Welcome!
#pod-031
Week #2, Day 4

(Reviewed by: Deepak)

neuromatch academy

facebook Reality Labs

Tutorial #1
Explanations

Objective

Overall goals:

1. *Implement a binary control task: a Partially Observable Markov Decision Process (POMDP)*
2. *Hidden markov model: agent seeks reward from fishing sites without direct observation*

Based on when and where fish are caught, update belief about the fish location, i.e. the posterior of the fish given past observations. Control position to get the most fish while minimizing the cost of switching sides.

a. *Measure when fish are caught, first if the school of fish doesn't move.*
b. *For moving fish, plot their dynamics and your belief about it based on measurements.*
c. *Compute the value for a given control policy.*
d. *Find the optimal policy for controlling your position.*

Let's go fishing!

Partially observable markov decision processes(PDMDP)

Experimental setup

There are two locations for the fish and agent (Left and Right). If agent and fish are on the same side, probability = q_{high} per discrete time step. Otherwise probability = q_{low}. One fish is worth 1 "point".

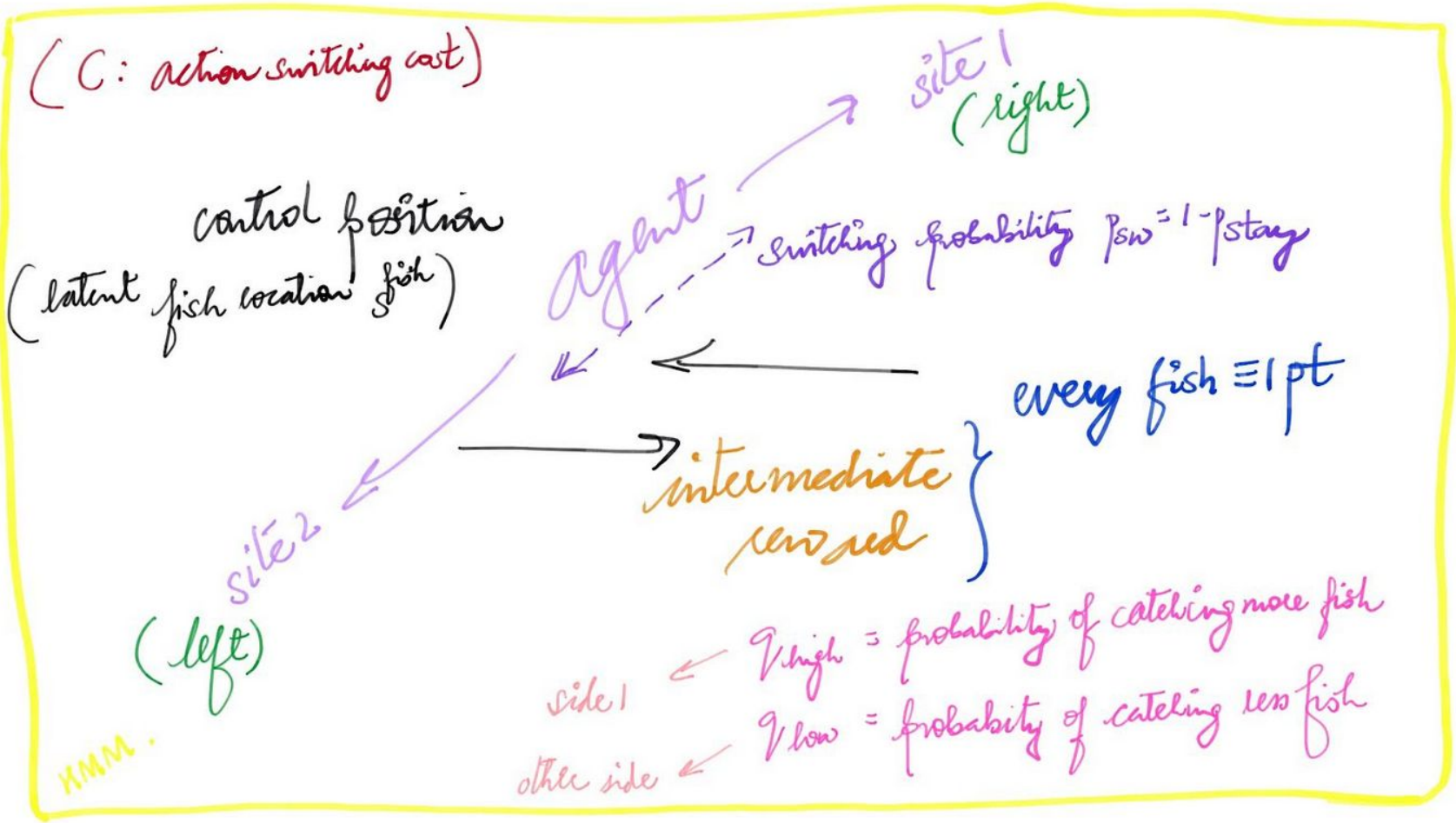
The fish location s_{fish} is latent. Secretly at each time step, the fish may switch sides with a certain probability

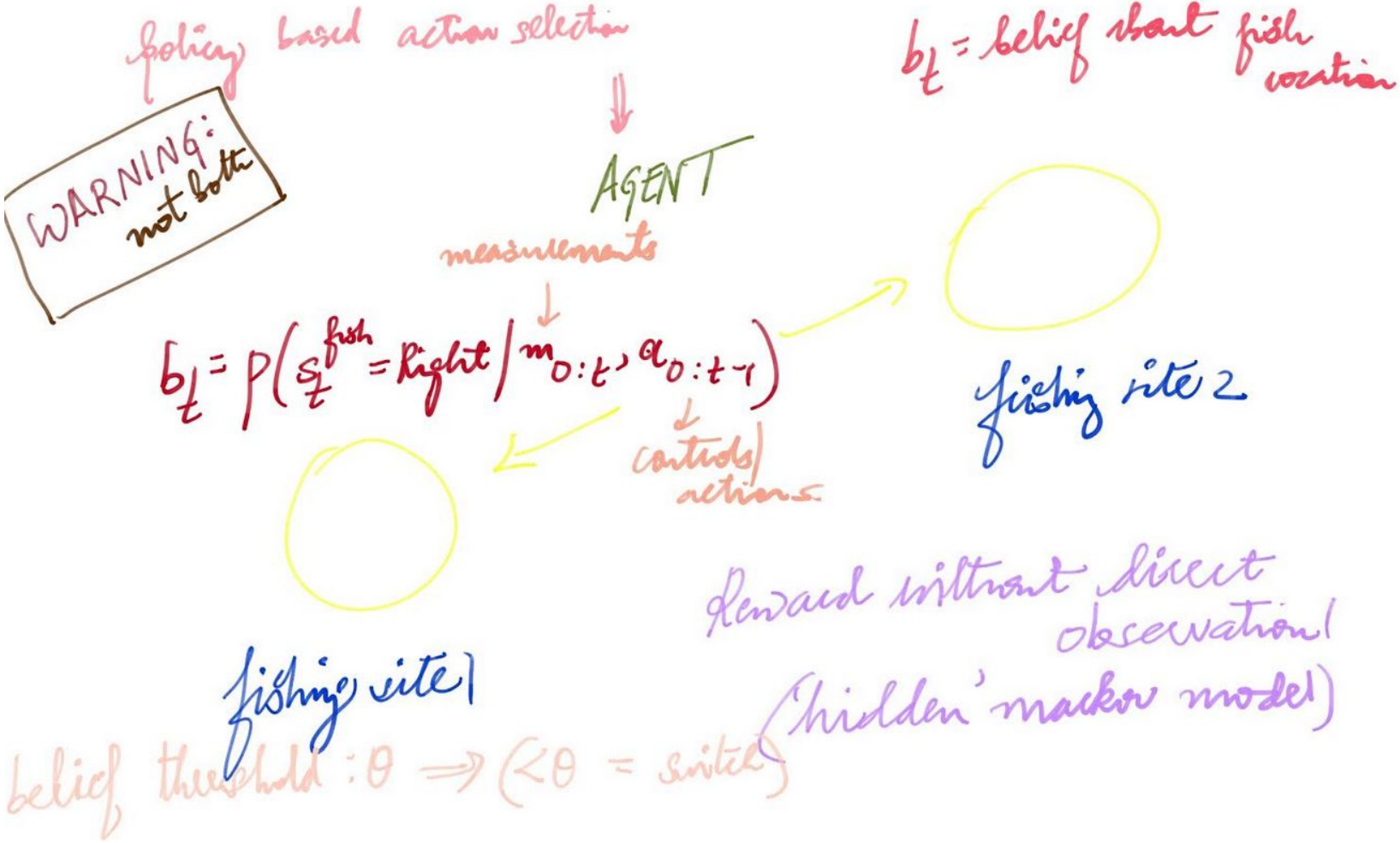
$p_{sw} = 1 - p_{stay}$.

Stay on current side with no cost, or switch to the other side and incur an action cost C (in units of fish).

Select controls or actions by following a **policy**. This defines what to do in any situation. Here the situation is specified by agent's location and belief b_t about the fish location. For optimal control, assume that this belief is the posterior probability over the current fish location, given all the past measurements.

ultimately parameterize the policy by a simple threshold on beliefs. (This happens to be optimal at the right threshold!) When belief about fish's current side falls below a threshold θ, you switch to the other side.


(C: action switching cost)
site 1
(right)
control position
(latent fish location fish)
agent
switching probability $p_{sw} = 1 - p_{stay}$
every fish ≡ 1 pt
intermediate reward
site 2
(left)
side 1
q_{high} = probability of catching more fish
other side
q_{low} = probability of catching less fish
HMM.


policy based action selection
WARNING: not both
AGENT
measurements
$b_t = P(s_t^{fish} = Right \mid m_{0:t}, a_{0:t-1})$
controls/ actions
b_t = belief about fish location
fishing site 2
fishing site 1
belief threshold : $\theta \Rightarrow (< \theta$ = switch)
reward without direct observation
('hidden' markov model)

Belief

Based on intermediate reward (posterior of fish given past observations)

Posterior probability is the probability an event will happen after all evidence or background information has been taken into account.

Control position

To get maximum reward while minimising cost of swtching sides

Selectⁿ of actions/controls: based on a policy!

fishing

probability q high per discrete time step

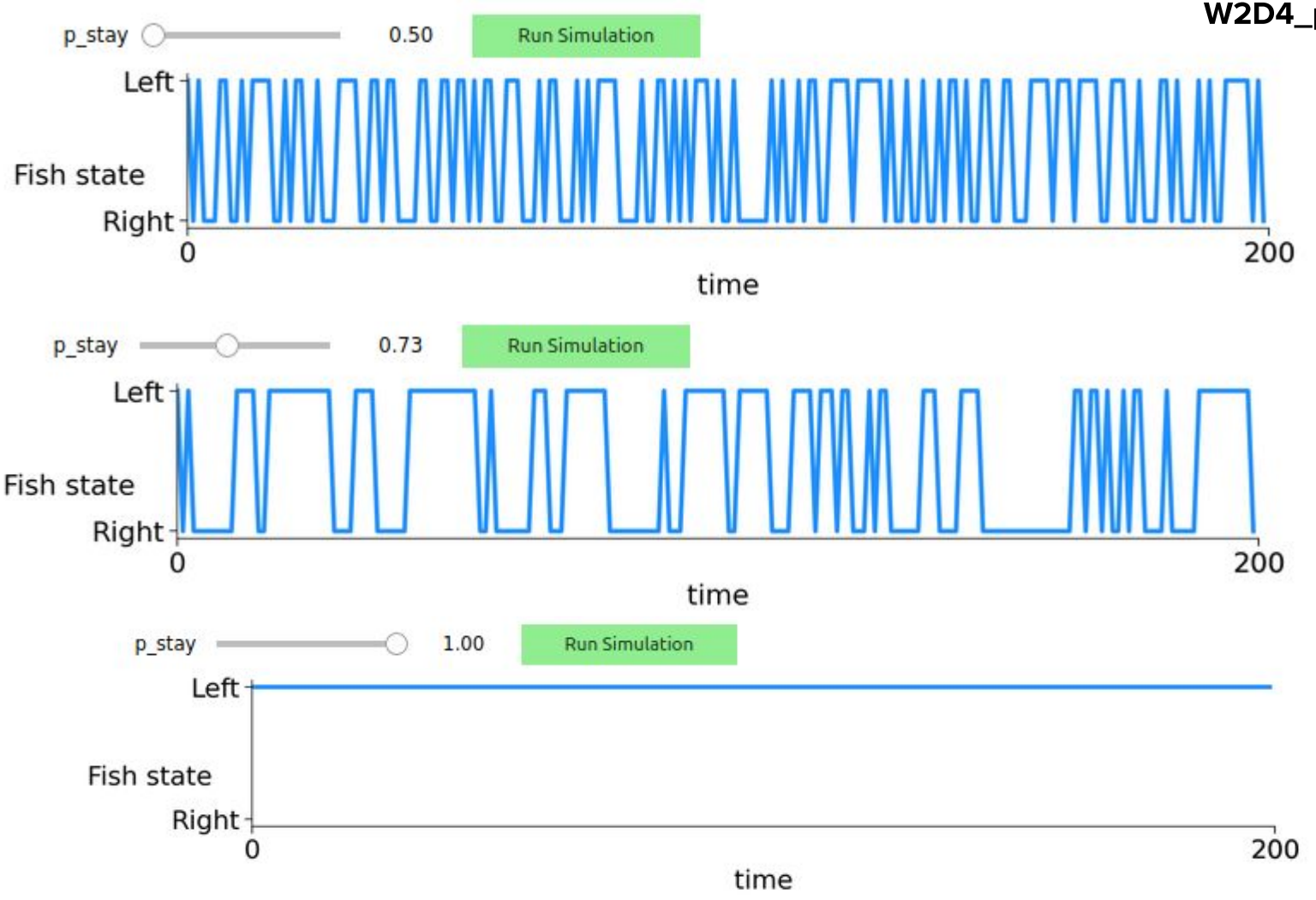
s Cost: 0 cost

probability q low

S cost: c

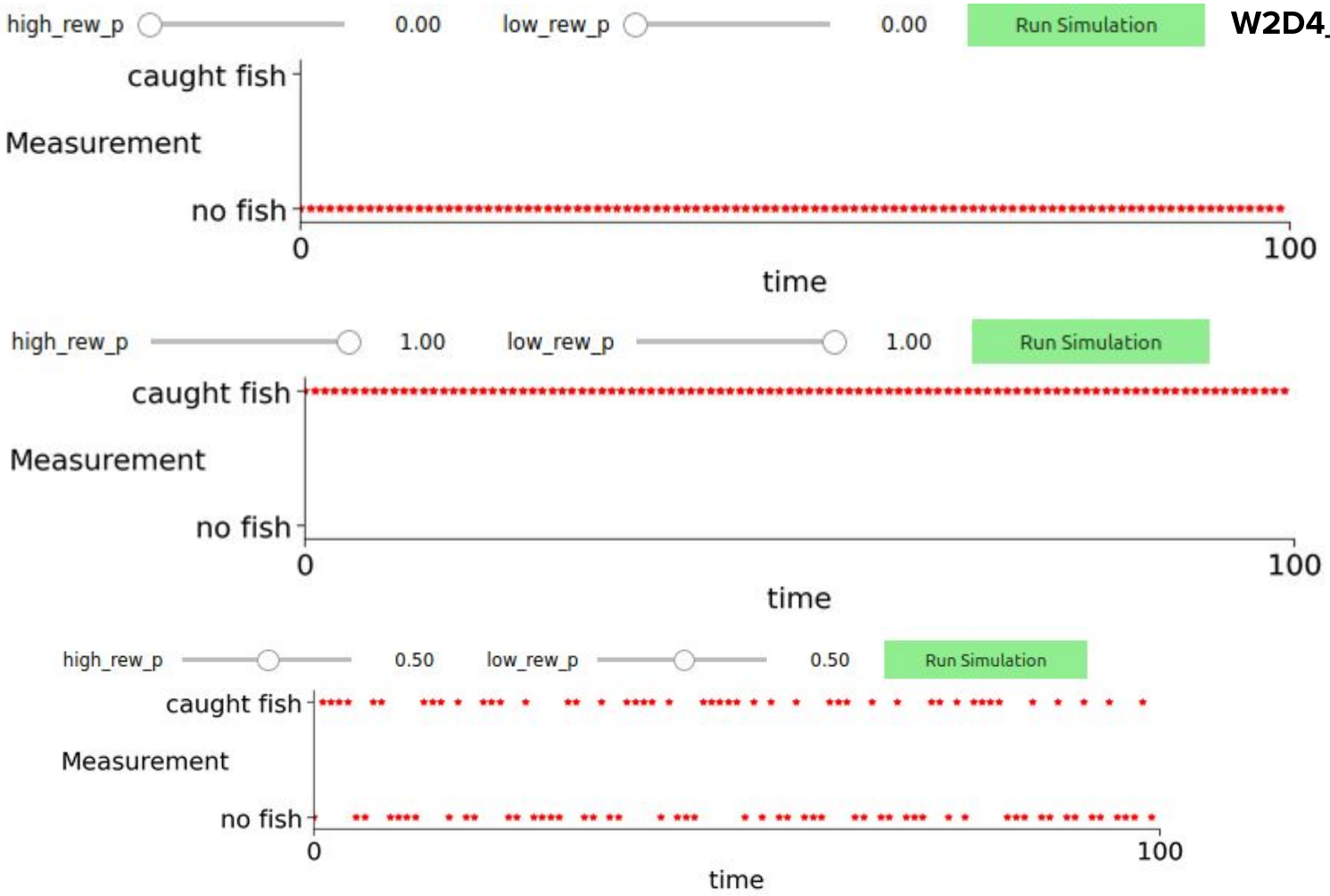
s fish fishing location is latent!

→ 'mean' fish: switch sides

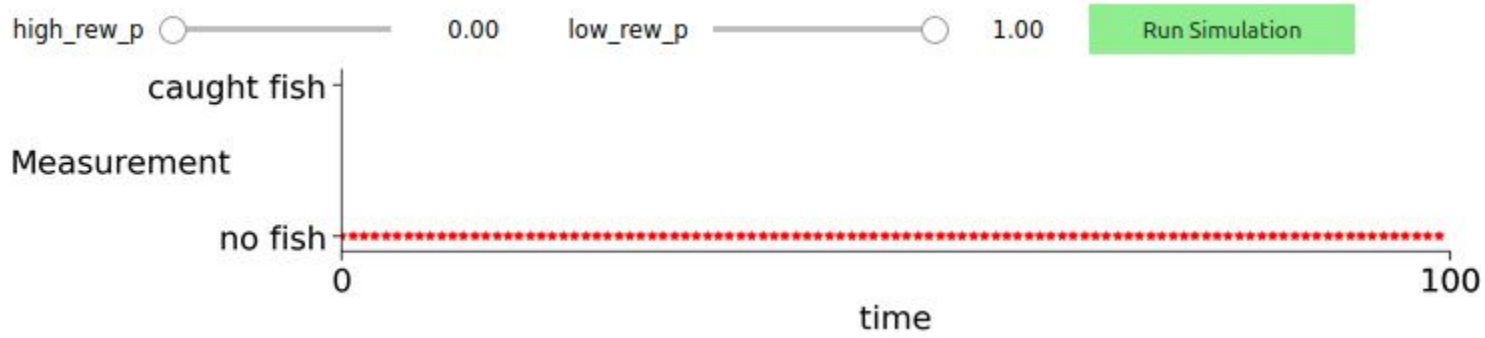

p_stay
0.50
Run Simulation
Left
Fish state
Right
0
200
time
p_stay
0.73
Run Simulation
Left
Fish state
Right
0
200
time
p_stay
1.00
Run Simulation
Left
Fish state
Right
0
200
time

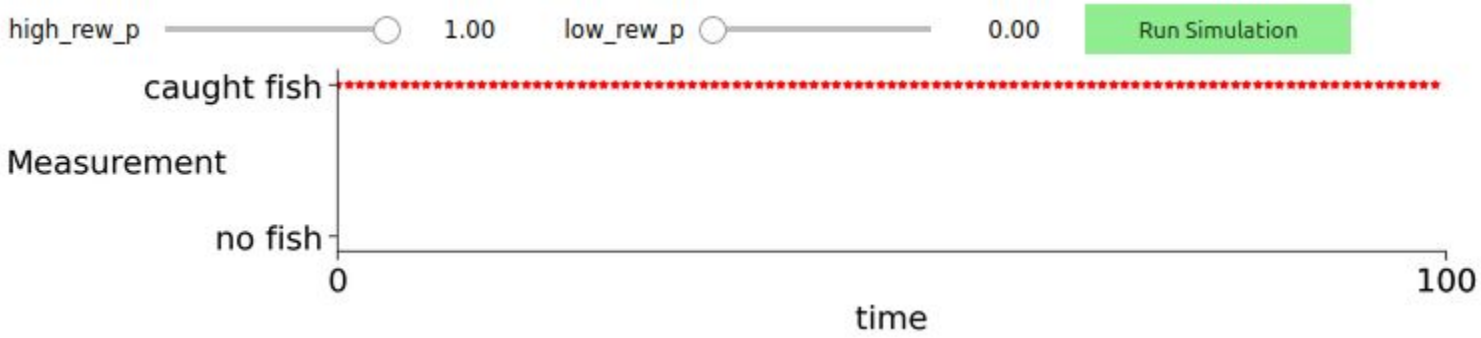
Observations

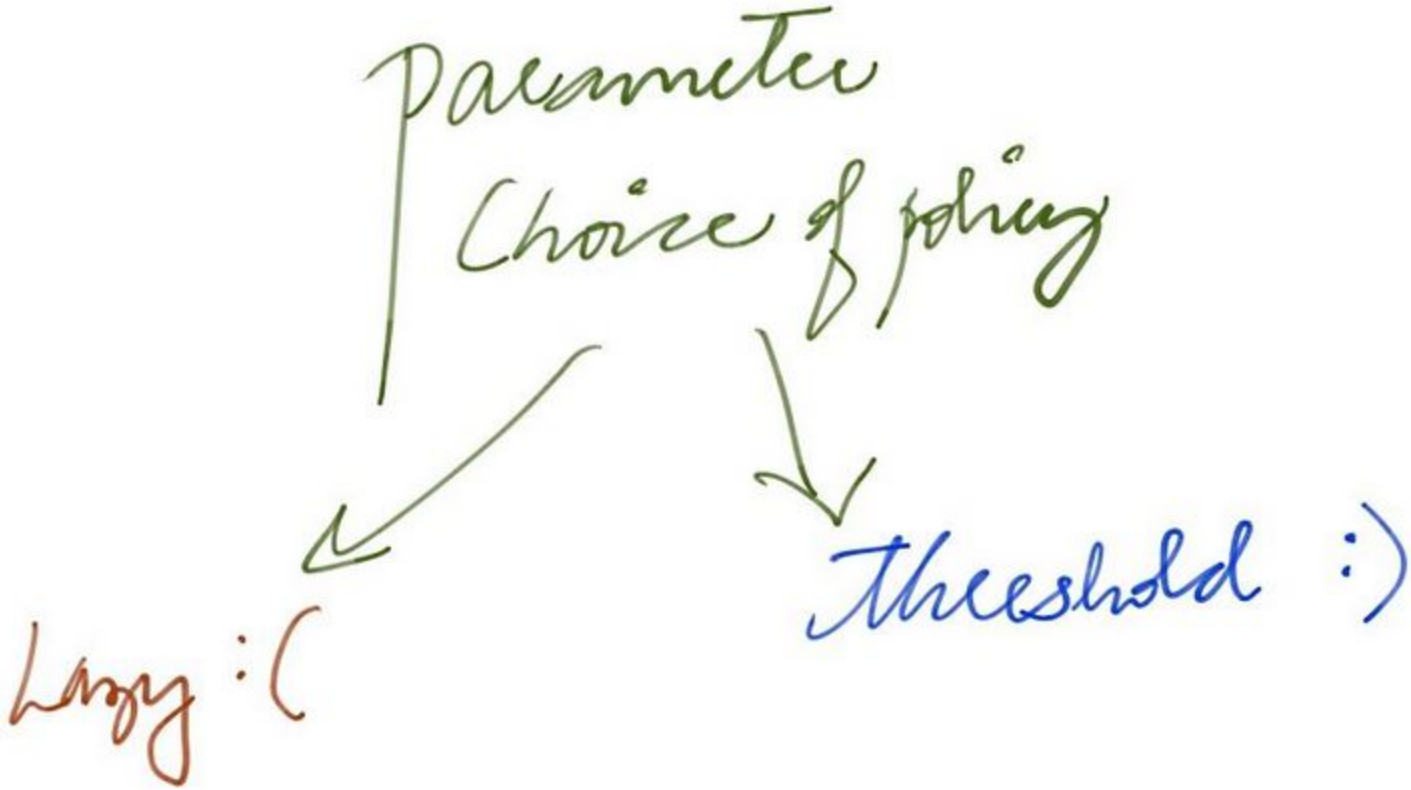
Set pstay =1 so that the state of the two sites are fixed, and we can directly see the chances of catching fish on each side. The variable fish_initial indicates the initial side of the fish, and loc_initial indicates agent's initial location. They each take value -1 for left and 1 for right.


high_rew_p
0.00
low_rew_p
0.00
Run Simulation
caught fish
Measurement
no fish
0
100
time
high_rew_p
1.00
low_rew_p
1.00
Run Simulation
caught fish
Measurement
no fish
0
100
time
high_rew_p
0.50
low_rew_p
0.50
Run Simulation
caught fish
Measurement
no fish
0
100
time

W2D4_pod 031


high_rew_p
0.00
low_rew_p
1.00
Run Simulation
caught fish
Measurement
no fish
0
100
time

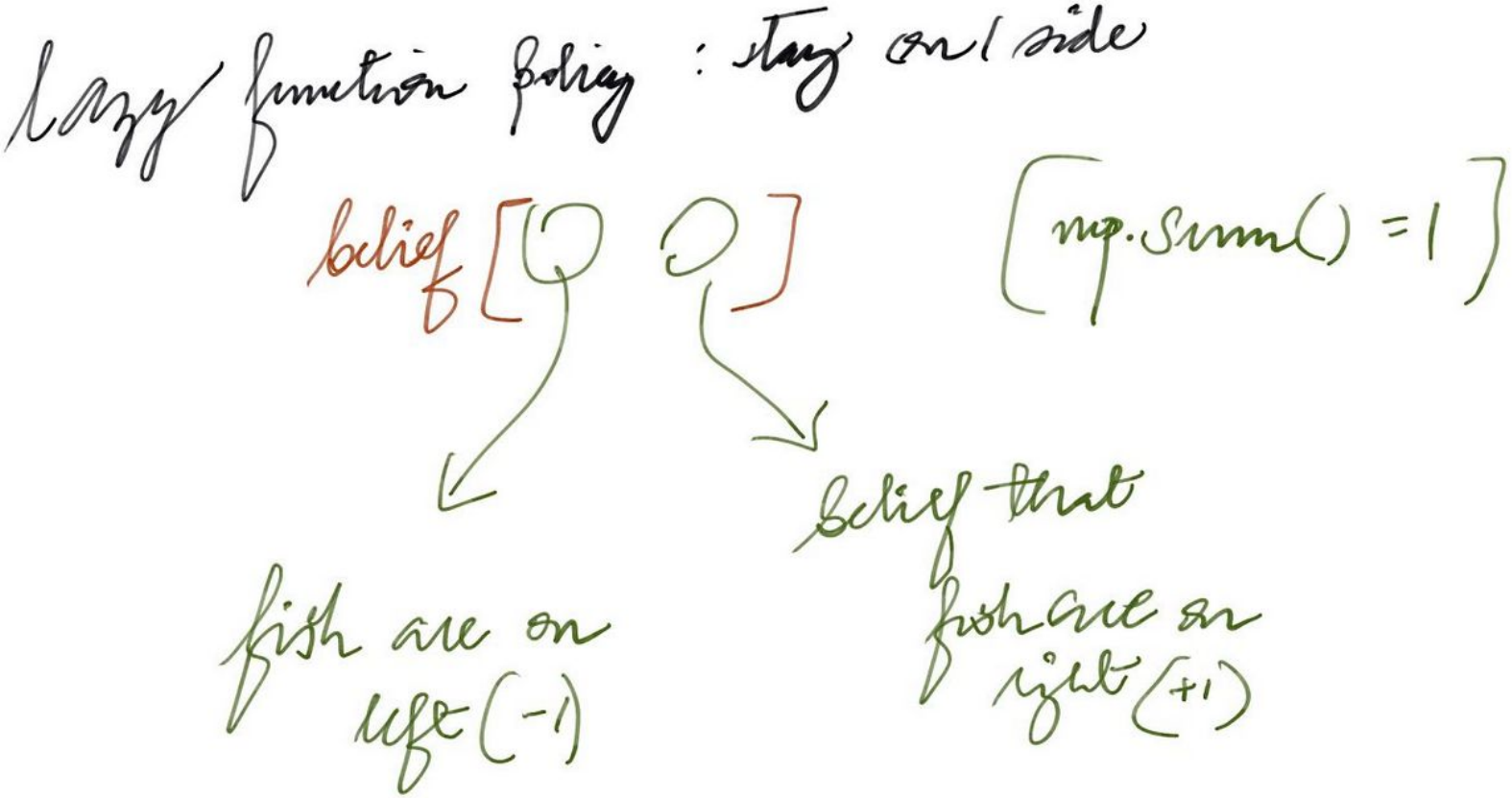

high_rew_p
1.00
low_rew_p
0.00
Run Simulation
caught fish
Measurement
no fish
0
100
time

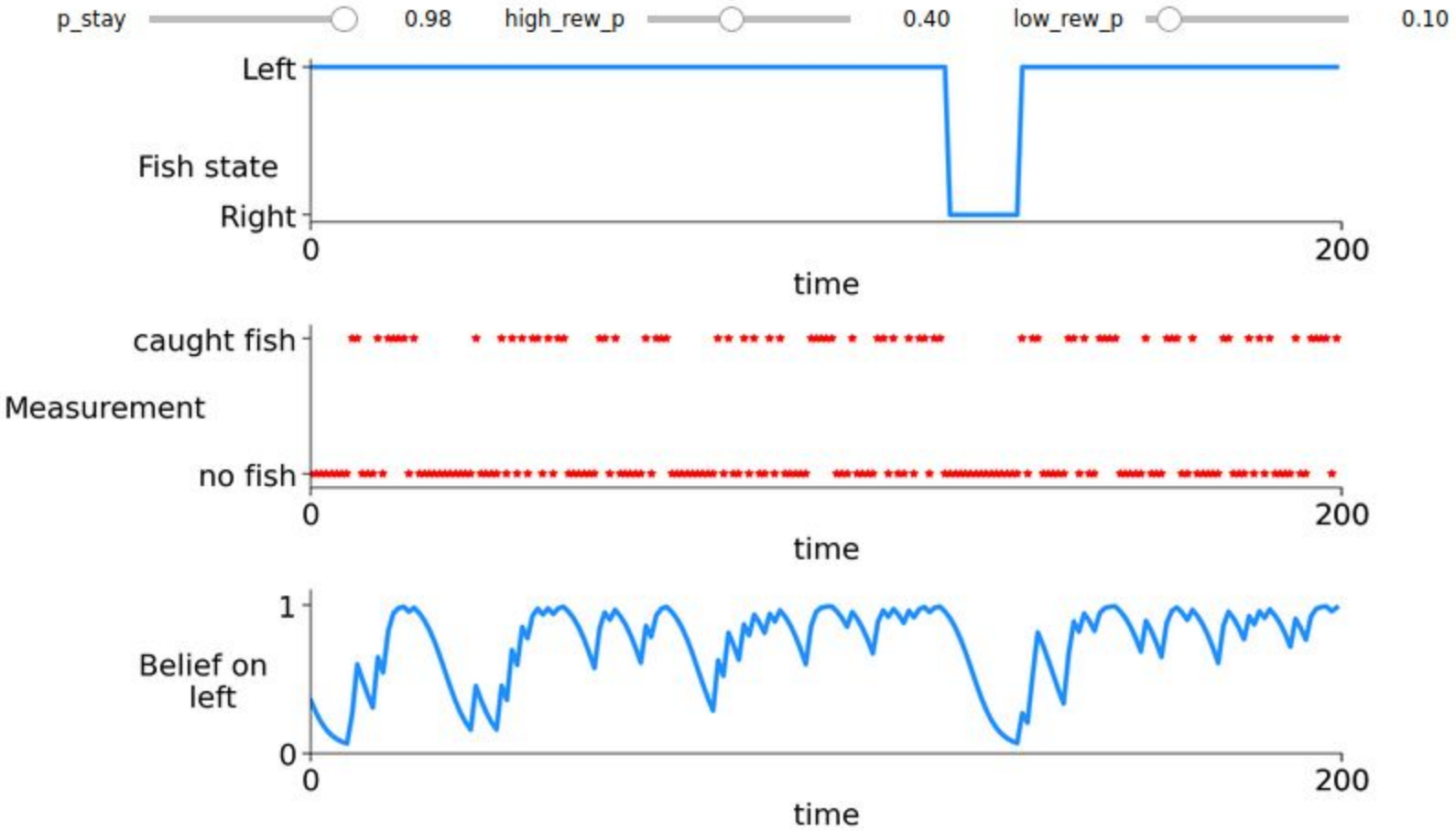

parameter
Choice of policy
Lazy :(
threshold :)

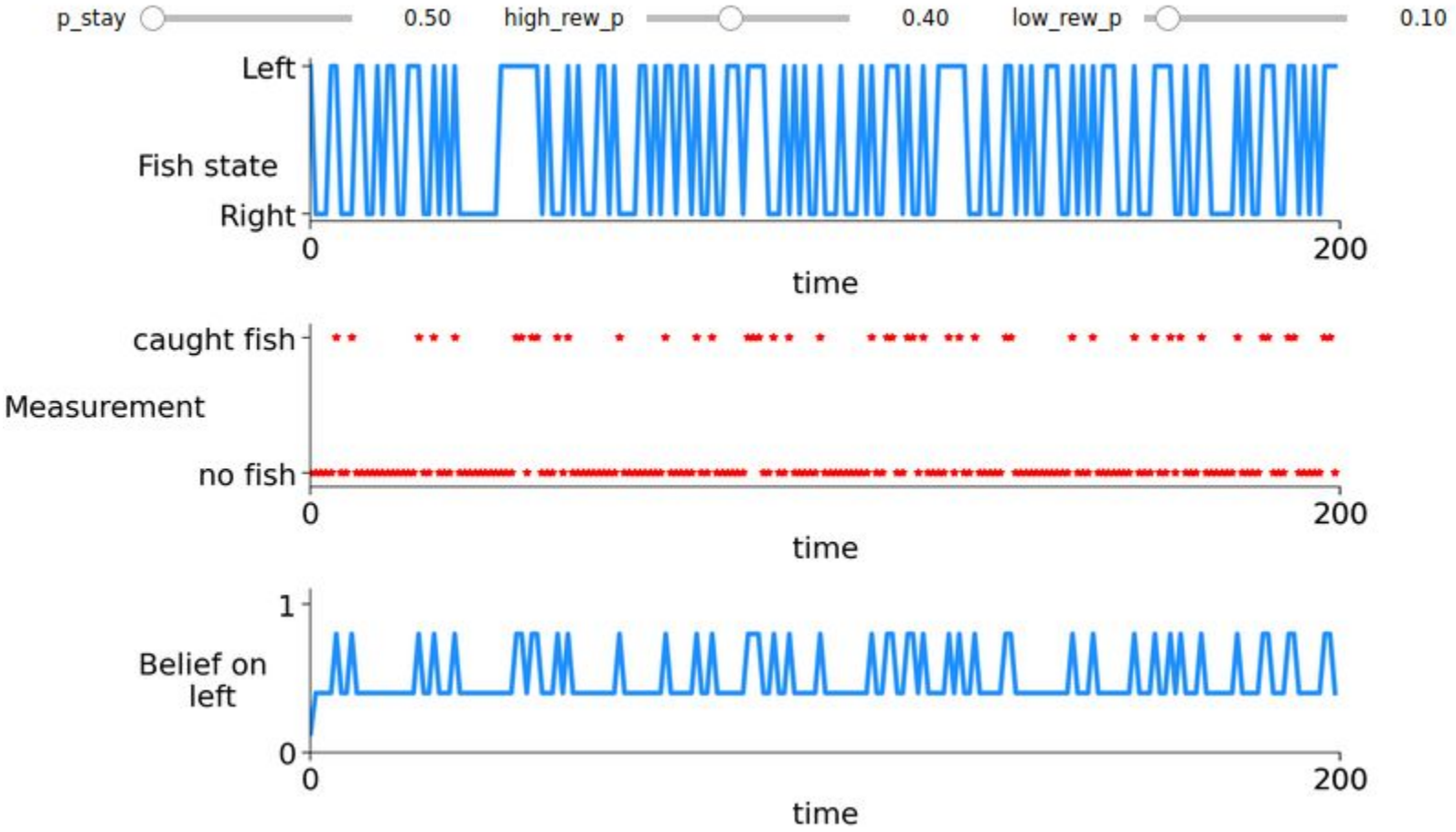
Lazy

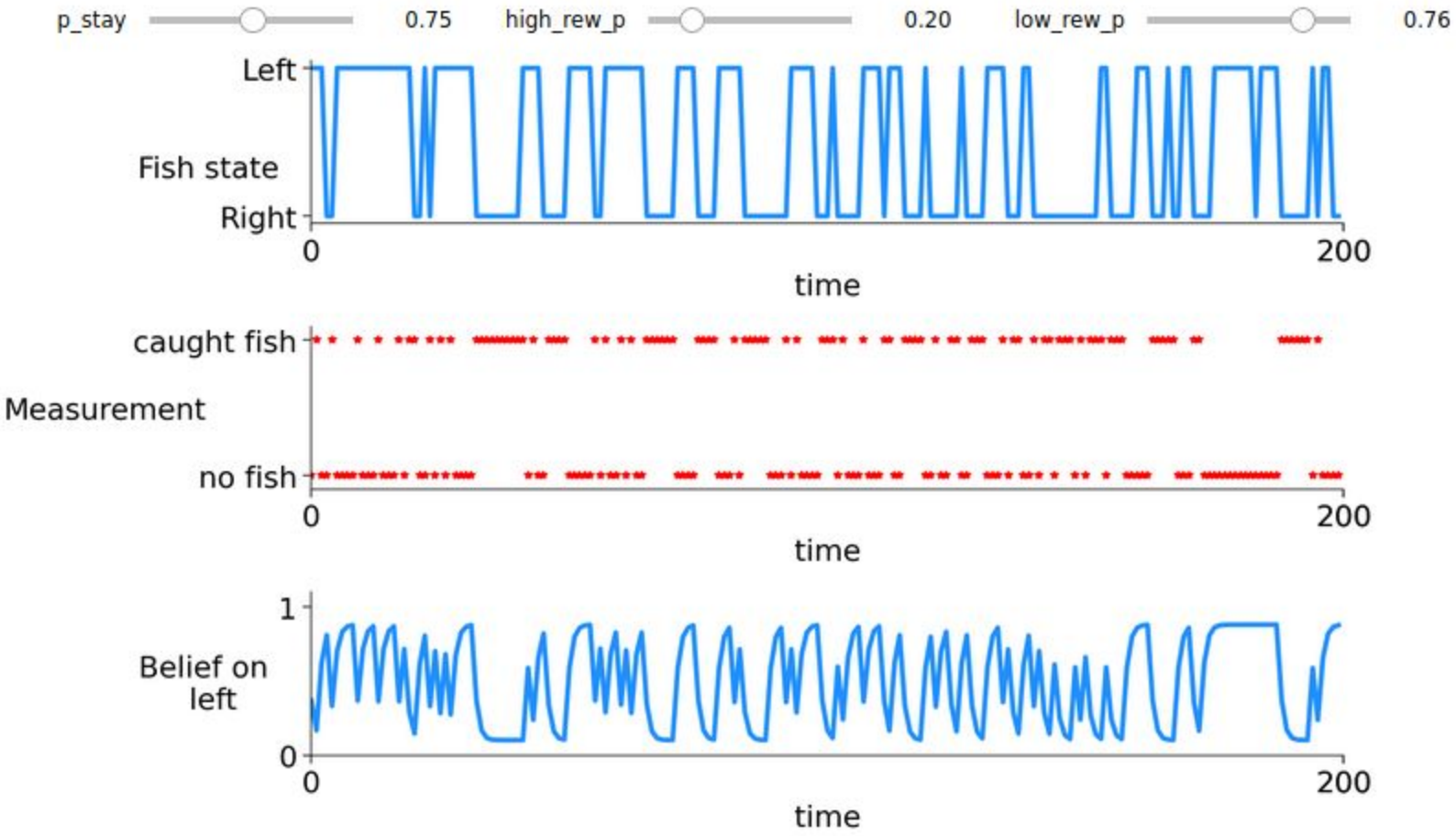
Agent always staying at one side following a lazy policy function.

Extend to generate the real dynamics, including beliefs and a moving agent to see how beliefs change over time, and how often different beliefs happen. For convenience, belief at time t is actually a 2-dimensional vector. The first element is the belief that the fish are on the left, and the second element is the belief the fish are on the right. At every time, these elements sum to 1.


Lazy function policy : stay on 1 side
belief [ ]
[np.sum() = 1]
fish are on left (-1)
belief that fish are on right (+1)


p_stay
0.98
high_rew_p
0.40
low_rew_p
0.10
Left
Fish state
Right
0
200
time
caught fish
Measurement
no fish
0
200
time
1
Belief on left
0
0
200
time


p_stay
0.50
high_rew_p
0.40
low_rew_p
0.10
Left
Fish state
Right
0
200
time
caught fish
Measurement
no fish
0
200
time
1
Belief on left
0
0
200
time

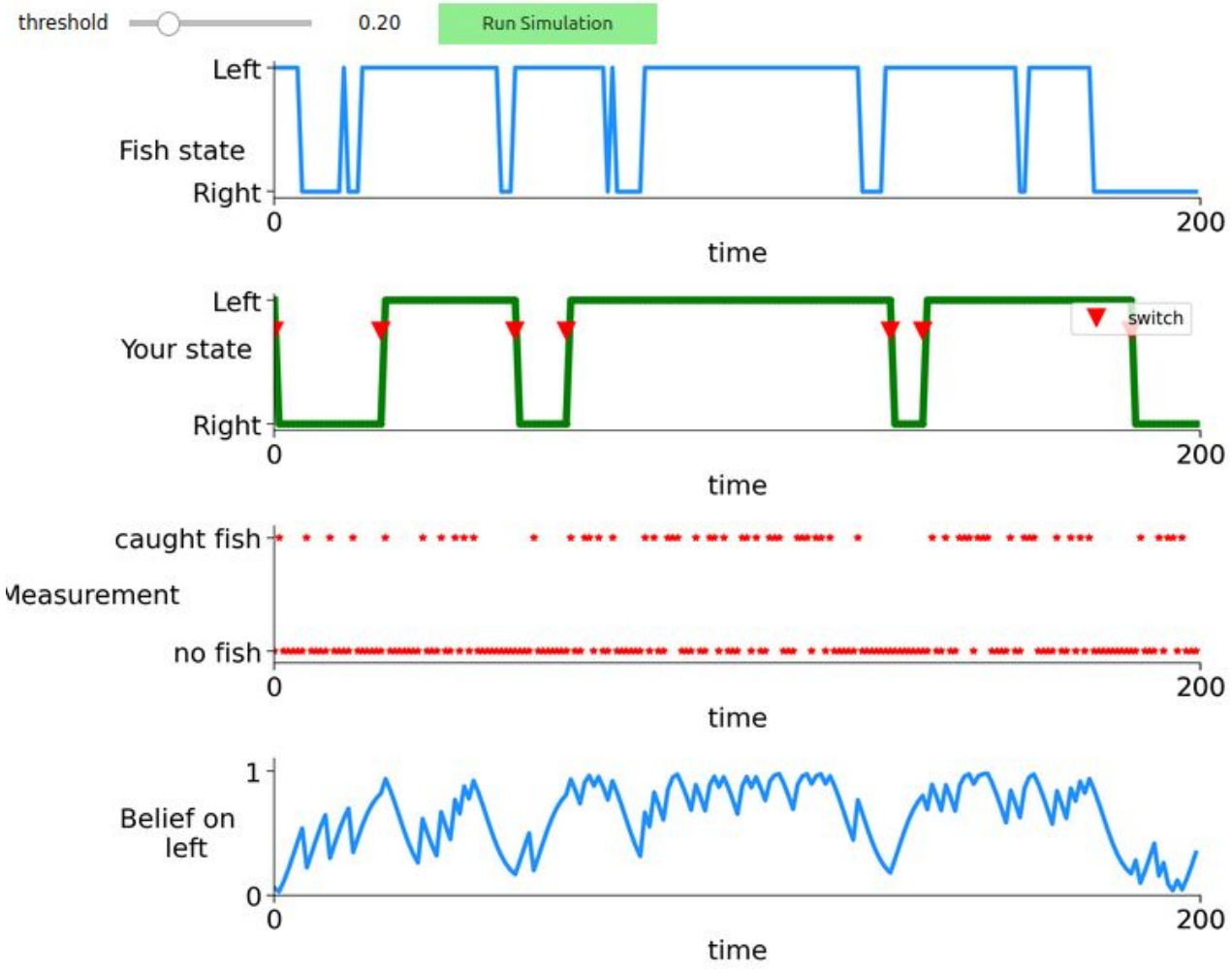

p_stay
0.75
high_rew_p
0.20
low_rew_p
0.76
Left
Fish state
Right
0
200
time
caught fish
Measurement
no fish
0
200
time
1
Belief on left
0
0
200
time

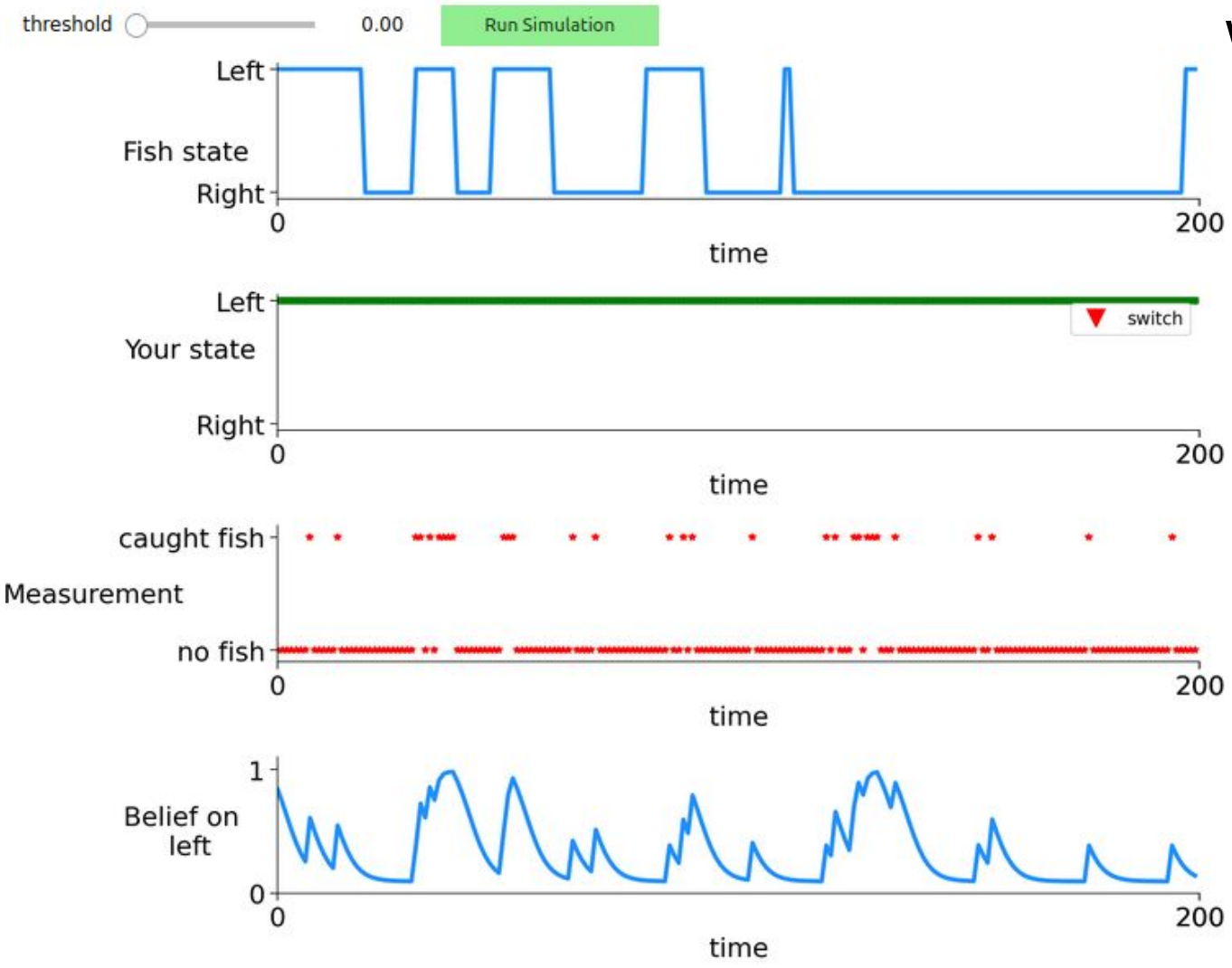
Threshold

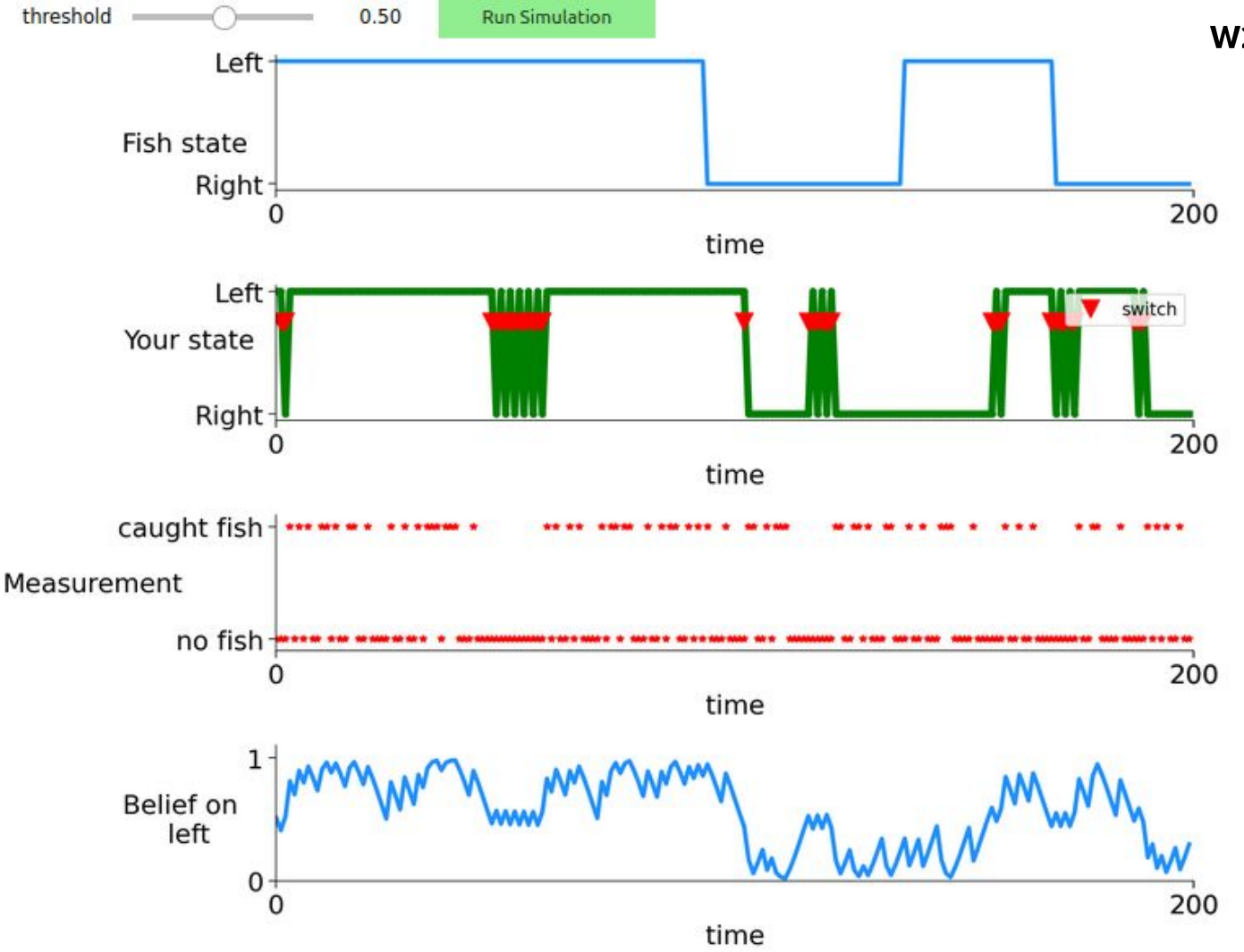
Lazy + task dynamics

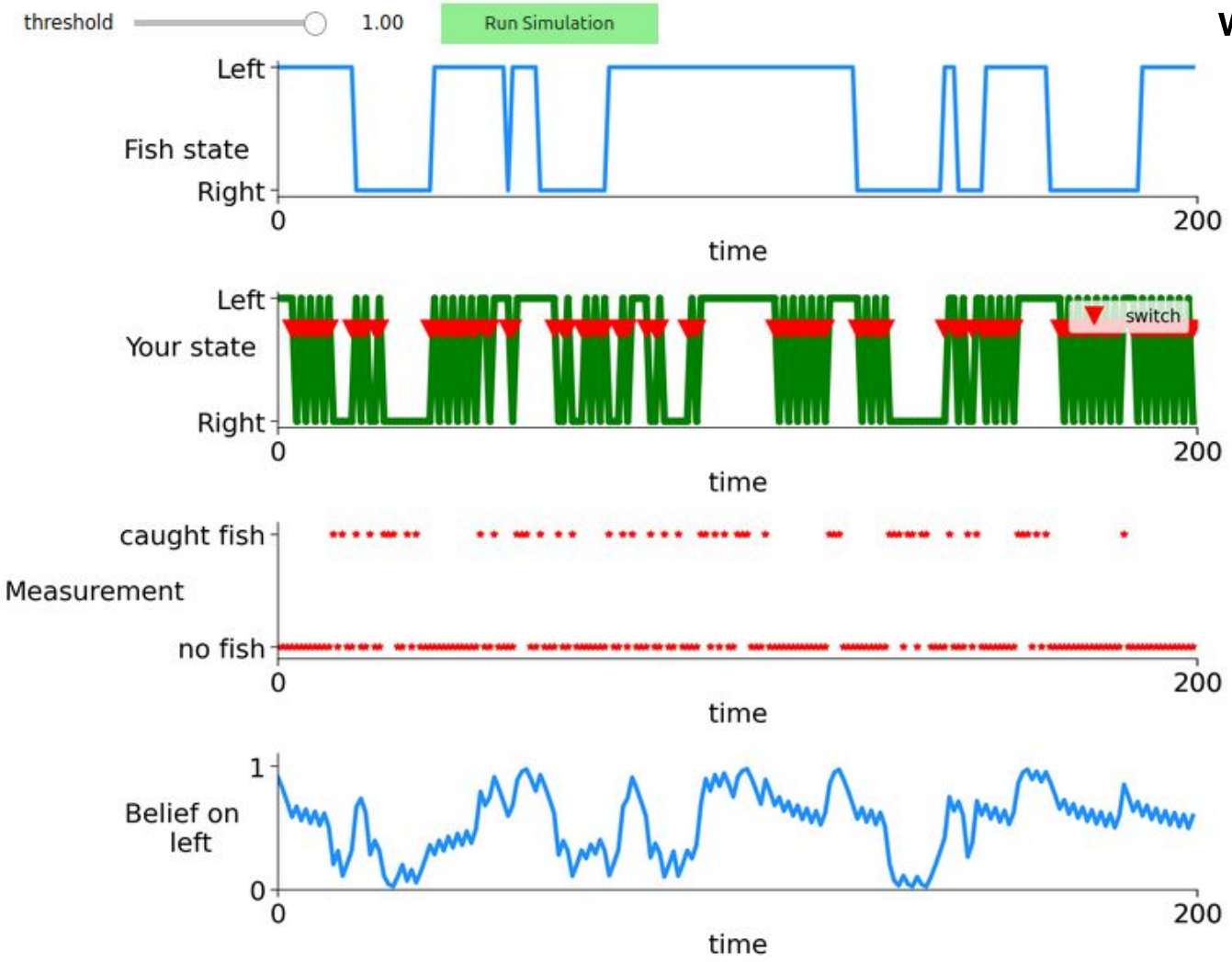
- The policy takes three inputs: belief about the fish state, location ("Left" or "Right"), and a belief threshold: when belief that agent is on the same side as the fish drops below this threshold, you choose to switch; otherwise you stay.

- You should return an action for each time t, which takes the value of "stay" or "switch".


threshold
0.20
Run Simulation
Left
Fish state
Right
0
200
time
Left
Your state
Right
switch
0
200
time
caught fish
Measurement
no fish
0
200
time
1
Belief on left
0
0
200
time


threshold
0.00
Run Simulation
Left
Fish state
Right
0
200
time
Left
Your state
Right
switch
0
200
time
caught fish
Measurement
no fish
0
200
time
1
Belief on left
0
0
200
time


threshold
0.50
Run Simulation
Left
Fish state
Right
0
200
time
Left
Your state
Right
switch
0
200
time
caught fish
Measurement
no fish
0
200
time
1
Belief on left
0
0
200
time


threshold
1.00
Run Simulation
Left
Fish state
Right
0
200
time
Left
Your state
Right
switch
0
200
time
caught fish
Measurement
no fish
0
200
time
1
Belief on left
0
0
200
time

We will use this value to compare different policies, and maximize the amount of fish we catch while minimizing our effort.

how good is your threshold?

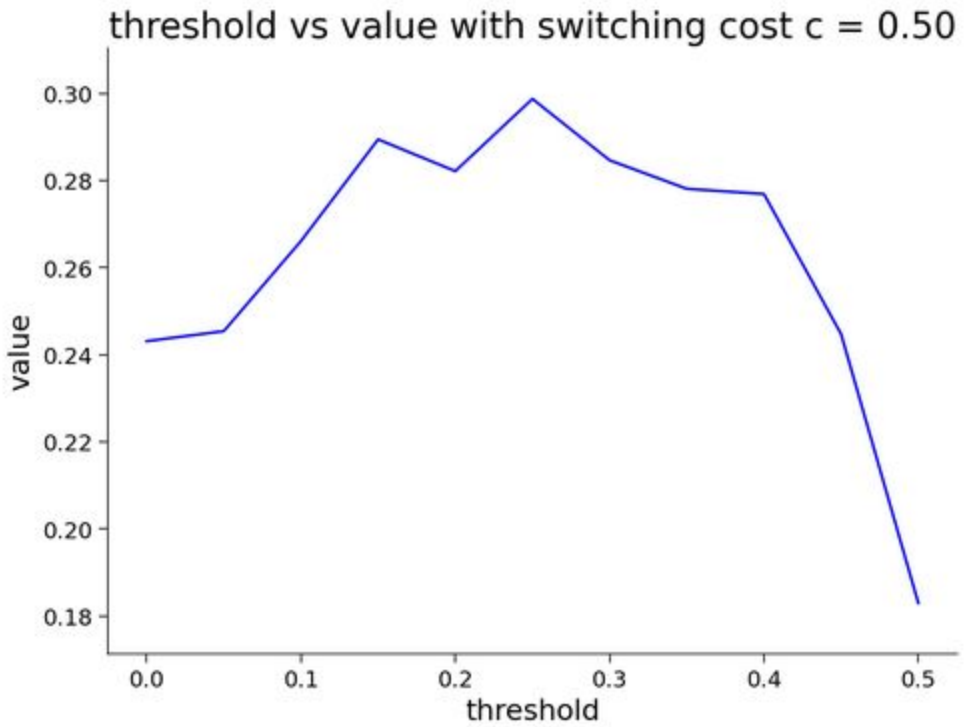
Value function

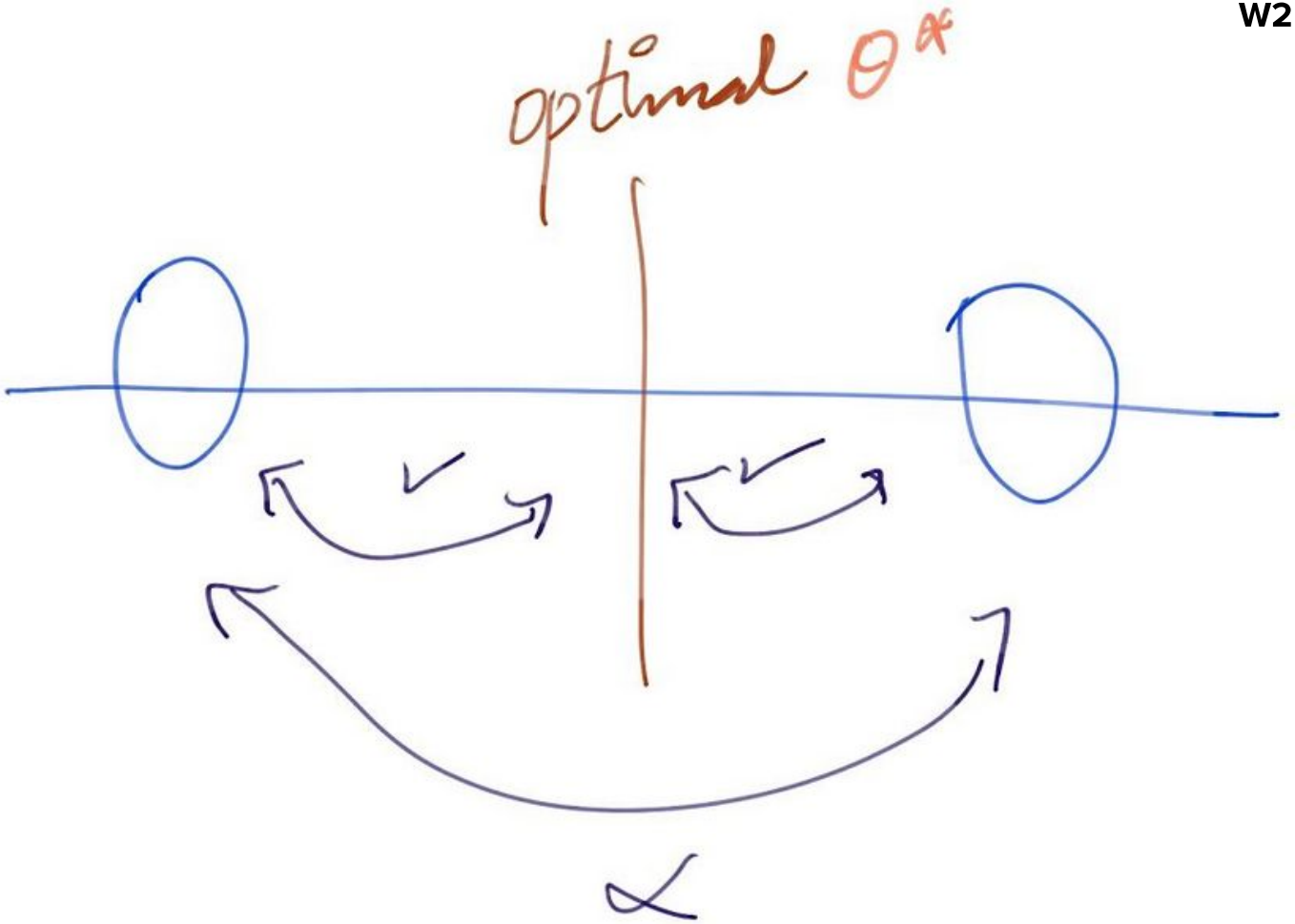
$$V(\theta) = \frac{1}{T}\left(\sum_t u_s(s_t) + u_a(a_t)\right)$$

$u_s(s_t)$: instantaneous utility / reward

$u_a(a_t)$: utility / negative cost for chosen action

action cost $= \begin{cases} 0 & \text{if stay} \\ \text{cost\_sw} & \text{if switch} \end{cases}$


threshold vs value with switching cost c = 0.50
value
0.30
0.28
0.26
0.24
0.22
0.20
0.18
0.0
0.1
0.2
0.3
0.4
0.5
threshold


optimal θ°
α

belief about fish location: b_t

optimal control = belief (posterior probability)

parameterise policy by single threshold on beliefs (θ)

right threshold? optimal!

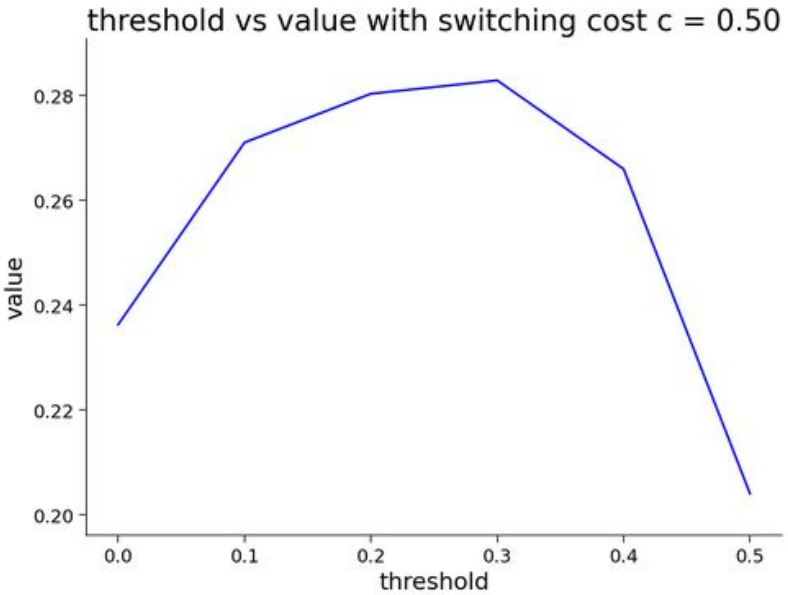
Tutorial #1 Bonus Explanations

Objective

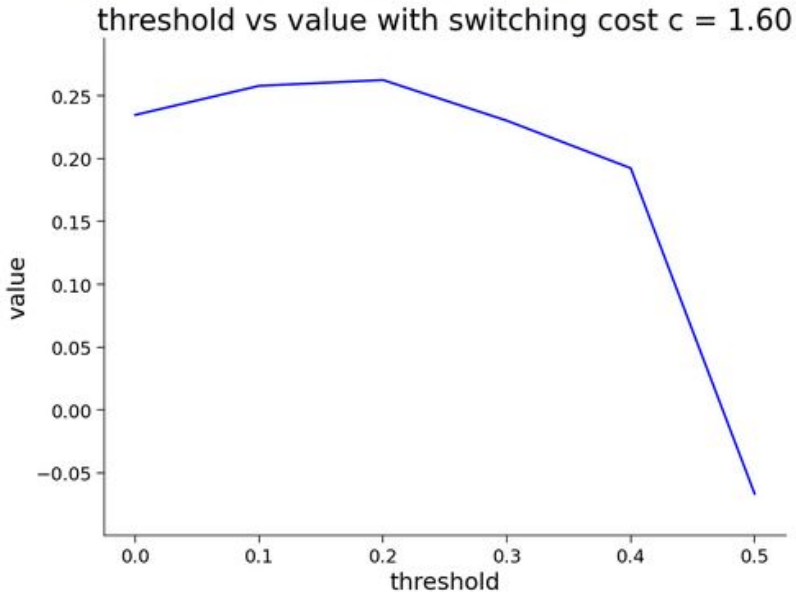
After plotting value versus threshold, observe how the optimal threshold moves with

- switching cost (cost_sw)
- fish dynamics (p_switch)
- probability of catching fish on each side, low_rew_p and high_rew_p

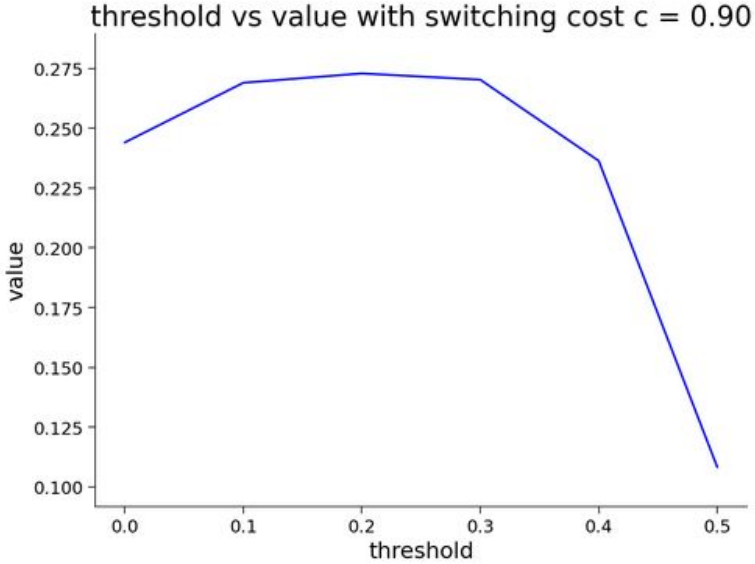
cost_sw 0.50 p_stay 0.95 high_rew_p 0.40 low_rew_p 0.10



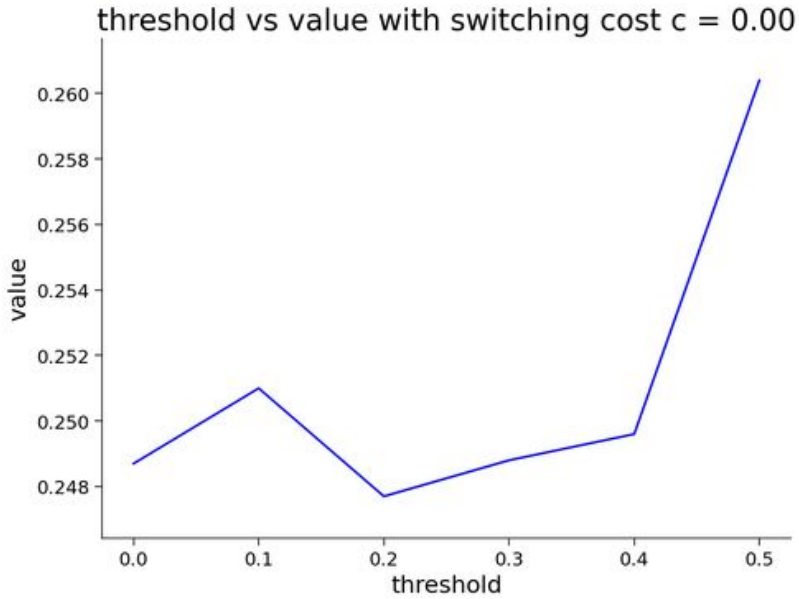
cost_sw 1.60 p_stay 0.95 high_rew_p 0.40 low_rew_p 0.10



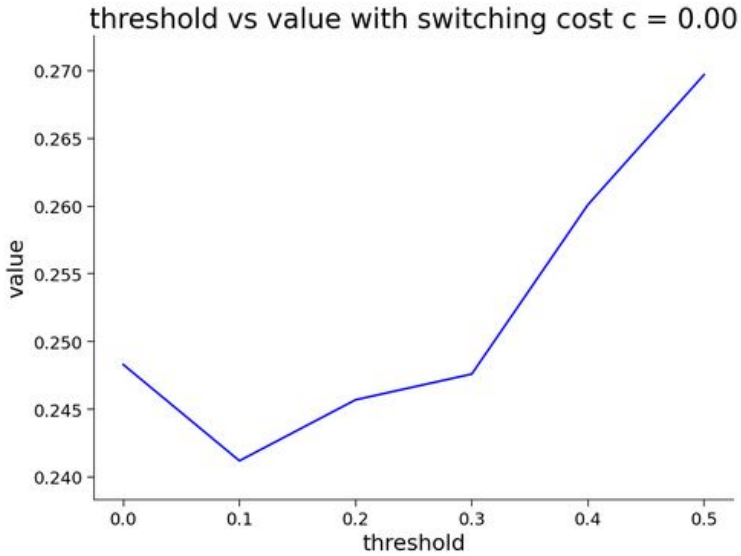
cost_sw 0.90 p_stay 0.95 high_rew_p 0.40 low_rew_p 0.10



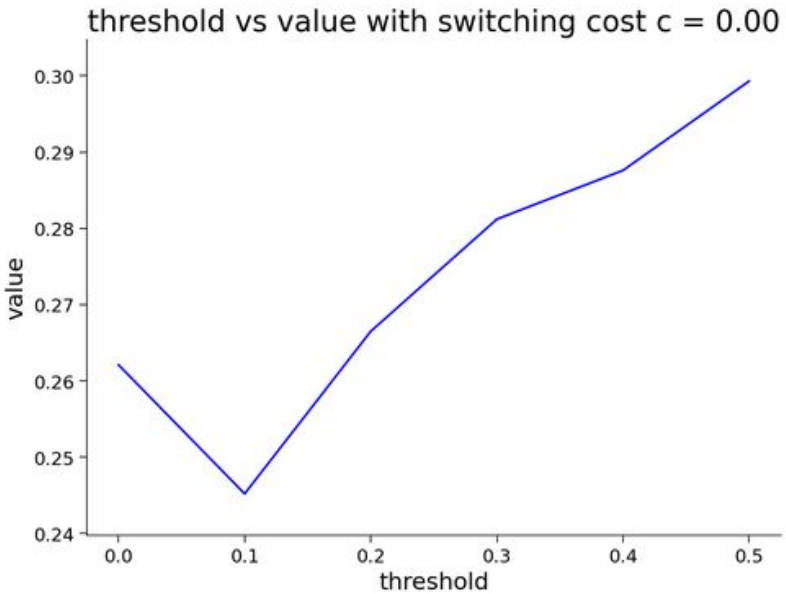
cost_sw 0.00 p_stay 0.59 high_rew_p 0.40 low_rew_p 0.10



cost_sw 0.00 p_stay 0.89 high_rew_p 0.40 low_rew_p 0.10



cost_sw 0.00 p_stay 0.89 high_rew_p 0.40 low_rew_p 0.10



Observations

- High switching cost means that you should be more certain that the other side is better before committing to change sides. This means that beliefs must fall below a threshold before acting. Conversely, a lower switching cost allows you more flexibility to switch at less stringent thresholds. In the limit of zero switching cost, you should always switch whenever you think the other side is better, even if it's just 51%, and even if you switch every time step.
- Faster fish dynamics (lower p_stay) also promotes faster switching, because you cannot plan as far into the future. In that case you must base your decisions on more immediate evidence, but since you still pay the same switching cost that cost is a higher fraction of your predictable rewards. And thus you should be more conservative, and switch only when you are more confident.
- When high_rew_p and/or low_rew_p decreases, your predictions become less reliable, again encouraging you to require more confidence before committing to a switch.

Tutorial #2
Explanations

objective

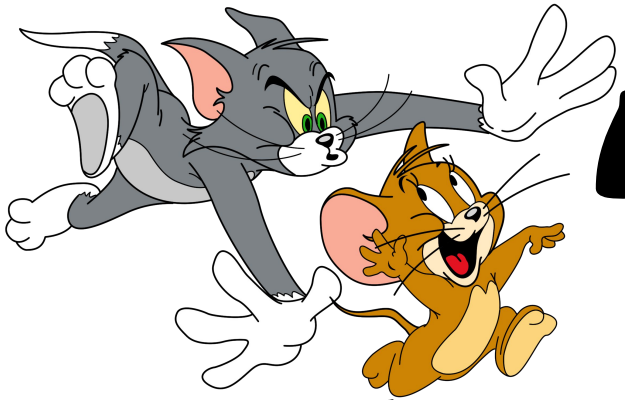
implement a continuous control task: design control inputs for a linear dynamical system to reach a target state. The state here is continuous-valued, i.e. takes on any real number from $-\infty$ to ∞.

Designing Controllers

Full observability of state(linear quadratic regulator LQR)

Partial observability of state(linear quadratic gaussian LQG)

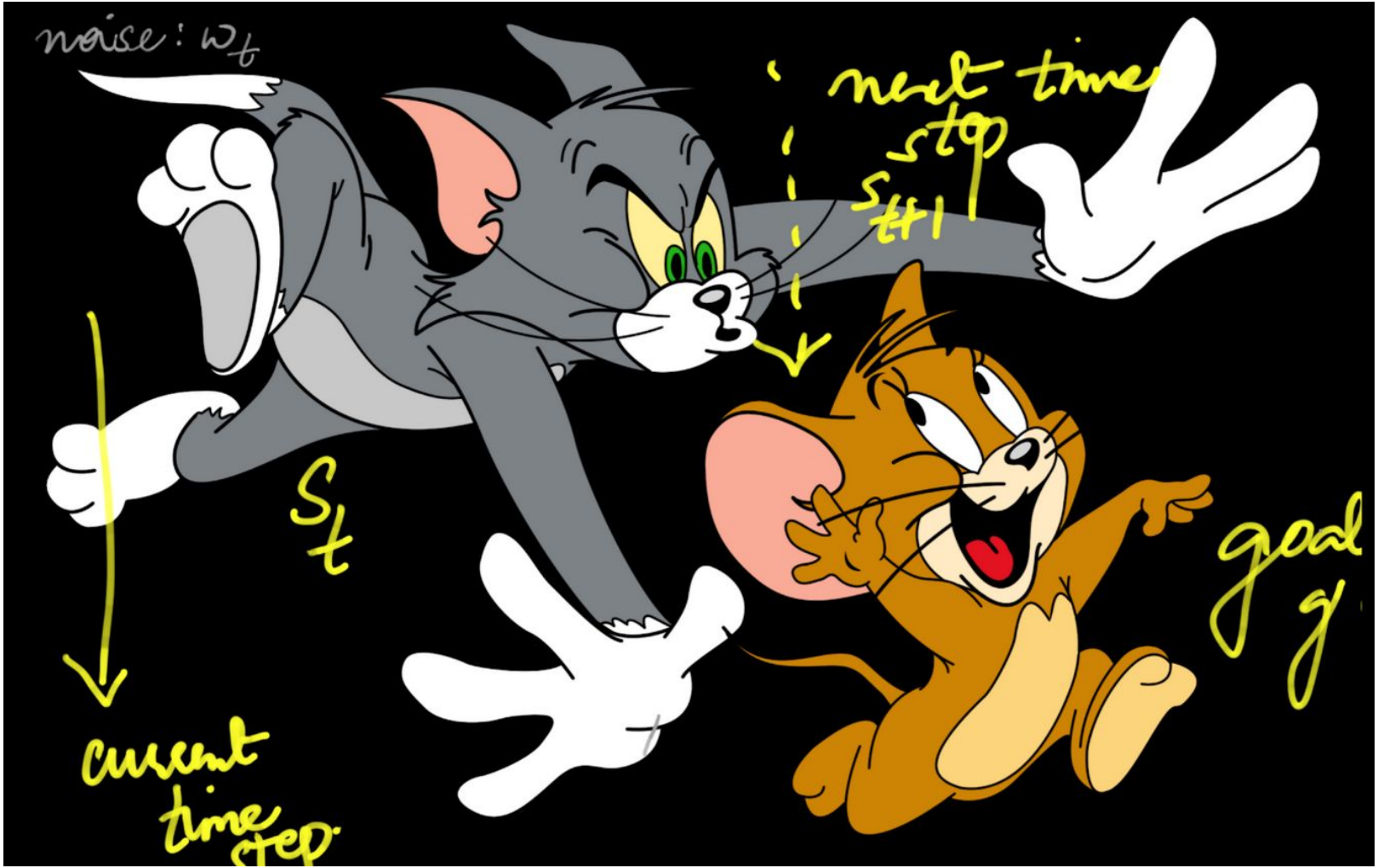
Exploring a Linear Dynamical System (LDS) with Open-Loop and Closed-Loop Control

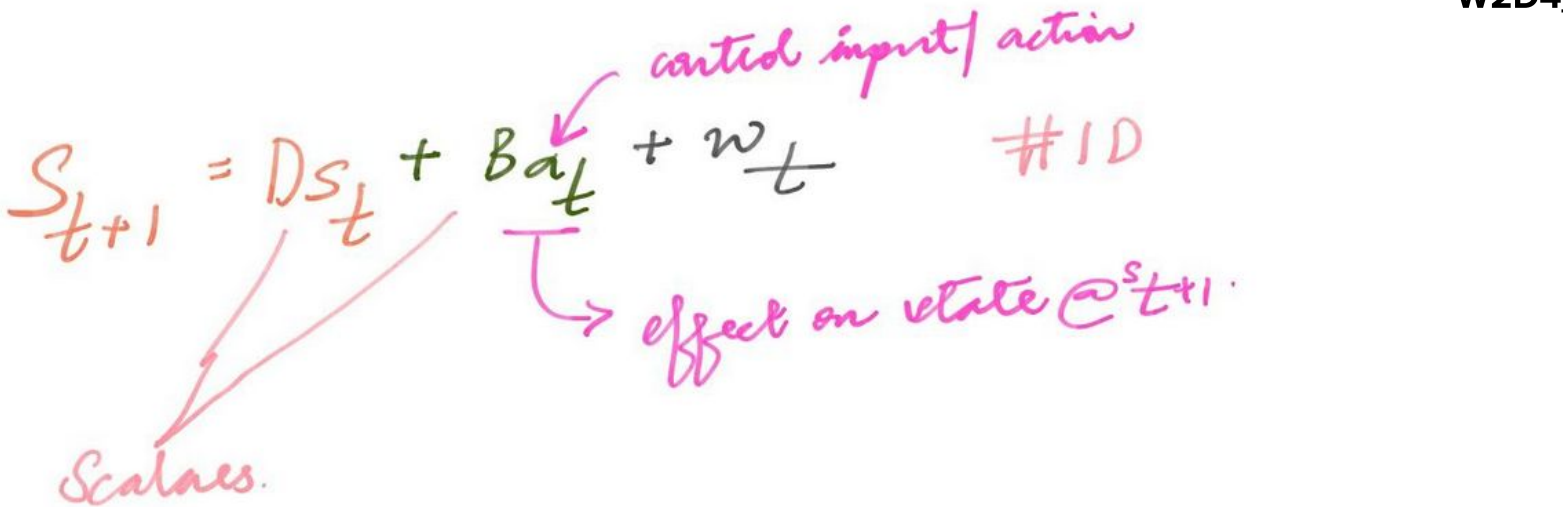


Experimental setup

Cat is trying to catch a mouse in space. The location of the mouse is the goal state g, here a static goal. Later on, we will make the goal time varying, i.e. $g(t)$. The cat's location is the state of the system st. The state has its internal dynamics: think of the cat drifting slowly in space. These dynamics are such that the state at the next time step $st+1$ are a linear function of the current state st. There is some environmental noise (think: meteorites) affecting the state, here modeled as gaussian noise wt.

The control input or action at is the action of the jet pack, which has an effect Bat on the state at the next time step $st+1$. In this tutorial, we will be designing the action at to reach the goal g, with known state dynamics.


noise : w_t
next time step
s_{t+1}
s_t
goal
g
current time step



$s_0 = s_{init}$

t : 1 to T (time horizon)
s_t : state @ t
a_t : action @ t
w_t : gaussian noise @ t

S_{t+1} depends on - current state S_t

& effect/act" on state

& gaussian noise (env. factors).

Stable system : finite o/p for any condition

Open loop control : $a_t \neq f(s_t)$

Closed loop control : $a_t = f(s_t)$ # linear function

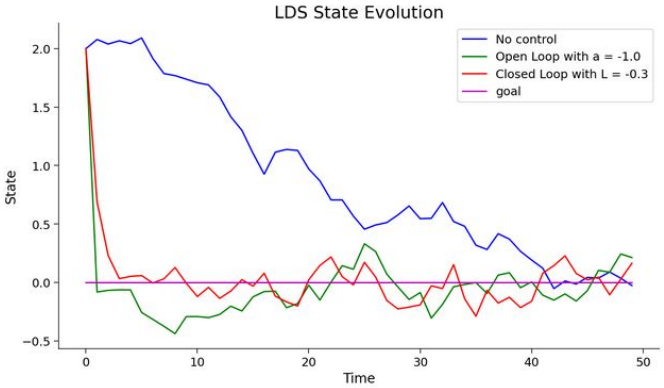
$a_t = L_t s_t$

↙

control gain

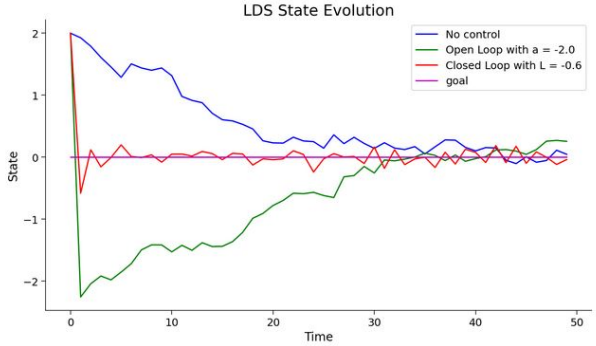
D 0.95 L -0.30 a -1.00 B 2.00 noise_var 0.10

ini_state 2.00 Run Simulation



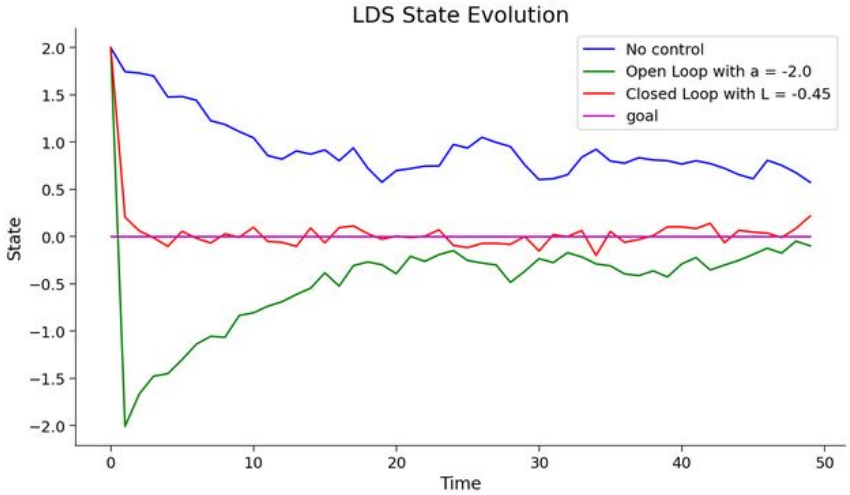
D 0.95 L -0.60 a -2.00 B 2.00 noise_var 0.10

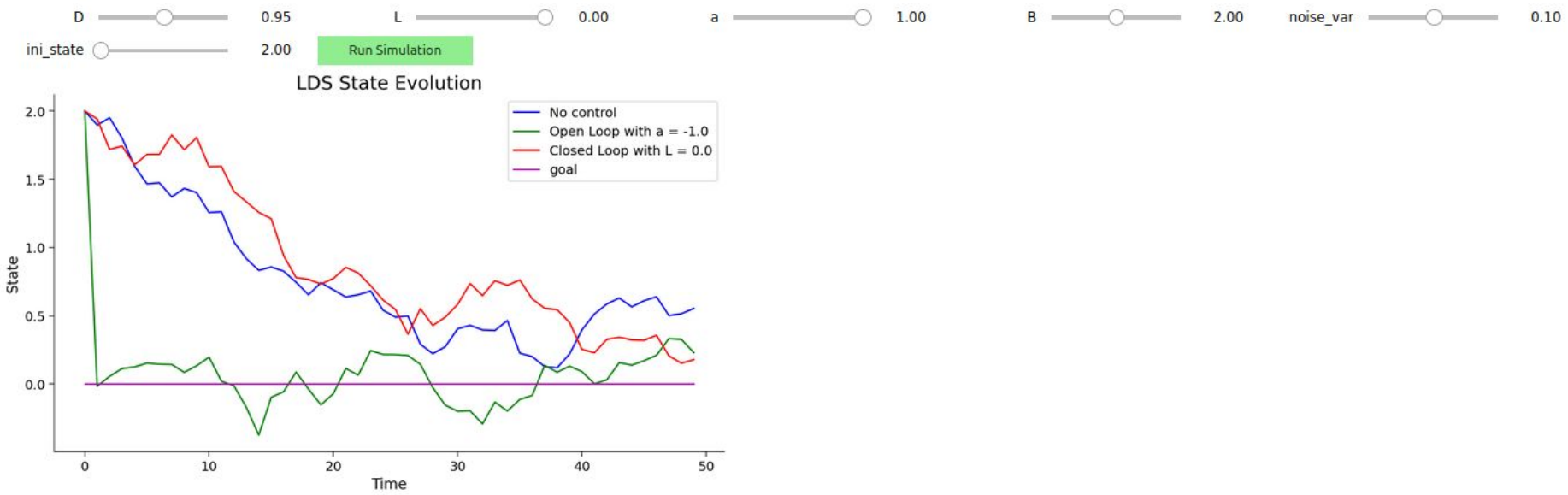
ini_state 2.00 Run Simulation



D 0.95 L 0.00 a -2.00 B 2.00 noise_var 0.10

ini_state 2.00 Run Simulation




D
0.95
L
0.00
a
1.00
B
2.00
noise_var
0.10
ini_state
2.00
Run Simulation
LDS State Evolution
No control
Open Loop with a = -1.0
Closed Loop with L = 0.0
goal
State
Time
2.0
1.5
1.0
0.5
0.0
0
10
20
30
40
50

Observations

* No control (blue): the dynamics parameter D controls how fast the dynamics decay towards 0. For -1<D<1, the system is stable and therefore approaches zero quickly. However, D>1 produces an unstable system - rapidly explodes (i.e., heads off to infinity)

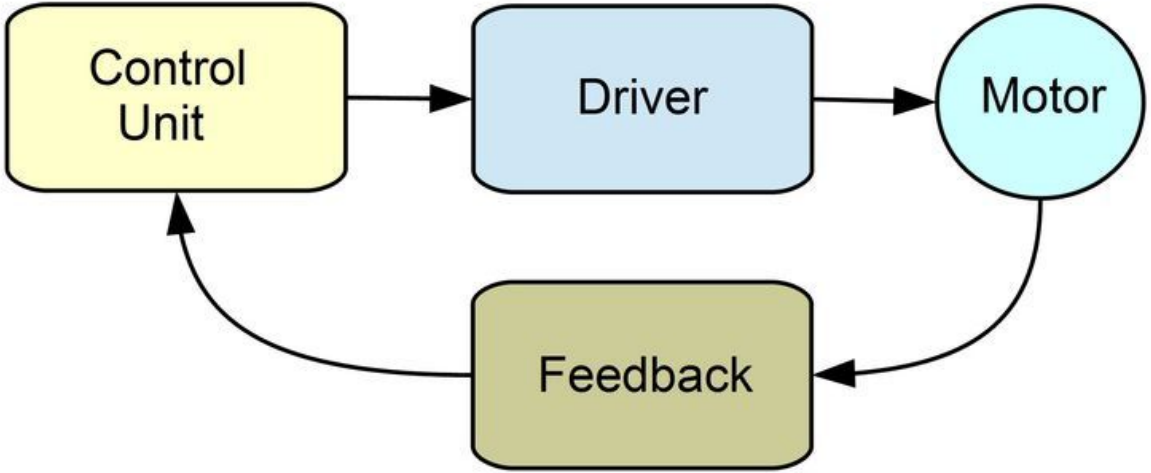
* Open-loop control: While the open-loop state often reaches the goal quickly, it may not stay there. Under high noise conditions, it tends to drift away from the goal.

* Closed-loop control: The closed-loop state (red curve) reaches the goal and stays there even in the presence of noise. It converges especially quickly for Ls around 0.45

In closed-loop control, we have a[t]=L[t] * s[t]. Note that with a constant control gain L[t]=L,
the state evolution equations can be rearranged to show that the stability of the closed-loop system now depends on the value of D+BL. If |D+BL|<1, our closed-loop system will be stable. More generally, the role of a closed-loop control input can be assumed as changing the system *dynamics* in an optimal way to reach the goal.

Open Loop Control System

Closed Loop Control System



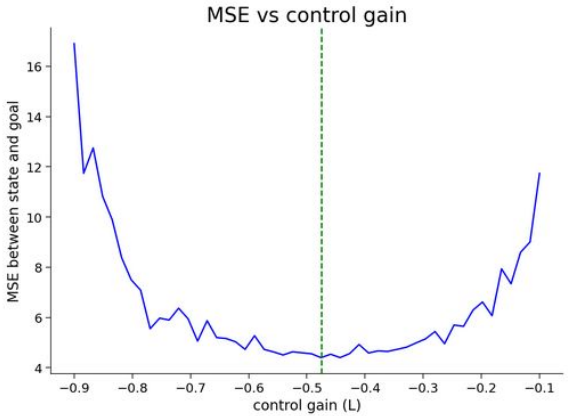
Why closed loop?

Change system dynamics in optimality to reach goal.
TASK: Find control gain that gives min MSE

Control gain

Gain is usually defined as ratio of the change in input to change in the output.

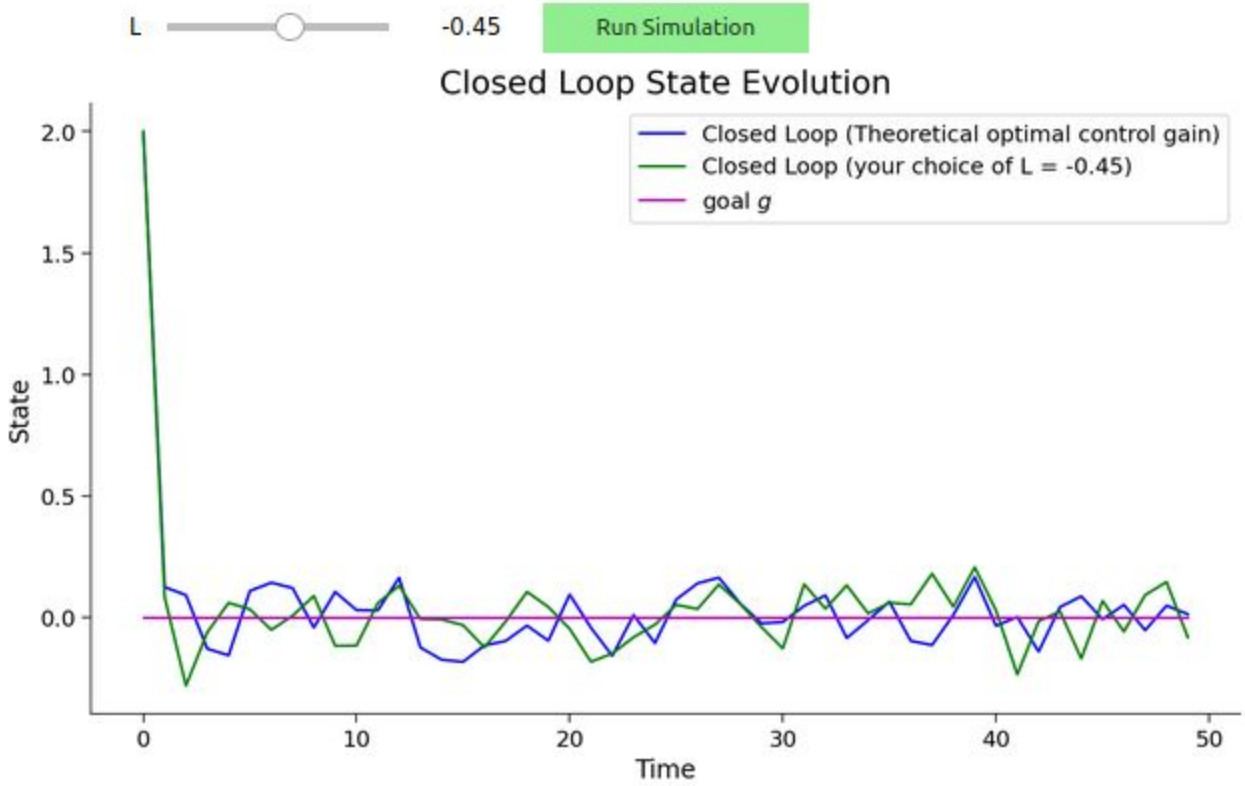
More the gain faster will be the loop response of the controller.

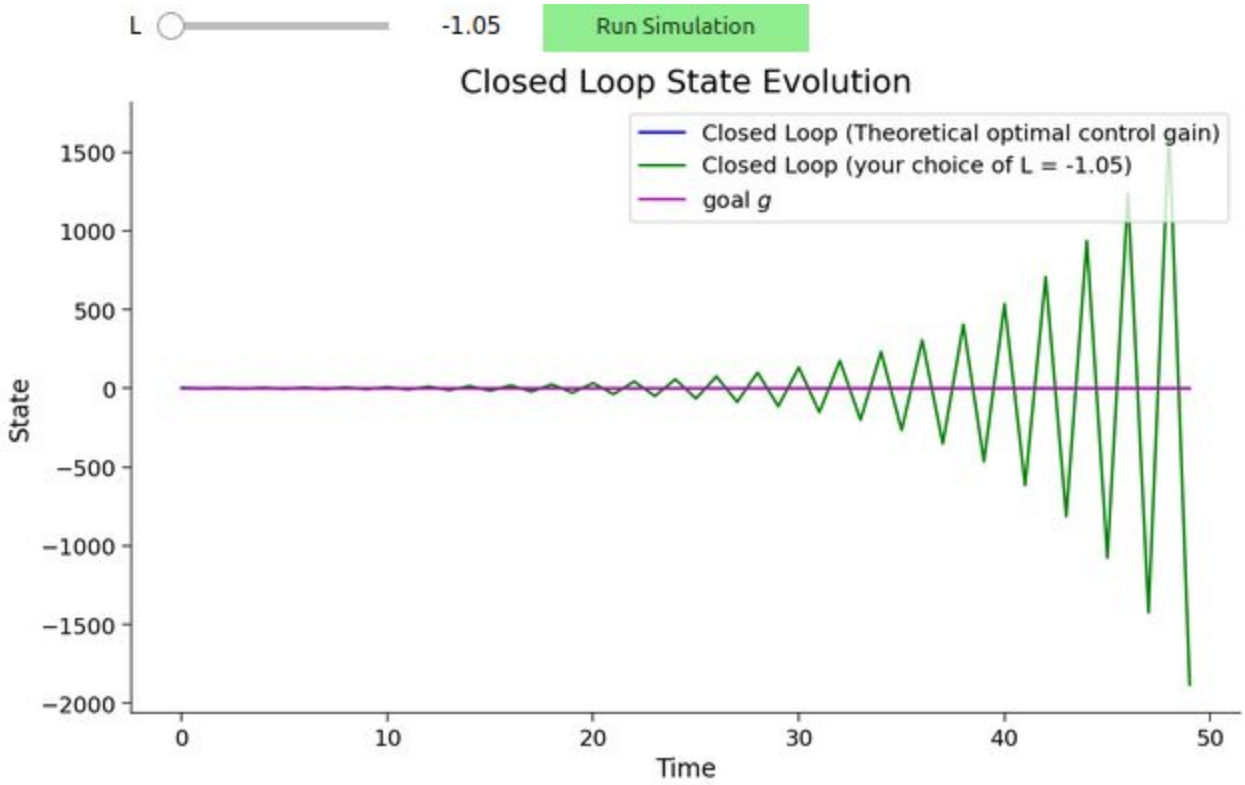


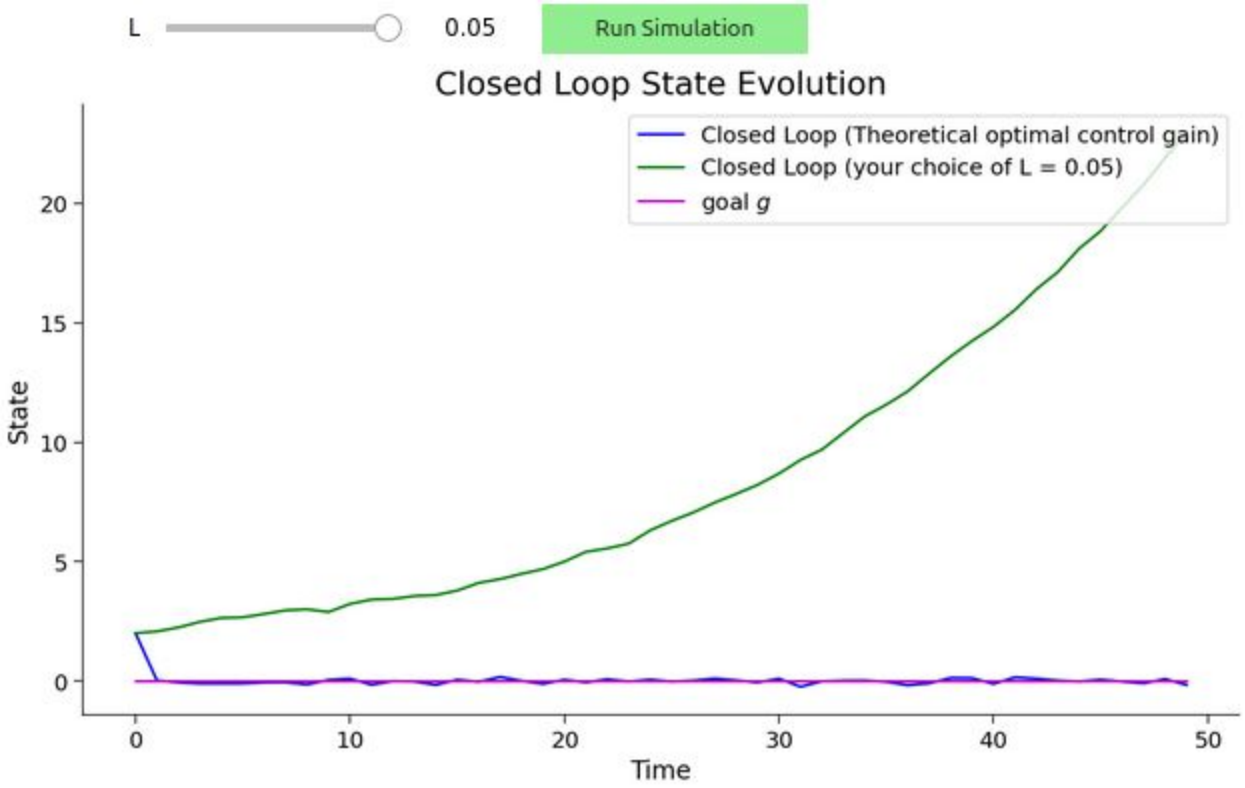
$$L = \frac{-D}{B}\left(0.95/2\right)$$

theoretical optimal gain control for min. MSE.

$$\begin{aligned} s_{t+1} &= Ds_t + Ba_t + w_t \\ &= Ds_t + B(Ls_t) + w_t \\ &= s_t(D + BL) + w_t \end{aligned}$$


L
-0.45
Run Simulation
Closed Loop State Evolution
Closed Loop (Theoretical optimal control gain)
Closed Loop (your choice of L = -0.45)
goal g
State
Time
2.0
1.5
1.0
0.5
0.0
0
10
20
30
40
50


L
-1.05
Run Simulation
Closed Loop State Evolution
Closed Loop (Theoretical optimal control gain)
Closed Loop (your choice of L = -1.05)
goal g
State
Time
1500
1000
500
0
−500
−1000
−1500
−2000
0
10
20
30
40
50


L
0.05
Run Simulation
Closed Loop State Evolution
Closed Loop (Theoretical optimal control gain)
Closed Loop (your choice of L = 0.05)
goal g
State
Time
0
5
10
15
20
0
10
20
30
40
50

Observations

Optimal control gain (L = -0.45) takes a short amount of time to get to the goal, and then stays there. We can try to get to the goal in an even shorter time using an 'over-ambitious' control gain (L < -0.45), but this may actually overshoot the goal and may cause oscillations in the system,
thus increasing the MSE. On the other hand, an 'under-ambitious' control gain takes a longer time to get to the goal and thus increases the MSE.
Finally, at L>0, the system runs away to infinity.

Why is L=-D/B optimal for reaching our goal? Recall that our next state is (D+B*L)*s[t] + noise. Plugging that L=-D/B causes that leading term to become zero, which is our goal. Since the noise has zero mean, it's not possible to do any better!

Optimal gain control

Minimum MSE

Over ambitious control gain

Oscillations in the system (overshoot the goal)

Under ambitious control gain

Gets to goal very slowly (increases MSE)

Linear Quadratic Regulator

The theory of optimal control is concerned with operating a dynamic system at minimum cost. The case where the system dynamics are described by a set of linear differential equations and the cost is described by a quadratic function is called the LQ problem.

Additional constraints on the system

$a_t \gg$ optimal α (system explodes)

keep a_t small as possible (maintain good control)

Static control gain $L_t \equiv L$. { principled way of designing optimal control input

Set up cost function

finite horizon LQR problem.

$$J(s,a) = J_{state}(s) + \lambda J_{control}(a)$$

$$= \sum_{t=0}^{T} (s_t - g)^2 + \lambda \sum_{t=0}^{T-1} a_t^2$$

weight on control effort cost

quadratic cost function

$$a = \{a_t\}_{t=0}^{T-1}$$

$$s = \{s_t\}_{t=0}^{T}$$

$$q = 0 \Rightarrow J_{state}(s) = \sum_{t=0}^{T} s_t^2$$

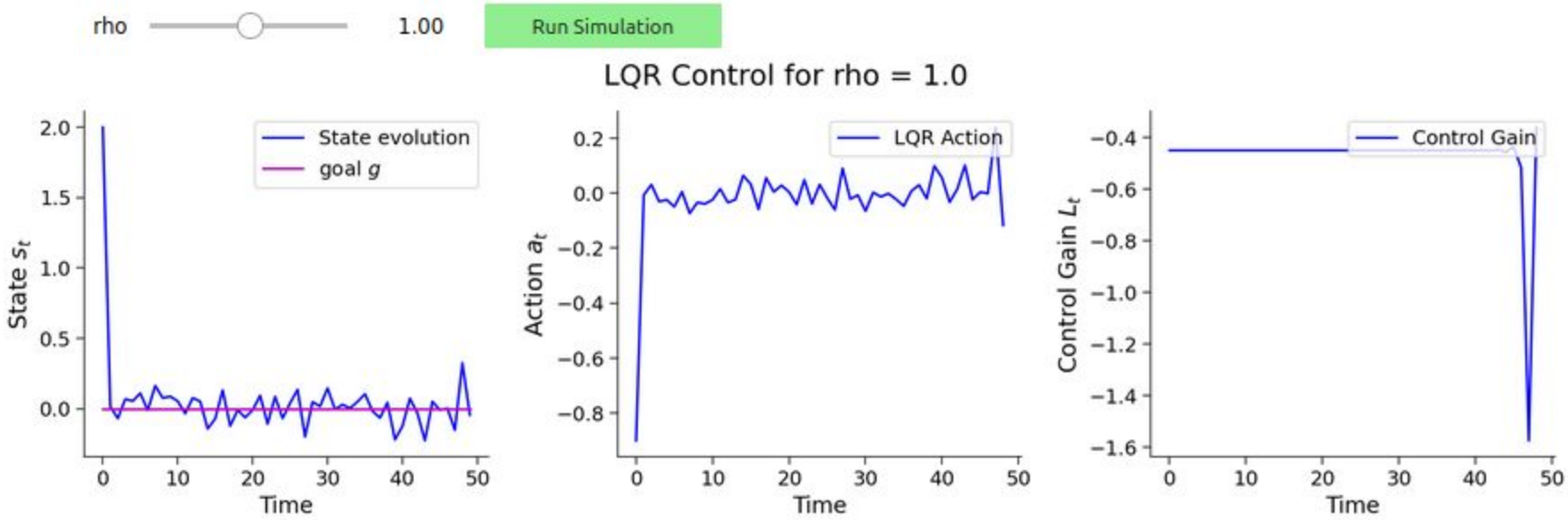
Goals of LQR problem

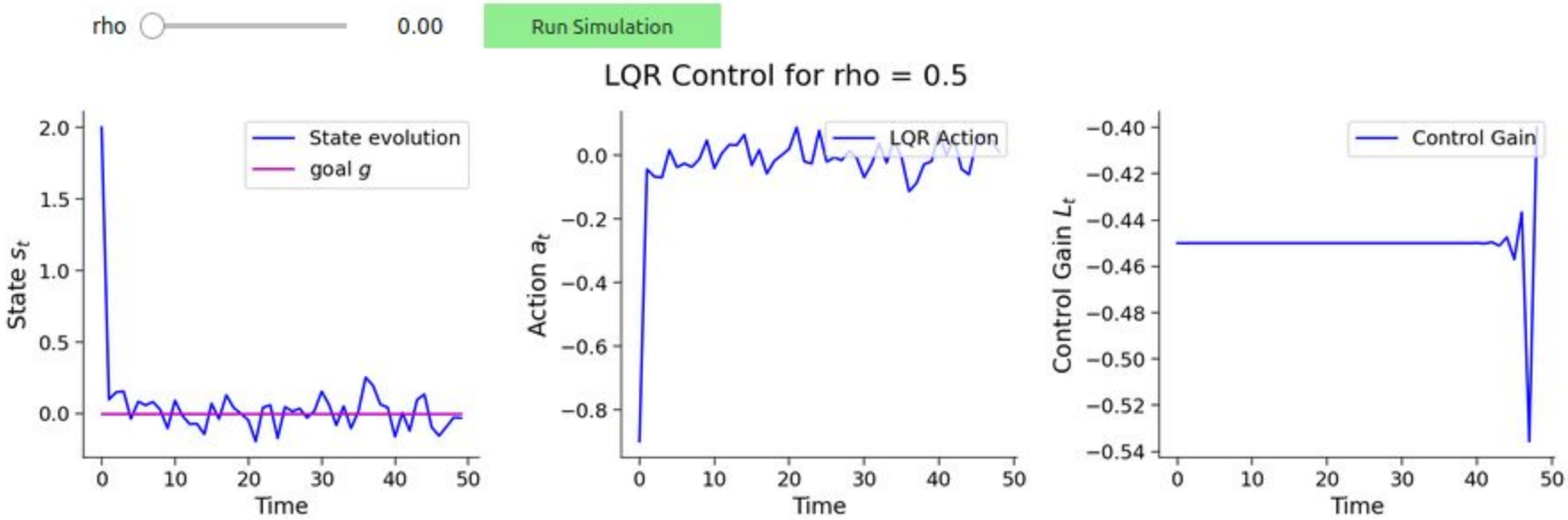
find control a such that $J(s, a)$ is minimised

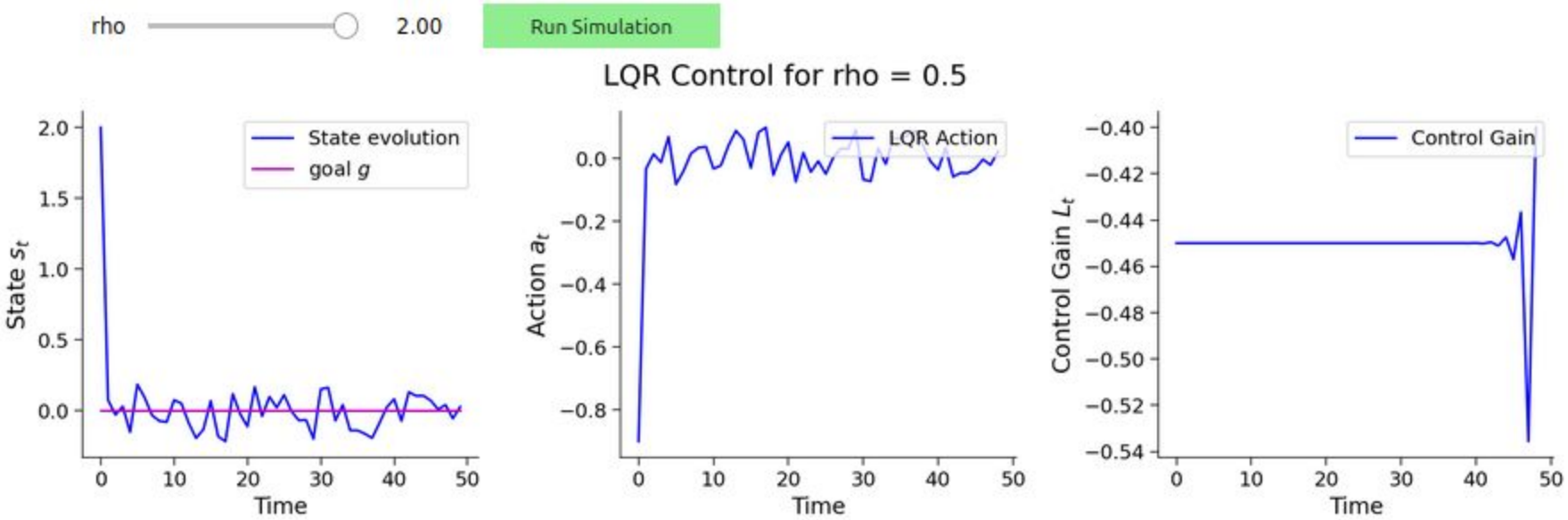
find control gain @ each time point

$$\underset{\{L_t\}_{t=0}^{T-1}}{\operatorname{argmin}} \; J(s, a)$$

$$a_t = L_t s_t$$


rho
1.00
Run Simulation
LQR Control for rho = 1.0
State evolution
goal g
State s_t
Time
LQR Action
Action a_t
Time
Control Gain
Control Gain L_t
Time


rho
0.00
Run Simulation
LQR Control for rho = 0.5
State evolution
goal g
State s_t
Time
LQR Action
Action a_t
Time
Control Gain
Control Gain L_t
Time


rho
2.00
Run Simulation
LQR Control for rho = 0.5
State evolution
goal g
State s_t
Time
LQR Action
Action a_t
Time
Control Gain
Control Gain L_t
Time

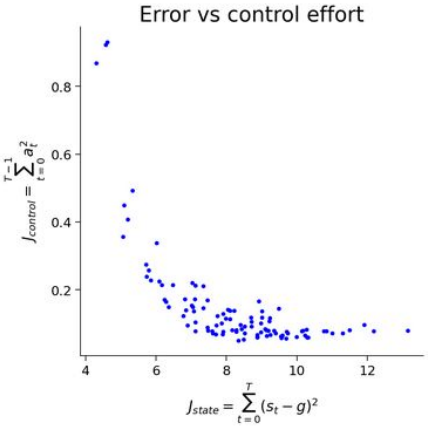
Observations

* rho=0 will get you the same cost and control gain as by zeroing out the cost term.

* A small value for rho will have a similar solution but with potentially large values for |a[t]|.

* A large value for rho, will lead to small values for |a[t]|. * The control gain becomes more time-varying (as opposed to fairly static) for large rho. For some parameter values, L[t] oscillates during the entire trajectory in order to keep |a_t| low. Try D = 0.9, B = 2 and rho = 2.

tradeoff between state cost and control cost



For a desired value of the state cost, we cannot reach a lower control cost than the curve in the above plot. Similarly, for a desired value of the control cost, we must accept that amount of state cost.

implications of trade-off

① for limited amt of fuel/resource,

more wanted cost = J_{wanted}^{more}

② can't track state with accuracy > J_{state} as given by graph.

goal moves ⇒ goal ~~state~~ trajectory ∫ target state g_t +0

cost function

$$J(a) = \sum_{t=0}^{T} (s_t - g_t)^2 + \int \sum_{t=0}^{T-1} (a_t - \bar{a}_t)^2$$

$\bar{a}_t$ → desired action

Controller considers goal for next time step, designs preliminary control action that gets state @ next time step to desired goal.

without taking into account noise n_t

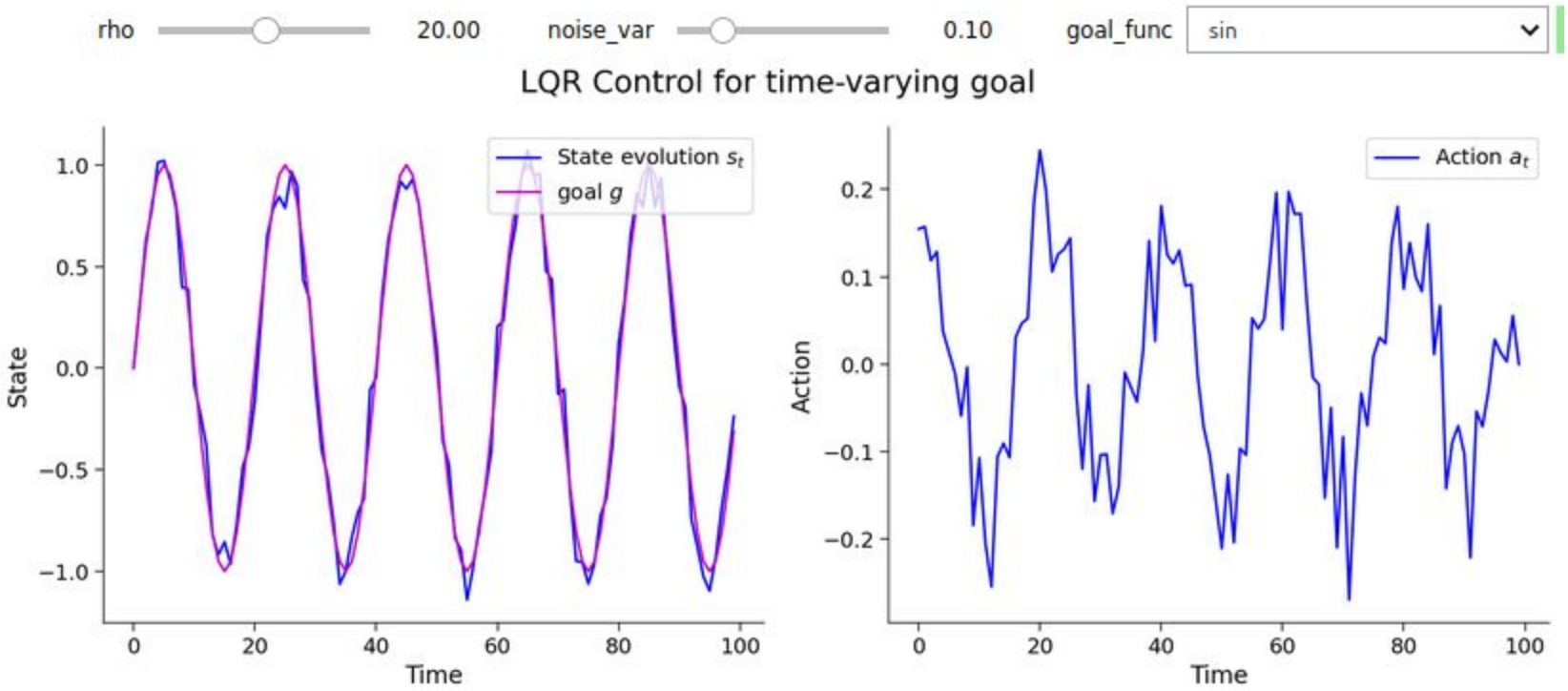
design $\bar{a}_t$ such that: $s_{t+1} = g_{t+1}$

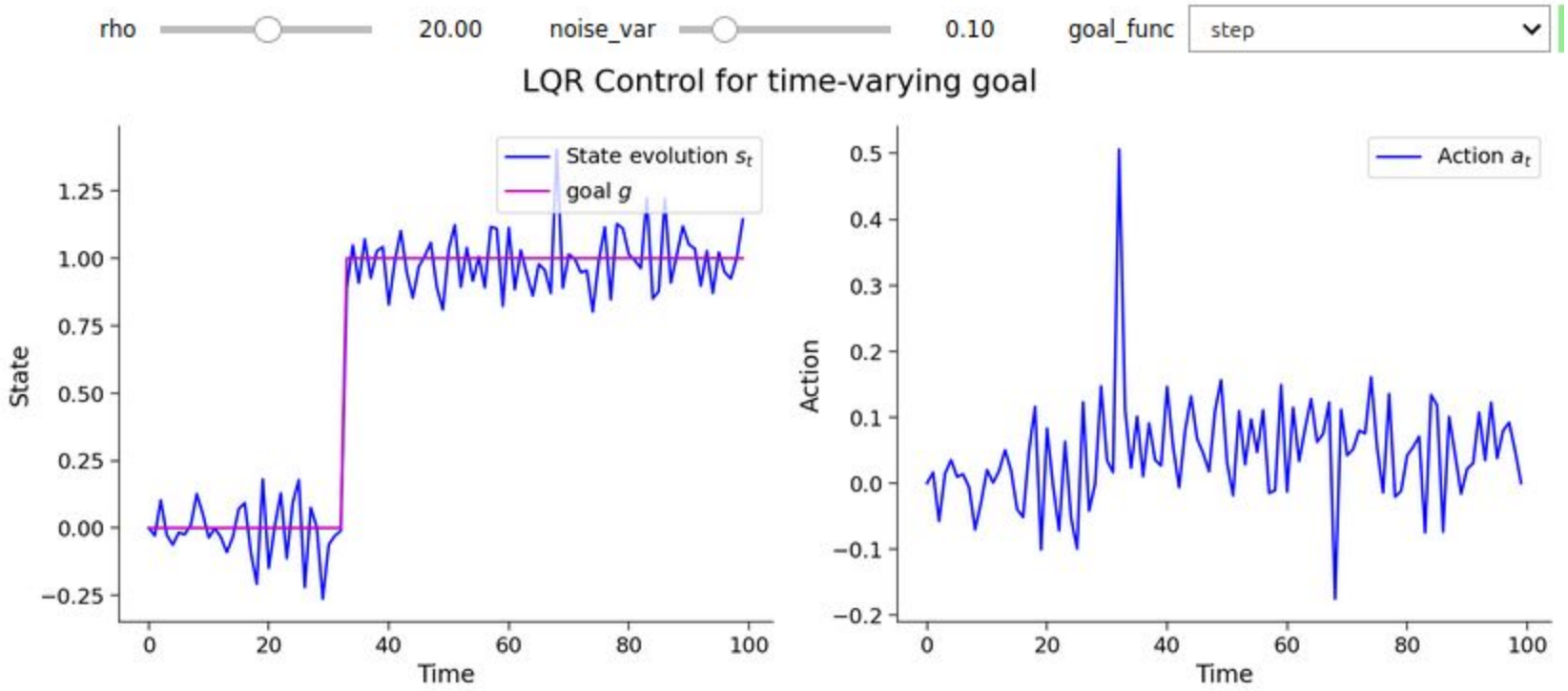
$$g_{t+1} = Ds_t + B\bar{a}_t$$

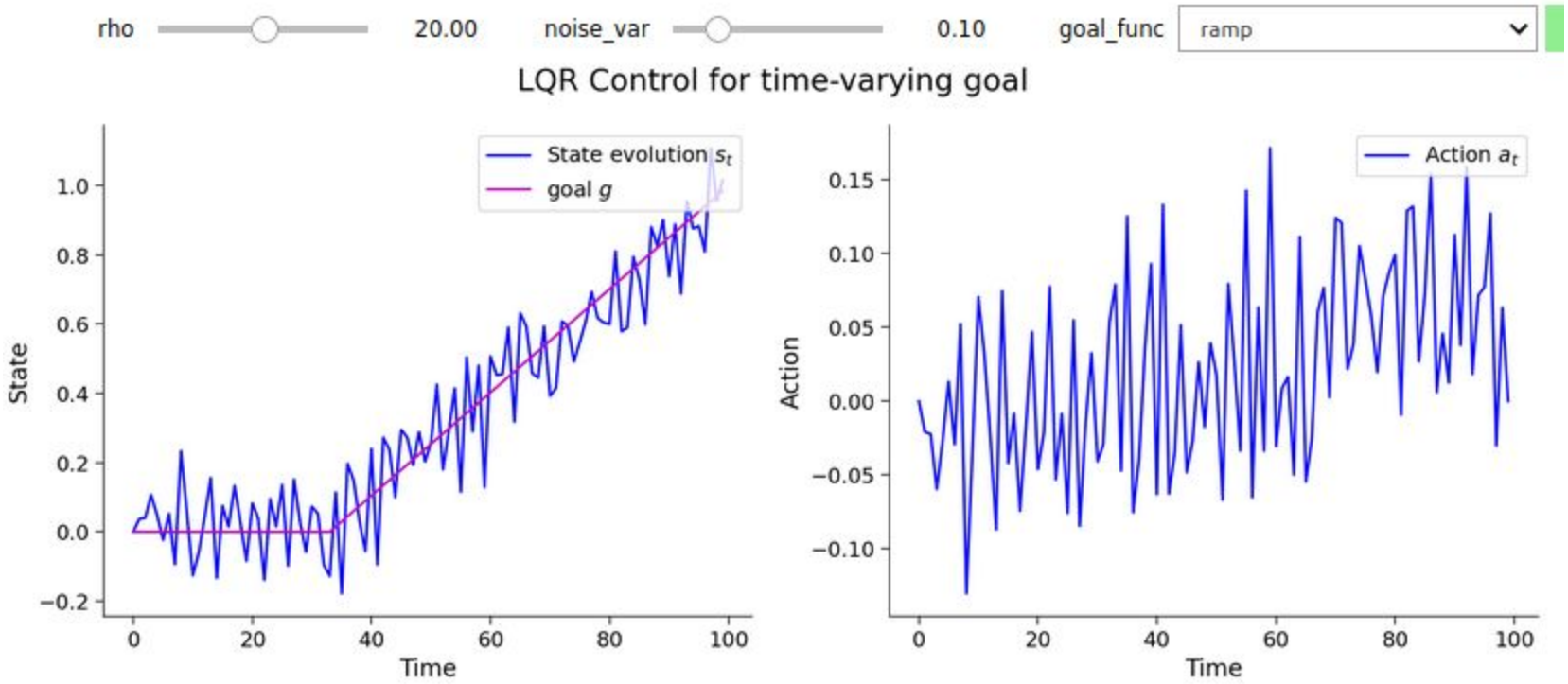
$$\bar{a}_t = \frac{-Ds_t + g_{t+1}}{B}$$

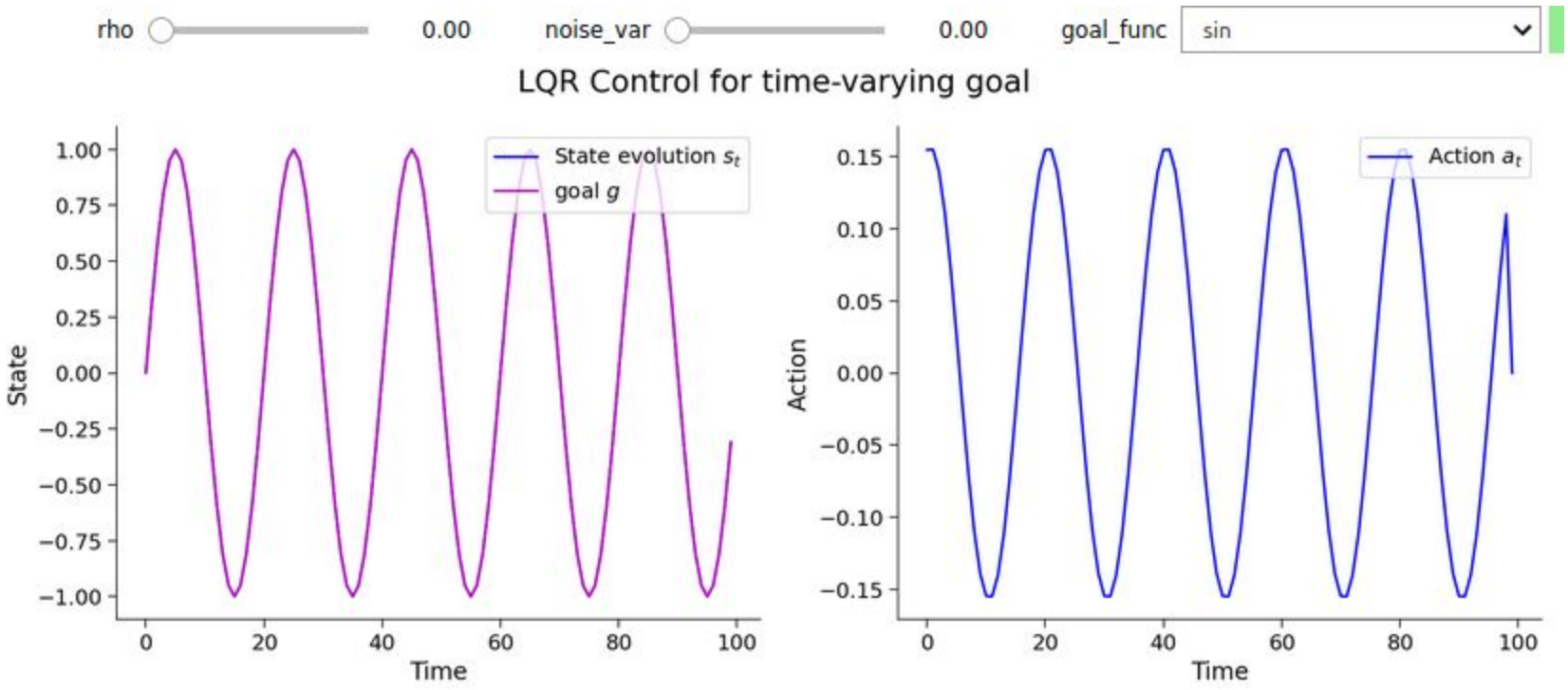
final control action: $a_t = \bar{a}_t + L_t(s_t - g_t)$

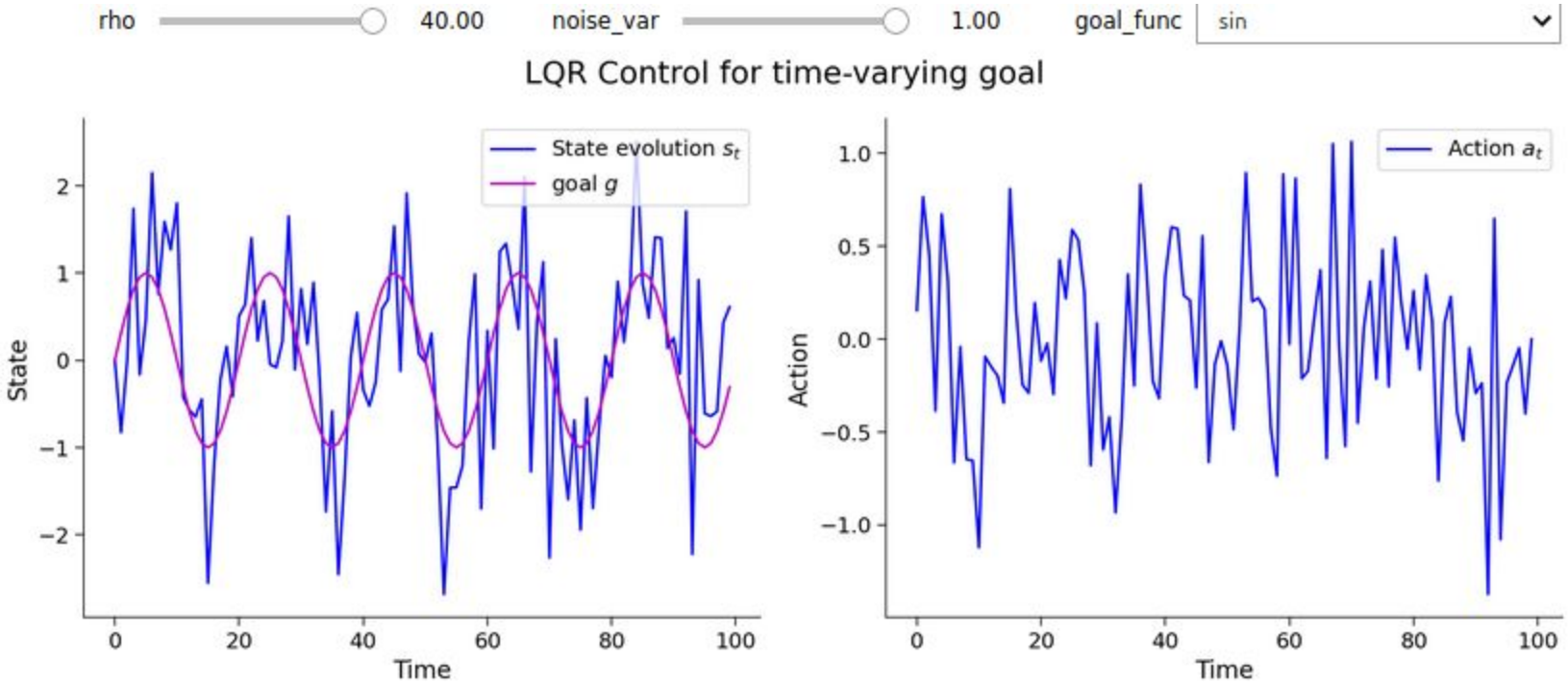
control gain

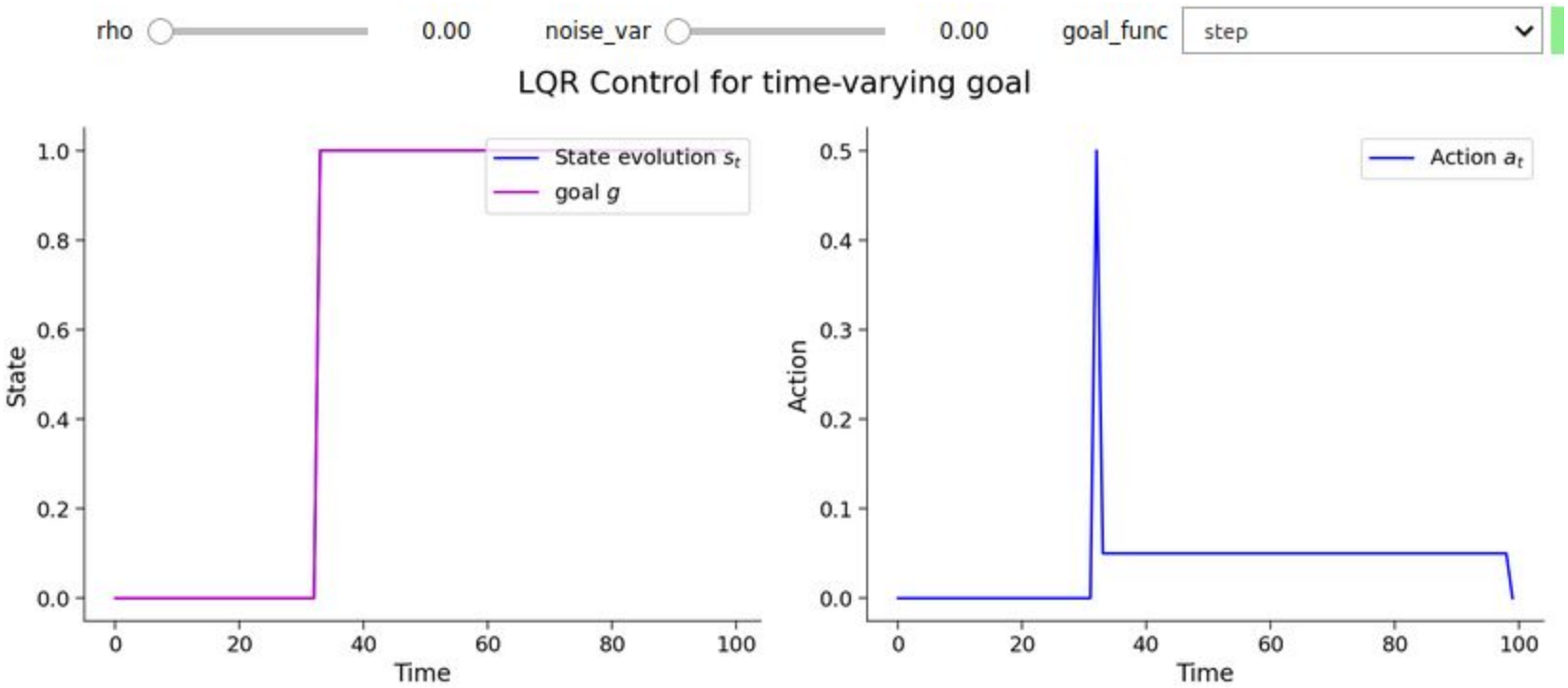

rho
20.00
noise_var
0.10
goal_func
sin
LQR Control for time-varying goal
State evolution s_t
goal g
State
Time
Action a_t
Action
Time

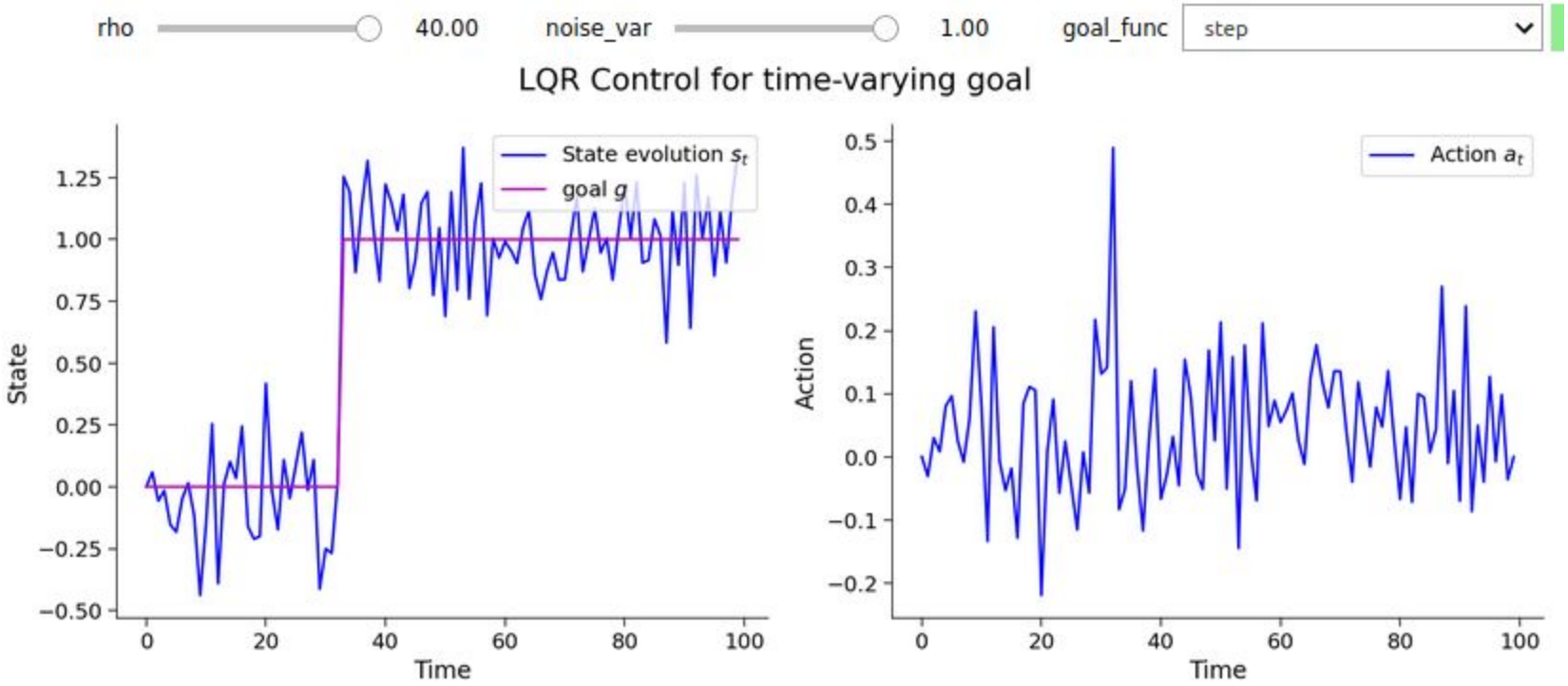

rho
20.00
noise_var
0.10
goal_func
step
LQR Control for time-varying goal
State evolution s_t
goal g
State
Time
Action a_t
Action
Time

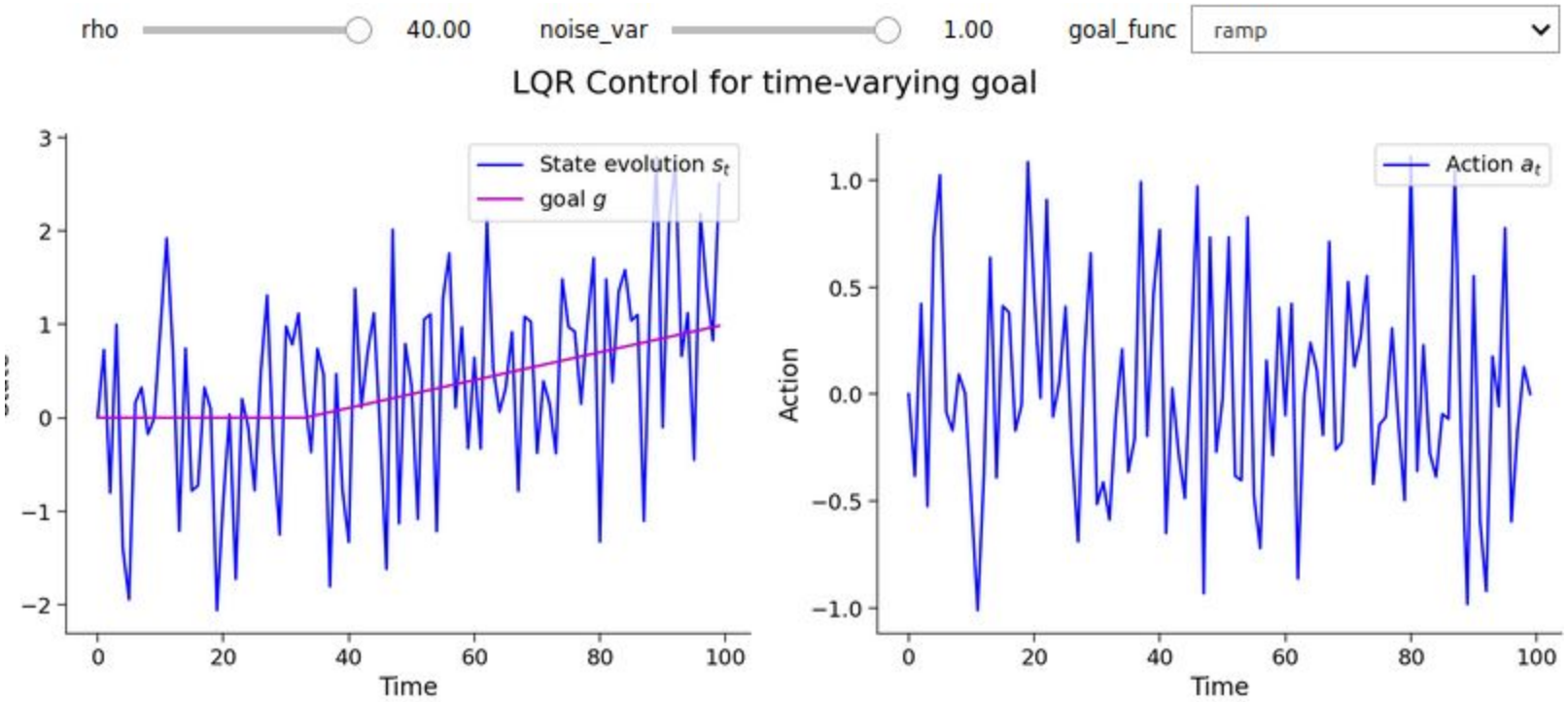

rho
20.00
noise_var
0.10
goal_func
ramp
LQR Control for time-varying goal
State evolution s_t
goal g
State
Time
Action a_t
Action
Time

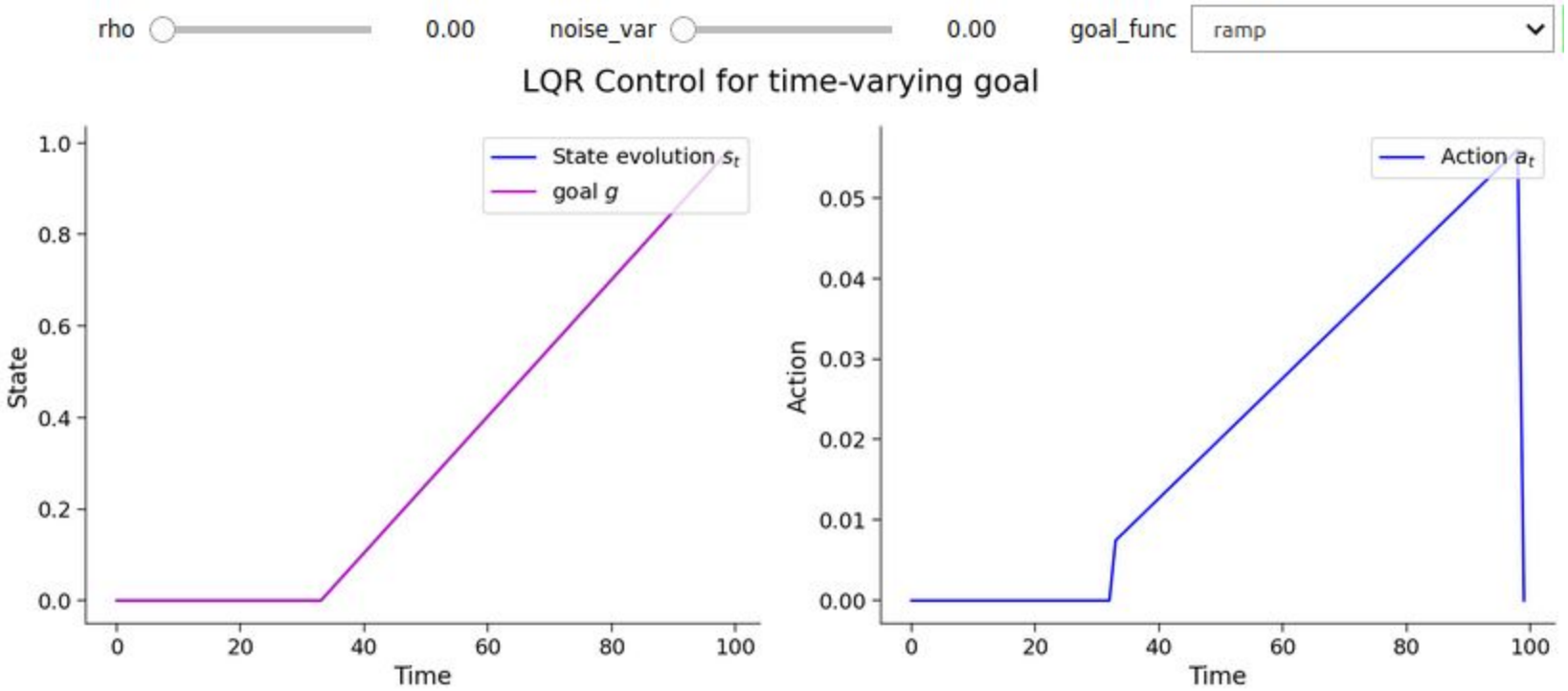

rho
0.00
noise_var
0.00
goal_func
sin
LQR Control for time-varying goal
State evolution s_t
goal g
State
Time
Action a_t
Action
Time


rho
40.00
noise_var
1.00
goal_func
sin
LQR Control for time-varying goal
State evolution s_t
goal g
State
Time
Action a_t
Action
Time


rho
0.00
noise_var
0.00
goal_func
step
LQR Control for time-varying goal
State evolution s_t
goal g
State
Time
Action a_t
Action
Time


rho
40.00
noise_var
1.00
goal_func
step
LQR Control for time-varying goal
State evolution s_t
goal g
State
Time
Action a_t
Action
Time


rho
40.00
noise_var
1.00
goal_func
ramp
LQR Control for time-varying goal
State evolution s_t
goal g
Time
Action a_t
Action
Time


rho
0.00
noise_var
0.00
goal_func
ramp
LQR Control for time-varying goal
State evolution s_t
goal g
State
Time
Action a_t
Action
Time

Observations

* The system follows time varying goals rather well, with little change to the cost function and the control equations.

* Setting rho=0 leads to noise in the first part of the time series. Here, we see that the control cost in fact acts as a regularizer.

* Larger values of the process noise variance lead to a higher MSE between the state and the desired goal.

Partially observed state

(controller doesn't have full access to the state)

Linear Quadratic Gaussian

Estimate state

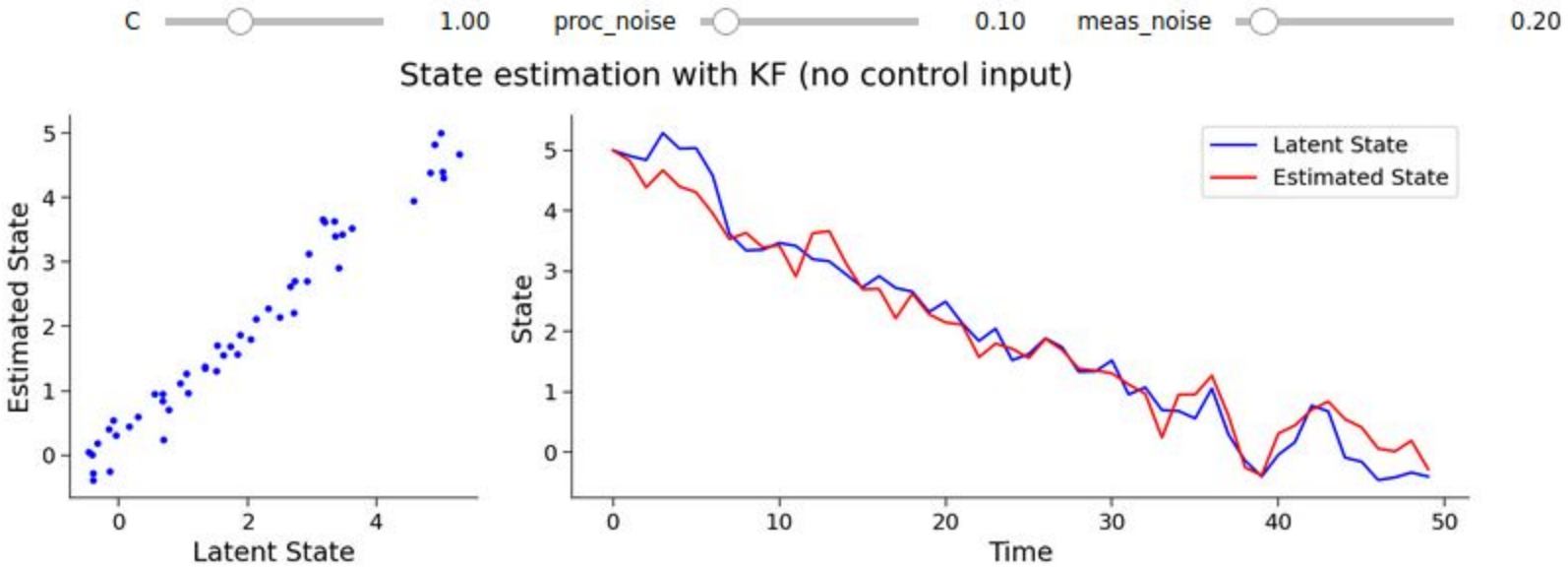
(hint: yesterday's lecture)

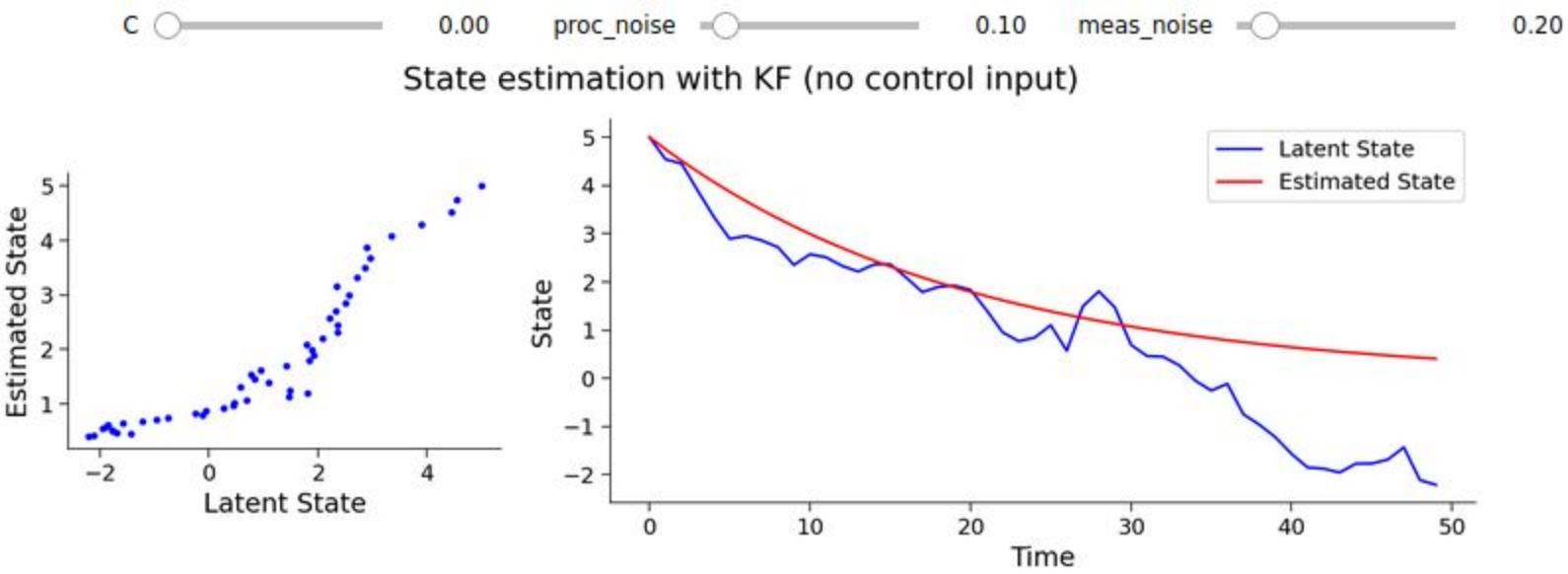
DESIGNING ACTIONS BASED ON ESTIMATES

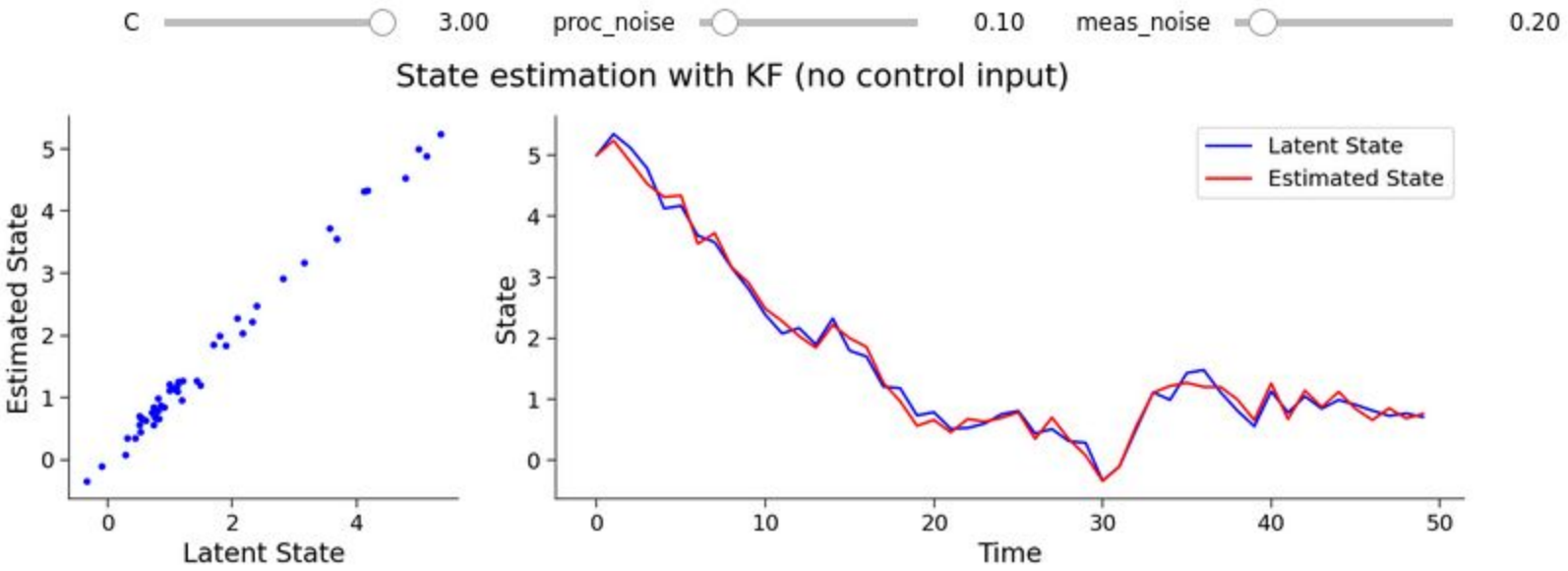
Hint: Previous slide(LQR)

Linear Quadratic Gaussian

This control law which is known as the **LQG** controller, is unique and it is simply a combination of a Kalman filter (a linear–quadratic state estimator (LQE)) together with a linear–quadratic regulator (LQR). Used when the controller does not have full access to the state. implements filtering in closed-loop feedback. It is a combination of generating samples (states st) and filtering (generating state estimates s^t). Thus, each st+1 gets an input at, which itself depends on the state estimate of the last time step s^t.

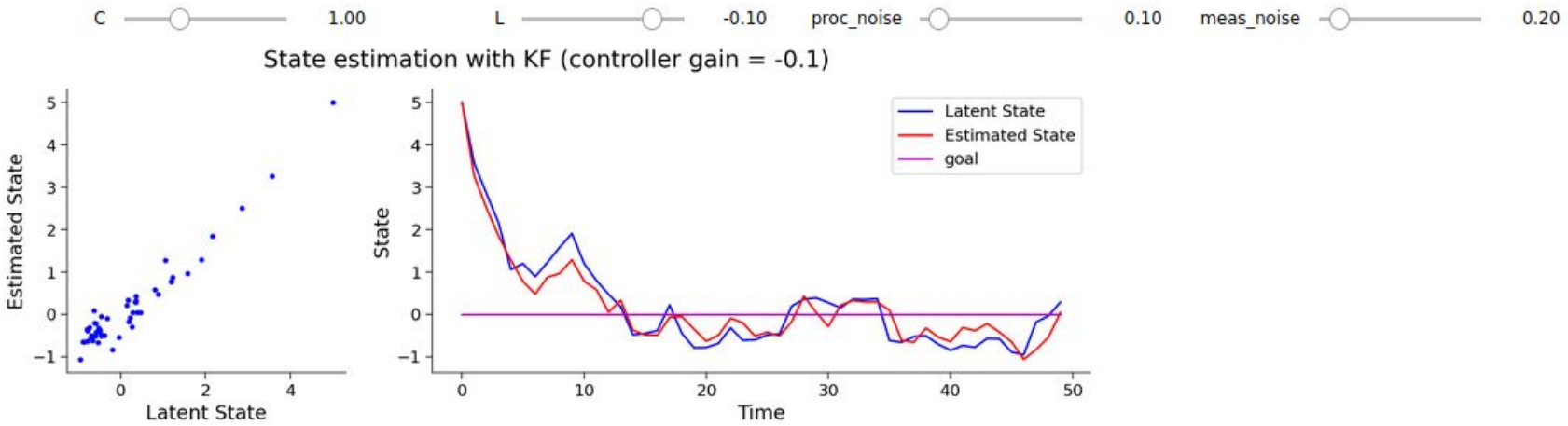

C
1.00
proc_noise
0.10
meas_noise
0.20
State estimation with KF (no control input)
Estimated State
Latent State
Latent State
Estimated State
State
Time

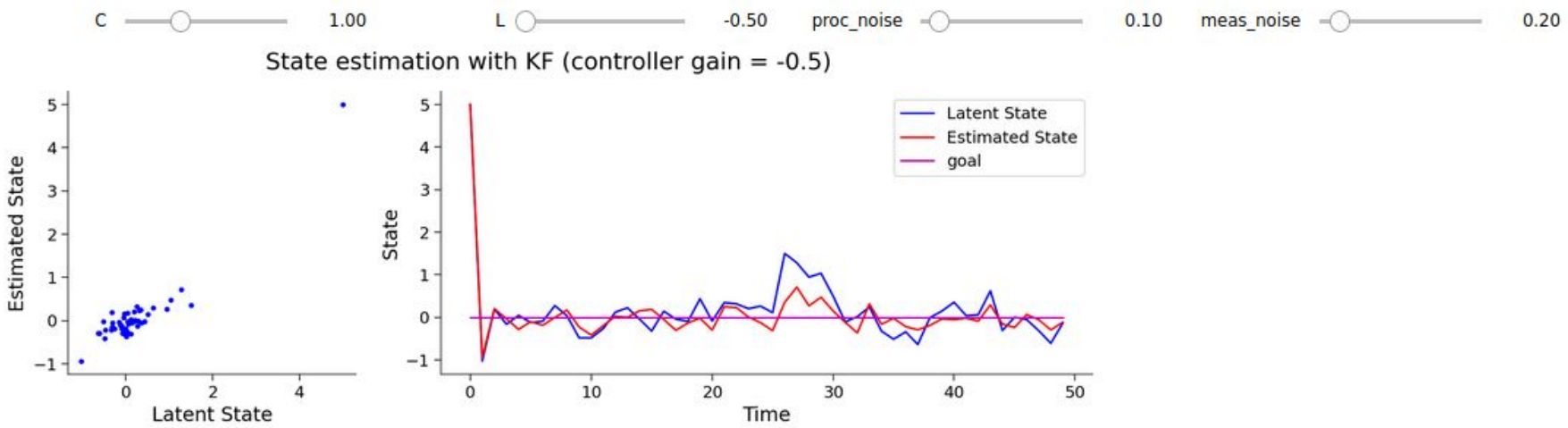

C
0.00
proc_noise
0.10
meas_noise
0.20
State estimation with KF (no control input)
Estimated State
Latent State
Latent State
Estimated State
State
Time

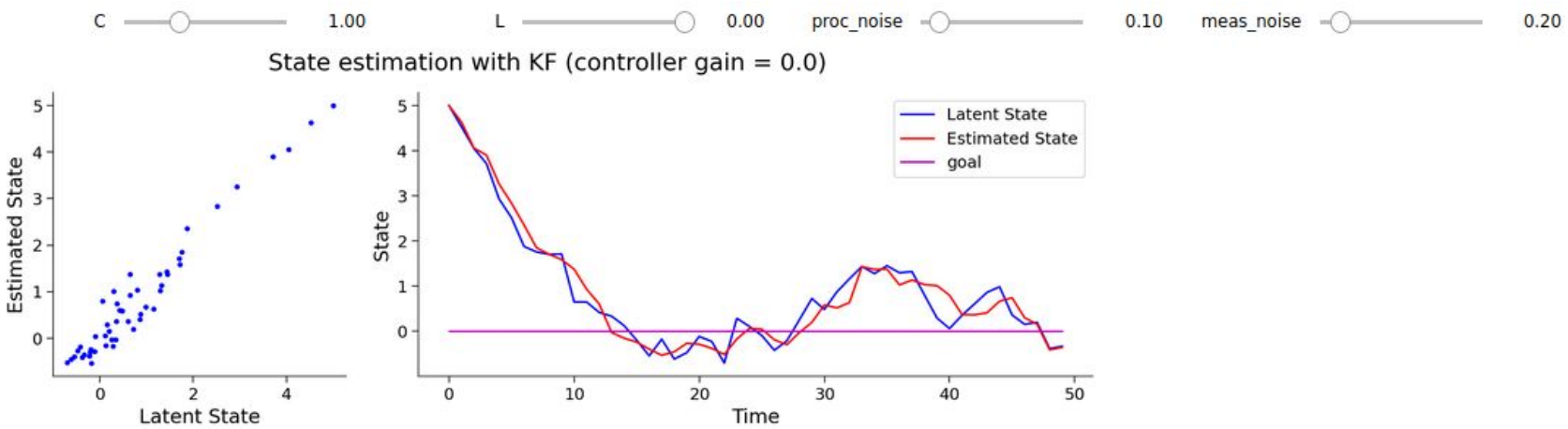

C
3.00
proc_noise
0.10
meas_noise
0.20
State estimation with KF (no control input)
Estimated State
Latent State
State
Time
Latent State
Estimated State

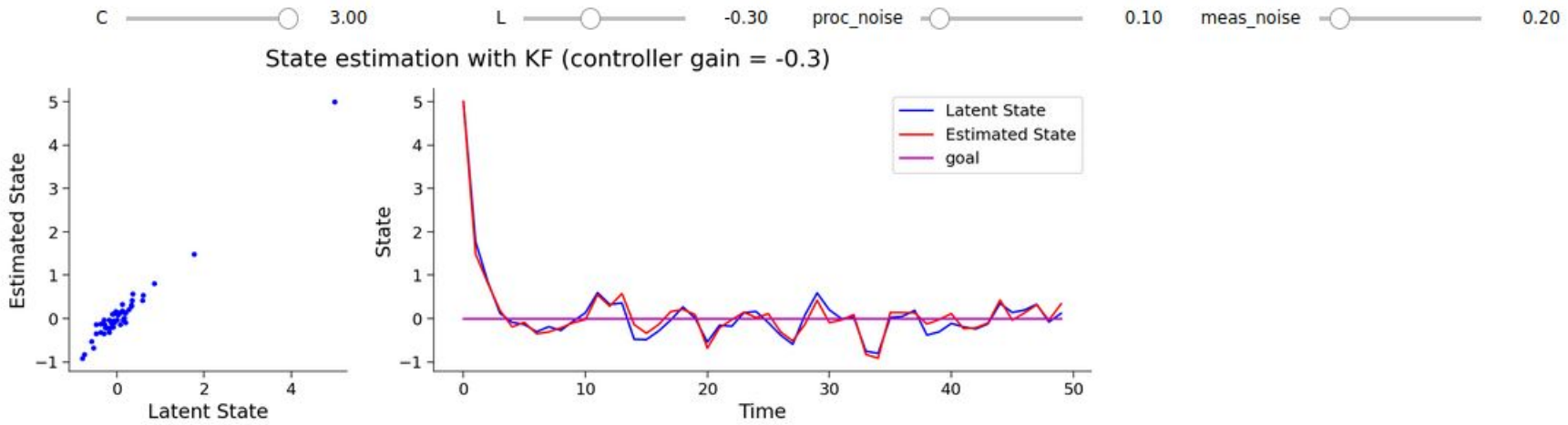
Observations

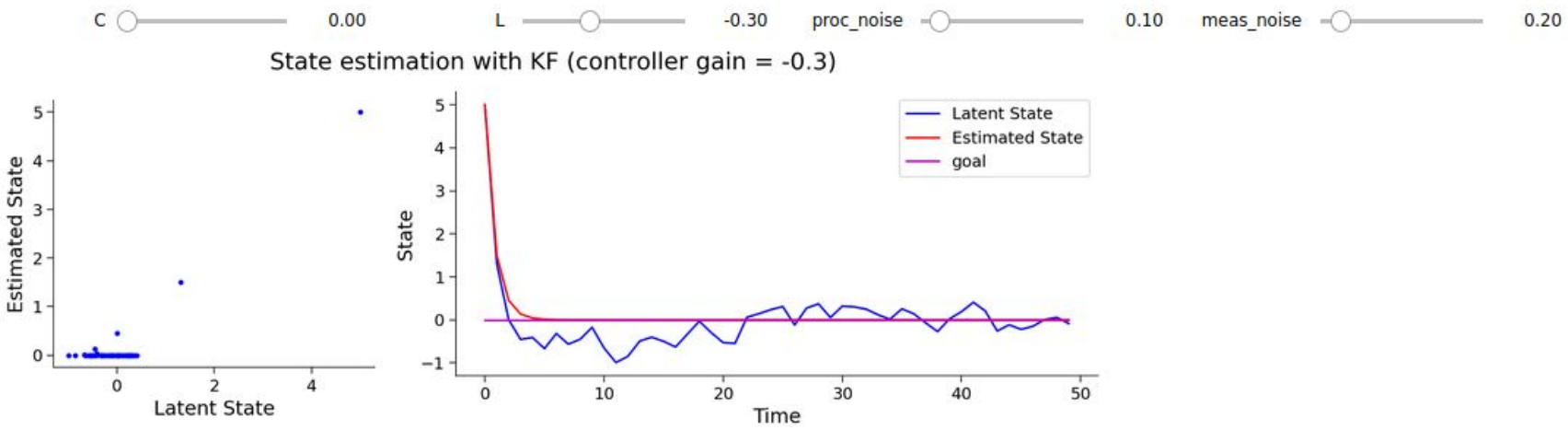
Kalman filter generally estimates the latent state accurately, even with fairly high noise levels, except when C=0.


C
1.00
L
-0.10
proc_noise
0.10
meas_noise
0.20
State estimation with KF (controller gain = -0.1)
Estimated State
Latent State
State
Time
Latent State
Estimated State
goal

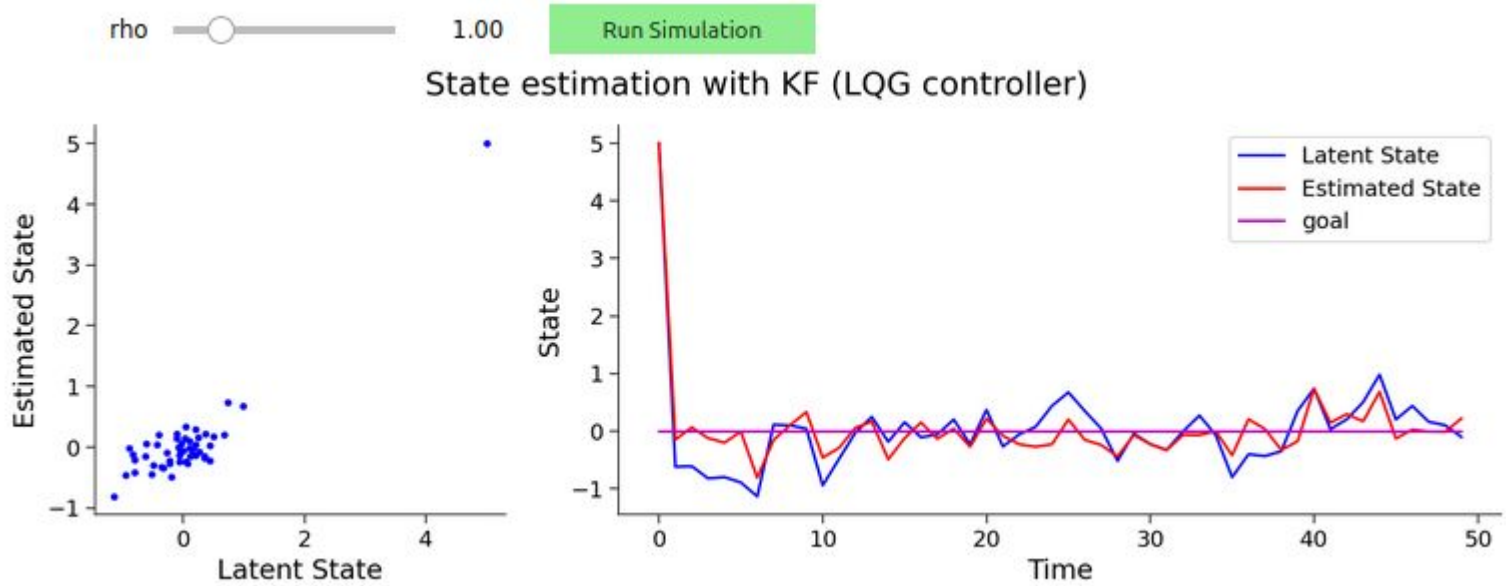

C
1.00
L
-0.50
proc_noise
0.10
meas_noise
0.20
State estimation with KF (controller gain = -0.5)
Estimated State
Latent State
State
Time
Latent State
Estimated State
goal

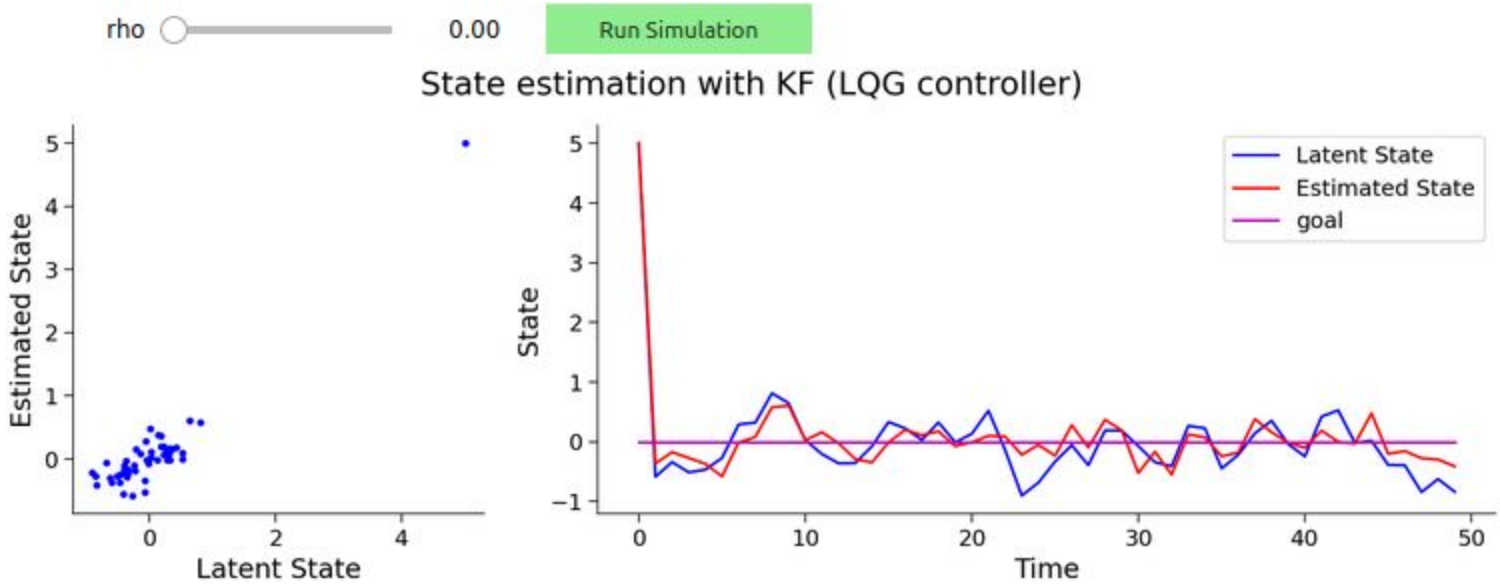

C
1.00
L
0.00
proc_noise
0.10
meas_noise
0.20
State estimation with KF (controller gain = 0.0)
Estimated State
Latent State
State
Time
Latent State
Estimated State
goal

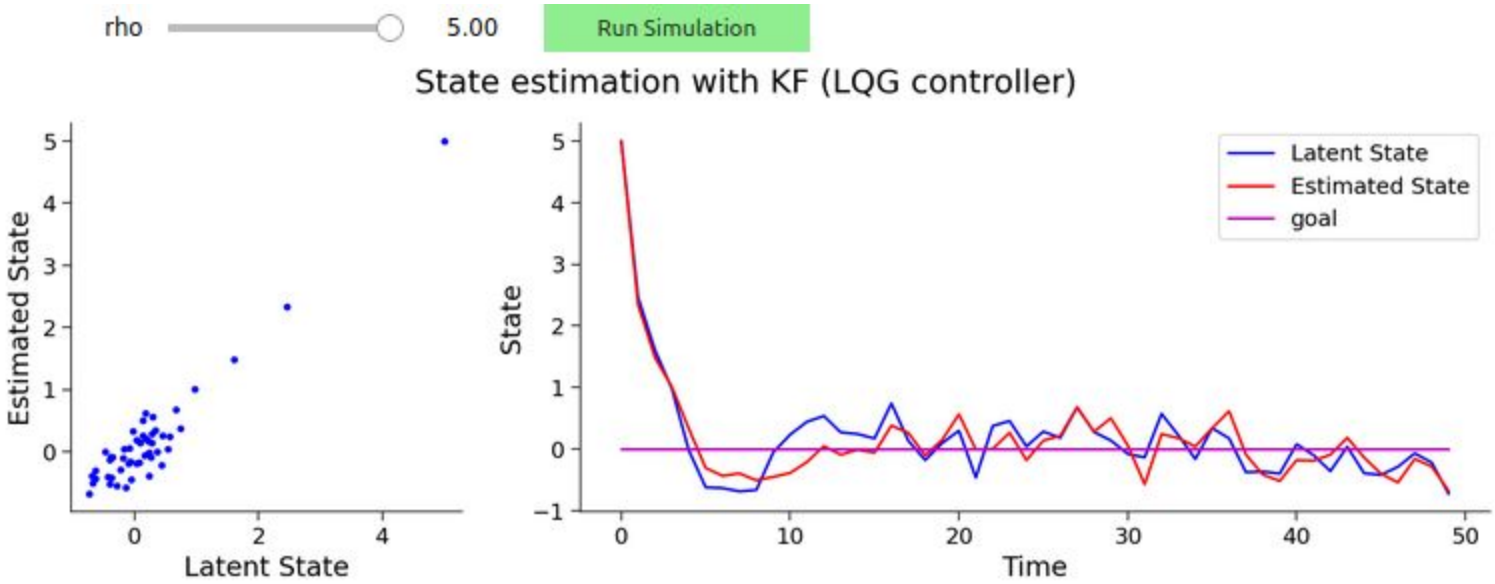

C
3.00
L
-0.30
proc_noise
0.10
meas_noise
0.20
State estimation with KF (controller gain = -0.3)
Estimated State
Latent State
State
Time
Latent State
Estimated State
goal


C
0.00
L
-0.30
proc_noise
0.10
meas_noise
0.20
State estimation with KF (controller gain = -0.3)
Estimated State
Latent State
State
Time
Latent State
Estimated State
goal

LQG controller gain, where the control gain is from a system with an infinite-horizon. In this case, the optimal control gain turns out to be a constant.




rho
0.00
Run Simulation
State estimation with KF (LQG controller)
Estimated State
Latent State
State
Time
Latent State
Estimated State
goal


rho
5.00
Run Simulation
State estimation with KF (LQG controller)
Estimated State
Latent State
State
Time
Latent State
Estimated State
goal

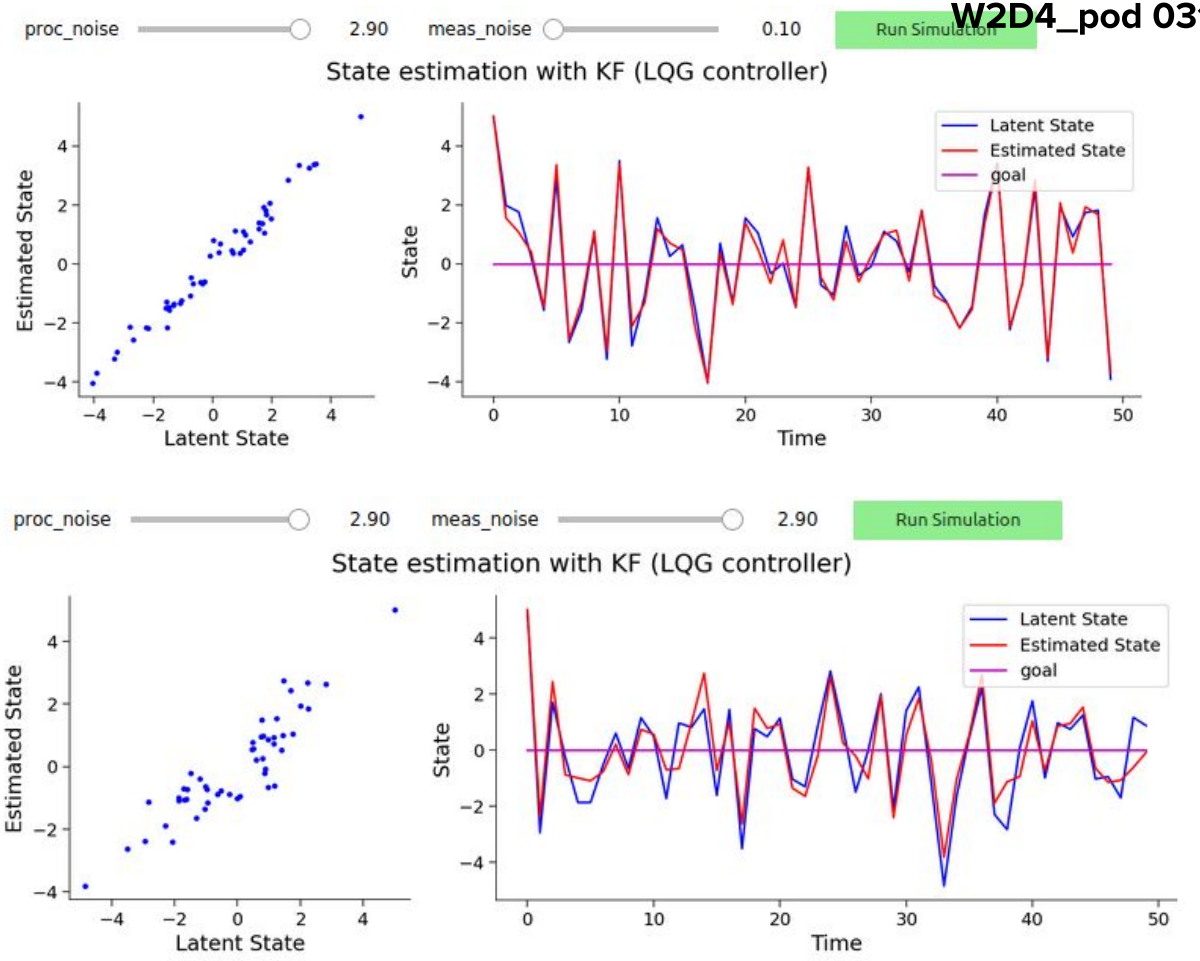
Noise!

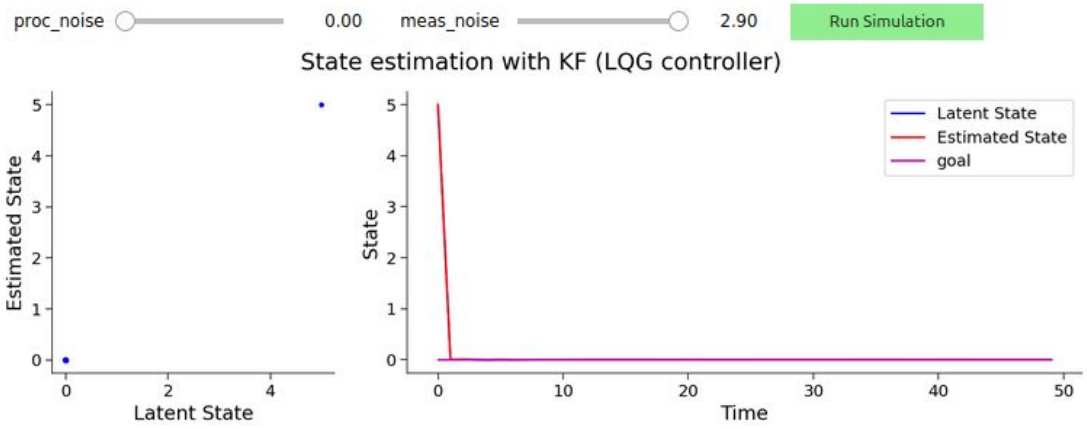
1. *Measurement noise (harder to estimate states)*
2. *Process noise*

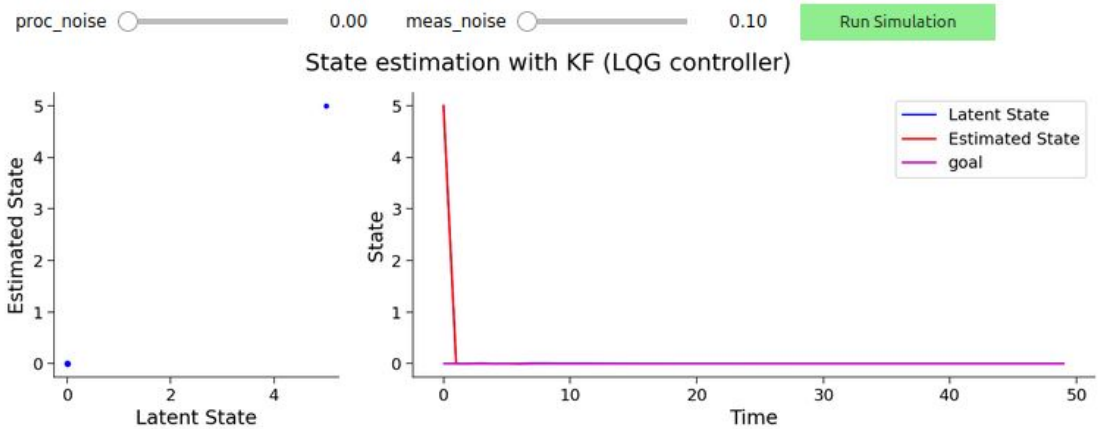
Process noise

the "**process noise**" represents the idea/feature that the state of the system changes over time, but we do not know the exact details of when/how those changes occur, and thus we need to model them as a random **process**.

how process noise and measurement noise influences the controlled state.




proc_noise
0.00
meas_noise
2.90
Run Simulation
State estimation with KF (LQG controller)
Estimated State
Latent State
State
Time
Latent State
Estimated State
goal

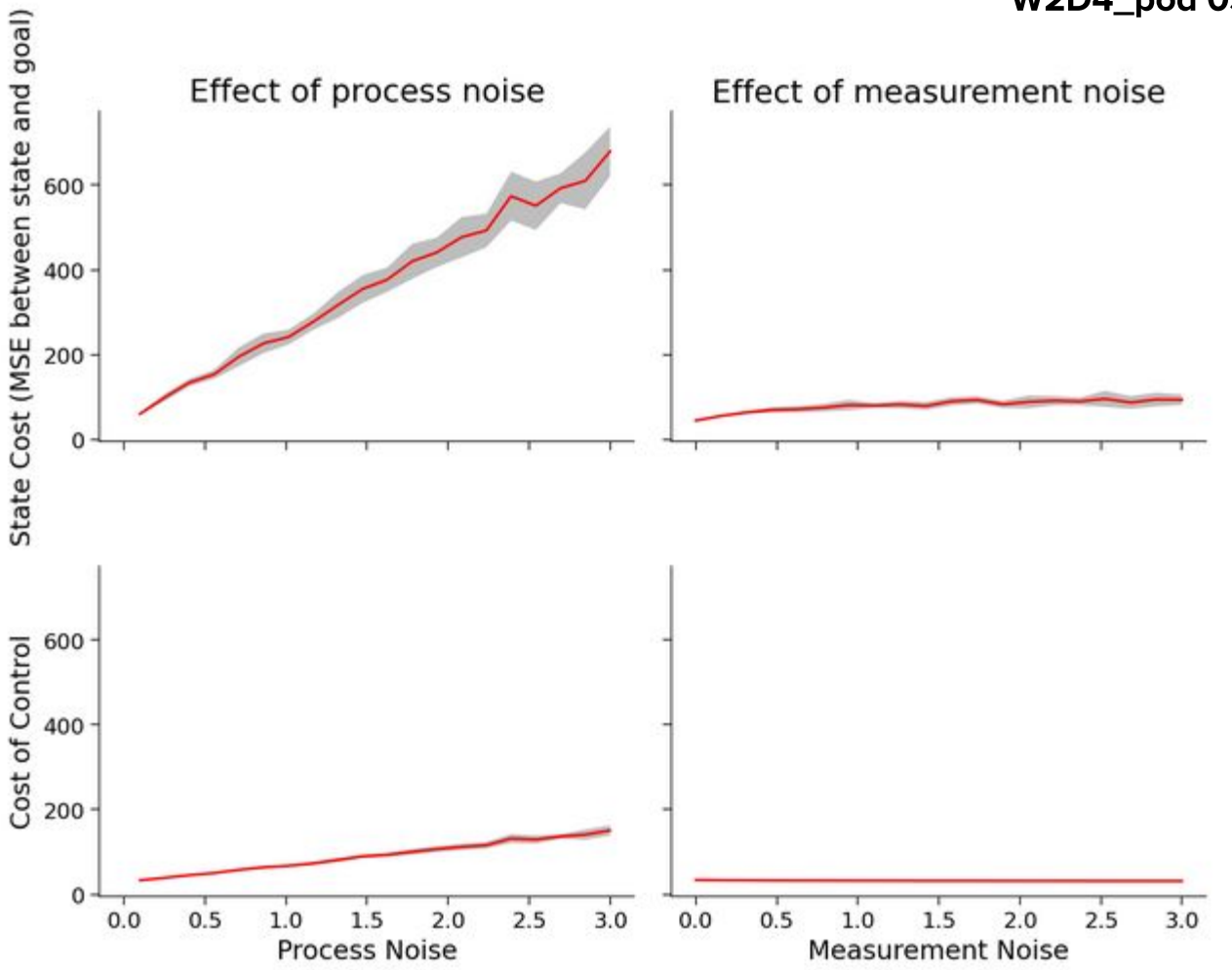

proc_noise
0.00
meas_noise
0.10
Run Simulation
State estimation with KF (LQG controller)
Estimated State
Latent State
State
Time
Latent State
Estimated State
goal

Observations

As process noise increases, it becomes more difficult to keep the state close to the goal g=0, even though we may have very little measurement noise (thus can estimate the state exactly).

On the other hand, as you increase the measurement noise, you will notice that it is harder to estimate the states, and this also may make it harder to keep the state close to the goal.

quantify how the state cost and control costs changes when we change the process and measurement noise levels.



Observations

While both sources of noise have an effect on the controlled state, the process noise has a much larger effect. As the process noise w[t] increases, state cost (MSE between state and goal) and control cost increase drastically.
To make matters worse, as the process noise gets larger, it's harder to keep the system close to the goal.

The measurement noise v[t] also has an effect on the accuracy of the controlled state. As this noise increases, the MSE between the state and goal increases. The cost of control in this case remains fairly constant with increasing levels of measurement noise.

Neurally?

Helps predict movements of mouse!

(Why? Predict goal trajectories)