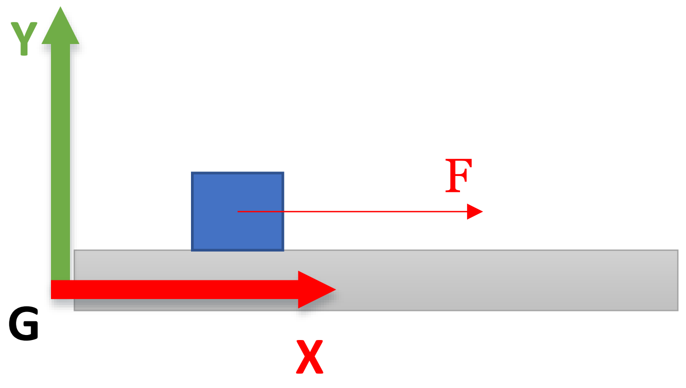
Controlled dynamics interesting when force is limited.

$$\ddot{x} = \mathrm{F} - k\dot{x}$$
$$\text{s.t. } |\mathrm{F}| < u_{max}$$

Can we design controllers that achieves $x_{goal} = 0$?

- As quickly as possible (get to your meeting on time)
- Without over-shooting (suppose the taxi is parking at your front door!)
- Using as little energy as possible (gas costs)

Phase plots: A useful analytical tool



Graph how the system evolves in terms of x and $\dot{x}$

- This lets us start to reason about what's possible and hints at the ideas we'll use for important control algorithms

For the idealized block on ice (no friction for a moment):

$$\ddot{x} = \mathrm{u}, \text{ let } \mathrm{u} = -1$$

$$\dot{x}(t) = \dot{x}(0) - t$$

$$x(t) = x(0) + t\dot{x}(0) - \frac{1}{2}t^2$$

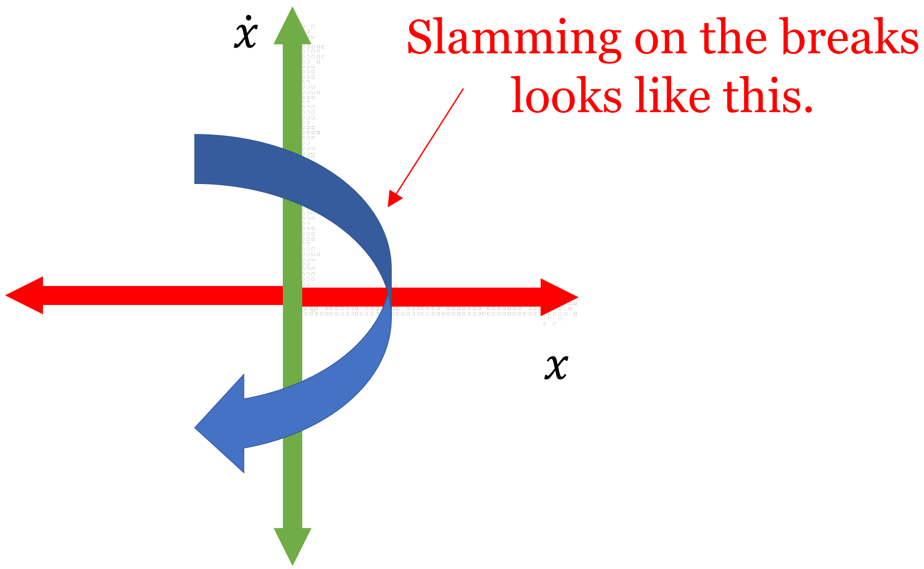
Solve for t in $\dot{x}$ equation: $\mathrm{t} = \dot{x}(0) - \dot{x}(t)$

Substitute into x equation

$$x(t) = x(0) + (\dot{x}(0) - \dot{x}(0))\dot{x}(0) - \frac{1}{2}(\dot{x}(0) - \dot{x}(t))^2$$

Note this is quadratic in $\dot{x}$, linear in x. Form:

$$\dot{x}(t)^2 = a\, x(\mathrm{t}) + \mathrm{b}$$



Phase Plot: u=0

Phase Plot: u=+1

Phase Plot: u=-1

An Intuitive Result

- Claim: Bang-bang controller is optimal for the shortest time problem
  - Why?

An Intuitive Result

- Claim: Bang-bang controller is optimal for the shortest time problem
  - Why?
  - Consider any controller that “pushes” less hard at first. Think about its phase portrait. This will, on average, be slower than the bang-bang. Not optimal.

An Intuitive Result

- Claim: Bang-bang controller is optimal for the shortest time problem
    - Why?
    - Consider any controller that “pushes” less hard at first. Think about its phase portrait. This will, on average, be slower than the bang-bang. Not optimal.
    - Consider any controller that “brakes” less hard at the critical point. It will overshoot the goal and have to come back around. This will have a faster velocity sometimes, but take longer overall
        - (harder to prove precisely, but you can do it as an exercise)

Optimal Control

- Formulate control problem as optimization of a cost function given some form of knowledge about the system

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$

$$g(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T Q \mathbf{x}_t + \mathbf{u}_t^T R \mathbf{u}_t$$

$$\underset{u_0,\ldots,u_N}{\operatorname{argmin}} \quad \sum_{t=0}^{t=N} g(\mathbf{x}_t, \mathbf{u}_t)$$

- Ideal solution for a robot is over continuous x, u, but we could discretize those to make some progress first off…

Dynamic Programming for Control

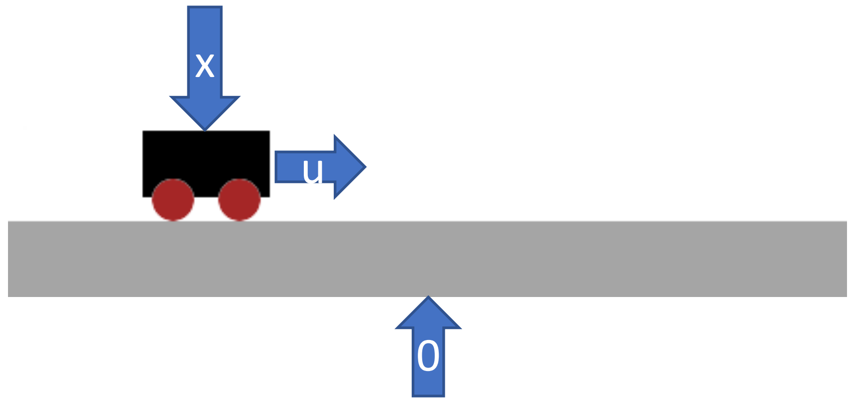
- Dynamic Programming (value iteration)
  - Discretize state-space, time and controls
  - Form discrete transition matrix that approximates dynamics, f
  - Form discrete cost that approximates cost, g
- Initialize and update optimal value function (cost-to-go) over states:
  - $J^*(x) = \min_u [g(x,u) + J^*(f(x,u))\,]$
- Important claims:
  - Knowing J^* everywhere gives us the optimal controller

$$u^* = \ arg\min_u \left[g(x,u) + {J^*}_{f(x,u)} \right]$$

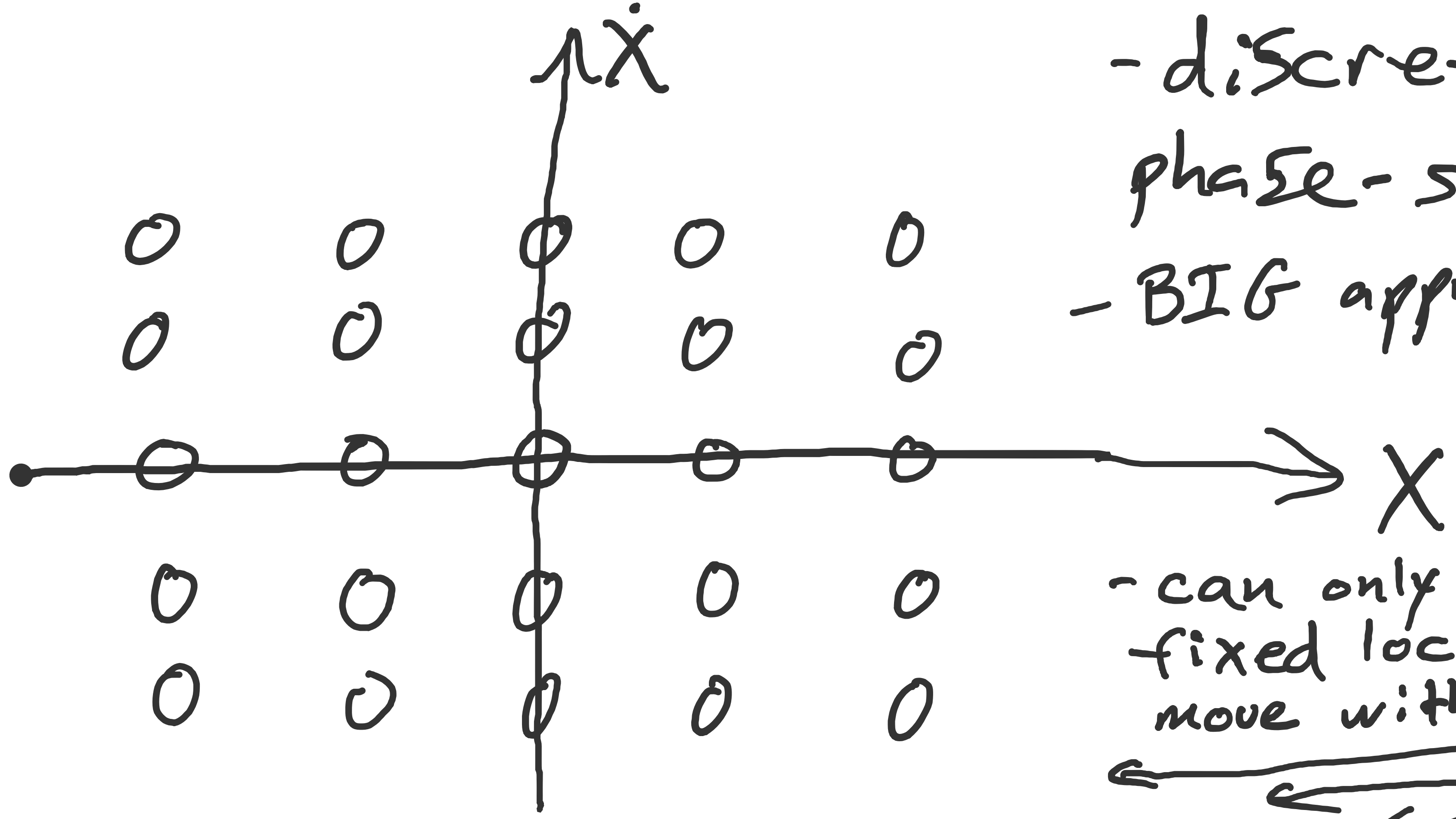
  - Knowledge of J^* in next states makes it easy to improve J^* locally, and a simple algorithm that makes an initial guess and updates everywhere converges! (Value Iteration)

Double Integrator Example

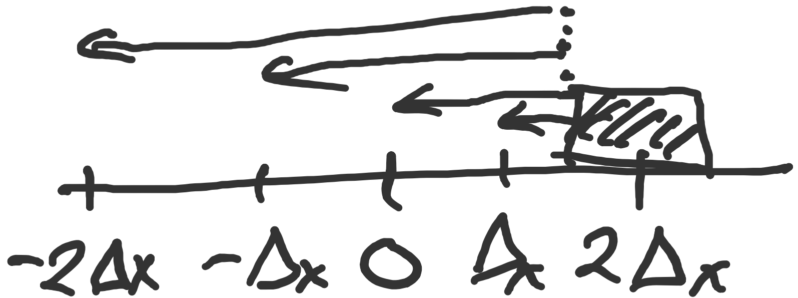
- Goal: arrive at x=0 as soon as possible
- Control a(t)=u, limited to |u|<=1
- Ideally solve over all x(0), v(0)
- Dynamics:
  - $v(t) = v(0) + ut$
  - $x(t) = x(0) + v(0)t + 0.5t^2$
- Cost (min-time):
  - g(x,u) = 0 if goal, else 1
- What is the intuitive solution?
- What does dynamic programming look like?



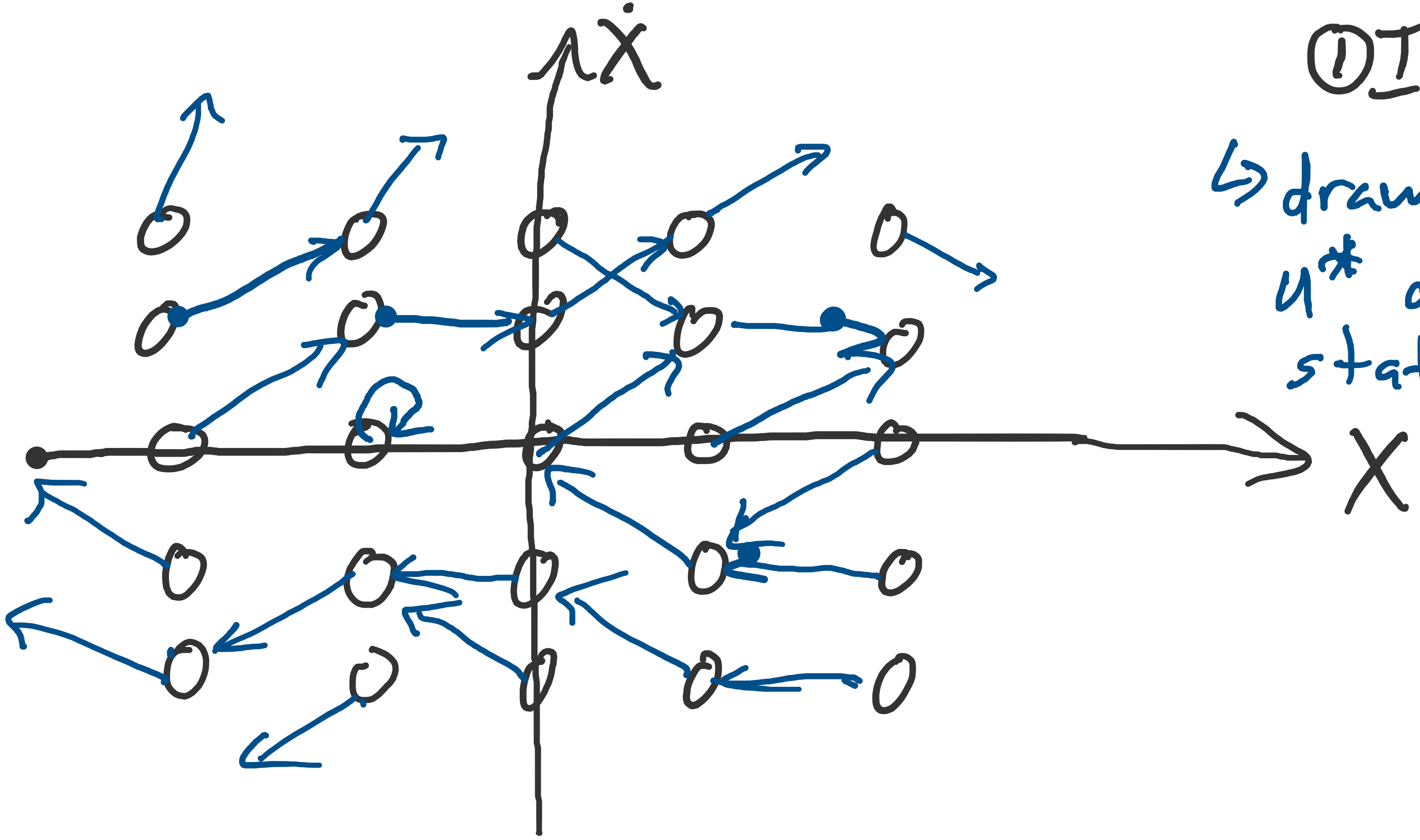
Value Iteration for the Double Integrator



- discretize phase-space
- BIG approximation.
- can only occupy fixed locations and move with fixed $\dot{x}$'s



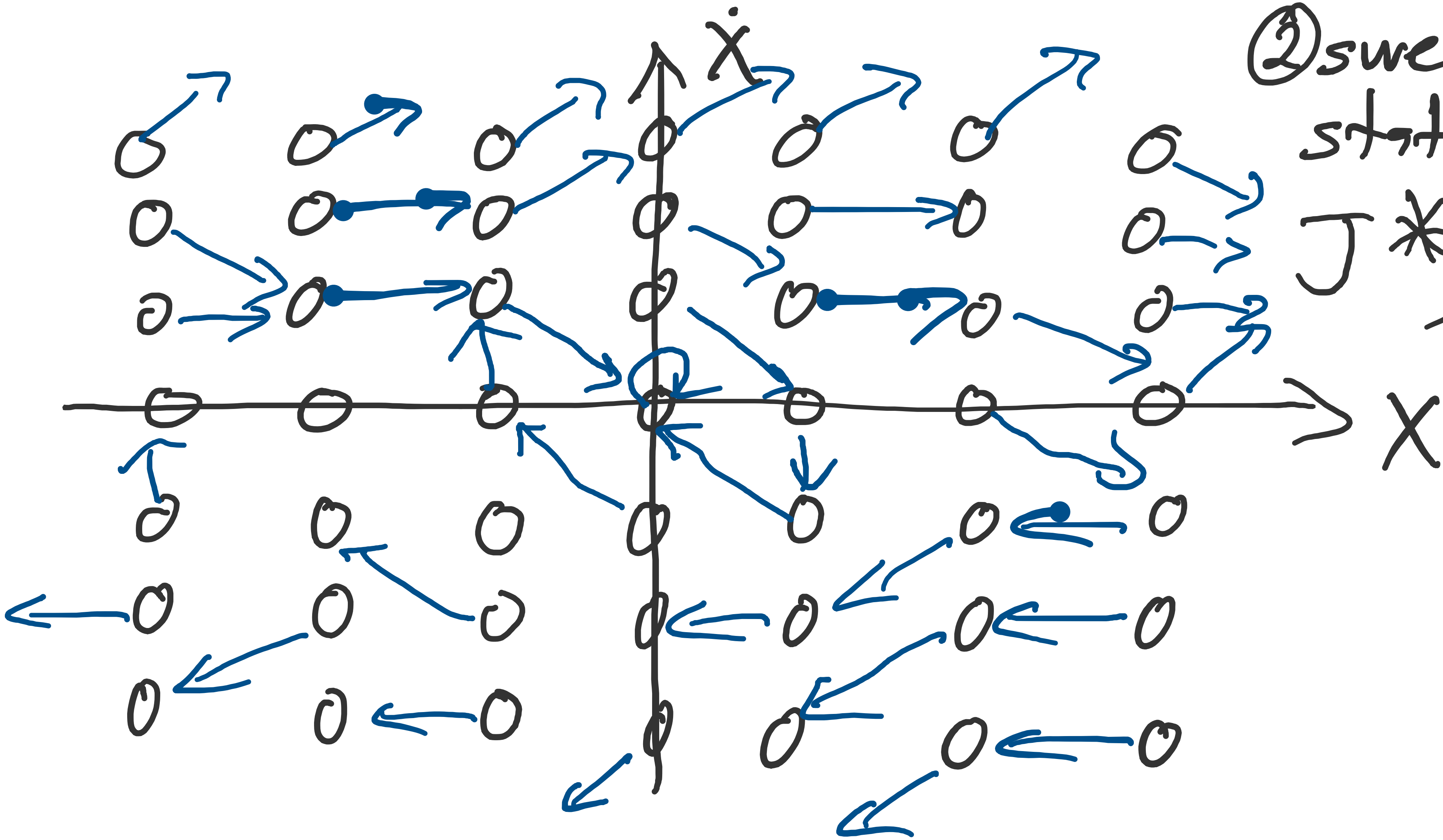
Value Iteration for the Double Integrator



① Init $J(x)$

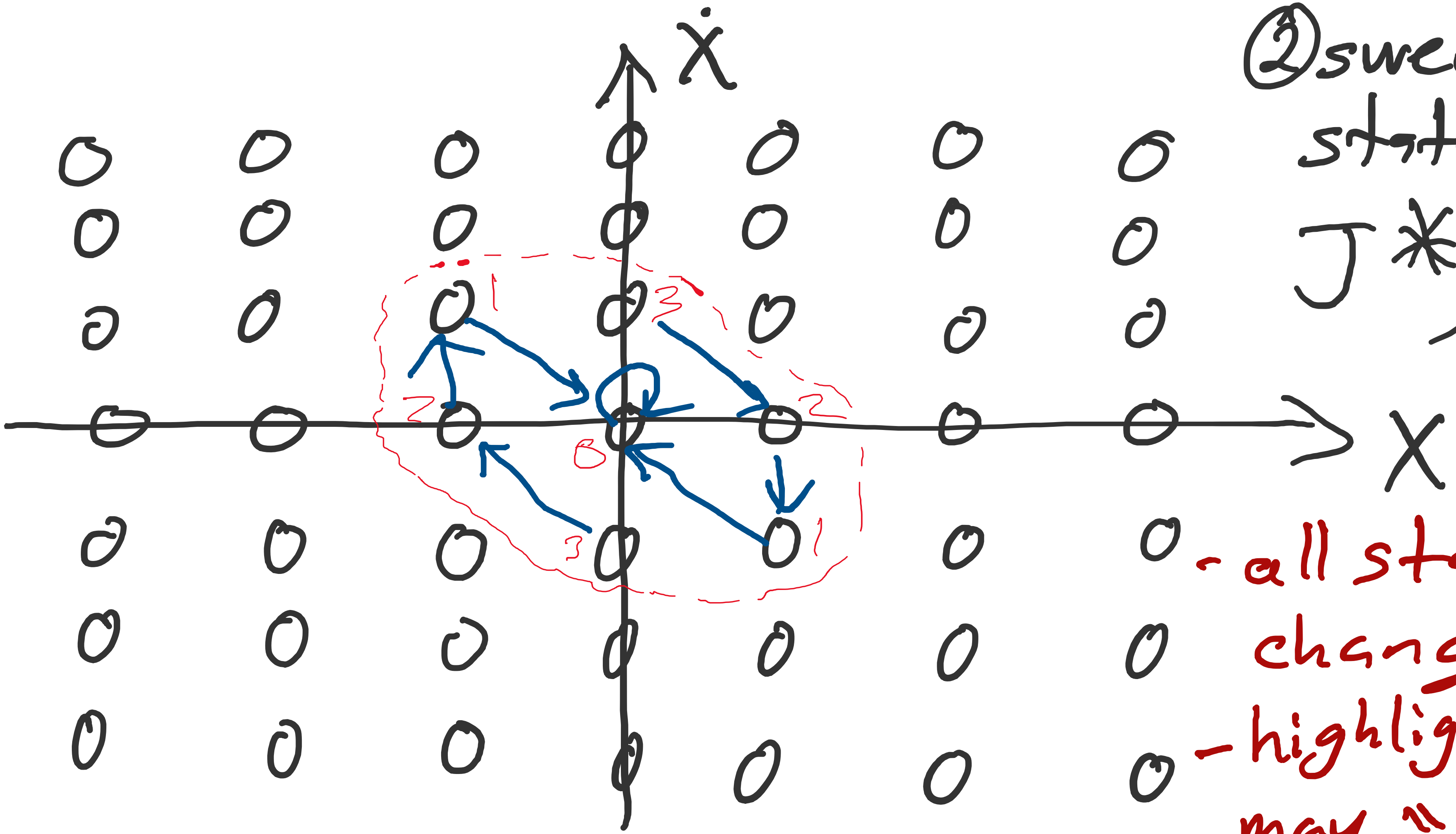
↳ draw implied u^* at each state.

Value Iteration for the Double Integrator



② sweep over states, compute J^*, update

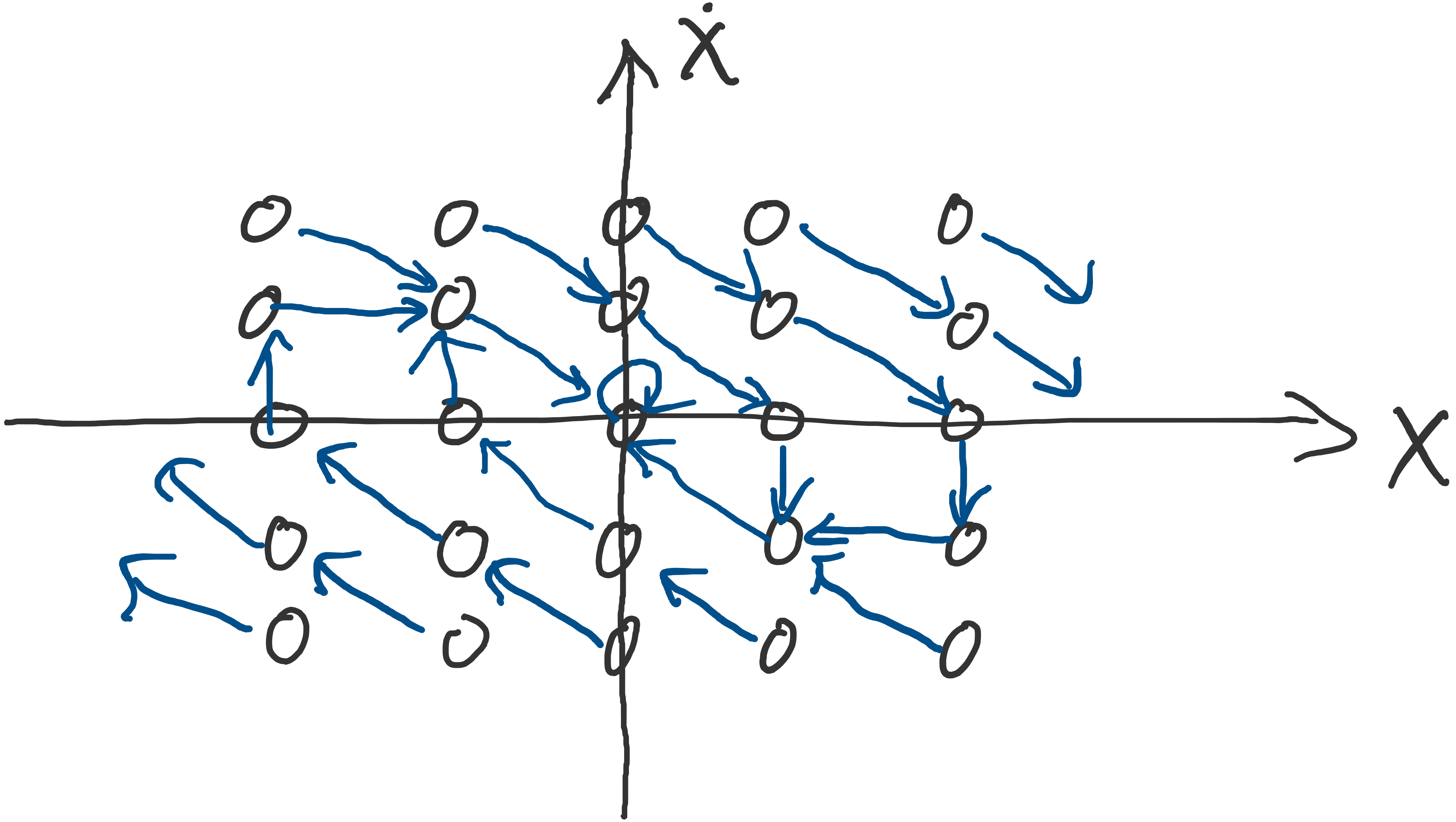
Value Iteration for the Double Integrator



② sweep over states, compute J^*, update

- all states change
- highlighted section may "converge"

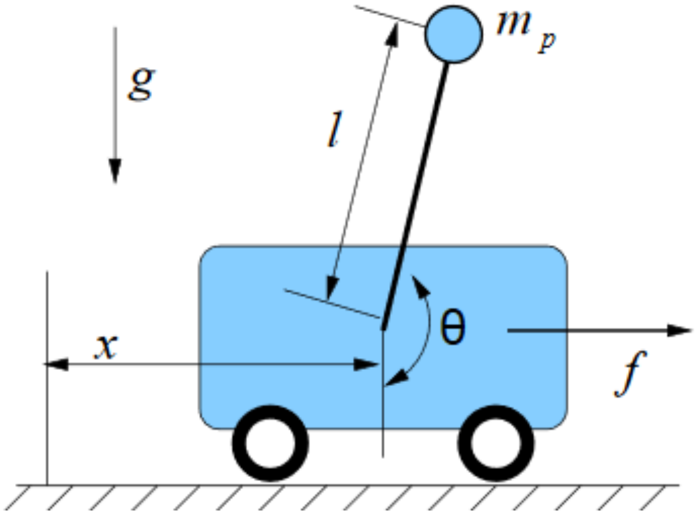
Value Iteration for the Double Integrator



Issues with dynamic programming

- Errors in formulation are related to "size" of discretization
- Naïve discretization scales exponentially with # of dimensions
- Better ways to discretize:
  - Multi-resolution and adaptive grids
  - Trajectory-based methods
- Improving robustness of solution to local errors:
  - 2nd derivative methods
  - Line search
  - Guiding prior information

State and control of a cartpole



$$\mathbf{x} = [{}^G p_x, {}^G \dot{p}_x, {}^G \theta, {}^G \dot{\theta}]$$

State = [Position and velocity of cart, orientation and angular velocity of pole]

$$\mathbf{u} = [f]$$

Control = [Horizontal force]

Cartpole properties

- Theta joint lacks a motor making this system underactuated
- We must sometimes sacrifice desirable cart position in order to "catch" the pole and right it
- This coupling comes from the dynamics equations
- Two canonical tasks:
  - Swing-up
  - Balancing