Dual–SPMA Framework for Convex MDPs

Speaker Script with Pronunciation Guide

Pegah Aryadoost Danielle Nguyen Shervin Khamooshian Ahmed Magd

December 2025

Contents

Speaker Assignments

Speaker	Section	Slides
Pegah	Background & Motivation	1–9
Danielle	Problem Formulation & Fenchel Duality	10–17
Shervin	Related Work & Our Method	18–27
Ahmed	Experiments & Conclusion	28–36

Notation Pronunciation Guide

Pronunciation Guide

Greek Letters:

- π [pie] — policy
- γ [gamma] — discount factor
- ρ [rho] — initial state distribution
- α [alpha] — step size / entropy coefficient
- β [beta] — dual step size
- λ [lambda] — Lagrange multiplier / dual variable
- μ [mu] — penalty weight
- τ [tau] — constraint threshold
- η [eta] — learning rate / step size
- θ [theta] — policy parameters
- ω [omega] — dual parameters
- ϕ [phi] — feature function

Calligraphic Letters:

- $\mathcal{S}$ [script-S / curly-S] — state space
- $\mathcal{A}$ [script-A / curly-A] — action space
- $\mathcal{D}$ [script-D / curly-D] — set of feasible occupancy measures

Functions and Operators:

- f^* [f-star] — Fenchel conjugate of f
- ∇ [nabla / del / gradient] — gradient operator
- $\sup$ [soup / supremum] — least upper bound
- $\inf$ [infimum] — greatest lower bound
- $\mathbb{E}$ [E / expectation] — expected value
- $\langle \cdot, \cdot \rangle$ [inner product / angle brackets] — dot product
- $\sum$ [sum / sigma] — summation
- $\prod$ [product / pi] — product
- $|\mathcal{S}|$ [size of S / cardinality of S] — number of states
- $[\cdot]_+$ [positive part / ReLU] — $\max(0, \cdot)$

Key Terms:

- $d_\pi(s, a)$ [d-pi of s-a] — occupancy measure under policy π
- $J(\pi)$ [J of pi] — expected return of policy π
- $J_r(\pi)$ [J-r of pi] — reward return
- $J_c(\pi)$ [J-c of pi] — cost return
- $V^\pi(s)$ [V-pi of s] — value function
- $Q^\pi(s, a)$ [Q-pi of s-a] — action-value function
- $A^\pi(s, a)$ [A-pi of s-a] — advantage function
- $L(\pi, y)$ [L of pi-y / Lagrangian] — saddle-point objective
- r_y [r-sub-y / r-y] — shaped reward
- $P(s'|s, a)$ [P of s-prime given s-a] — transition probability

Subscripts/Superscripts:

- s_t [s-sub-t / s at time t] — state at time t
- a_t [a-sub-t / a at time t] — action at time t
- π_k [pi-sub-k / pi at iteration k] — policy at iteration k
- y_k [y-sub-k] — dual variable at iteration k
- π^* [pi-star / optimal pi] — optimal policy
- $\hat{d}_\pi$ [d-hat-pi / estimated d-pi] — estimated occupancy

1 Background & Motivation (Speaker: Pegah)

Slide 1 — *A Dual–SPMA Framework for Convex MDPs*

Speaker: Pegah

- “Hi everyone, thanks for being here. I’m Pegah, and this project is joint work with Ahmed, Shervin, and Danielle from SFU’s School of Computing Science.”
- “Our title is ‘A Dual–SPMA Framework for Convex MDPs’.”
- “The one-sentence claim for this talk is: **combining Fenchel duality with a fast policy optimizer, SPMA, gives a simple and fairly competitive way to solve convex MDPs**.”
- “We’ll compare this ‘Dual–SPMA’ recipe against a strong baseline called NPG–PD, which is a natural policy-gradient primal–dual method.”

Slide 2 — *Outline*

Speaker: Pegah

- “Here’s the plan for the next 20–25 minutes.”
- “First, I’ll quickly review basic reinforcement learning background.”
- “Then I’ll motivate **convex MDPs**—why we care about them beyond standard RL.”
- “Danielle will talk about the core math: how we use Fenchel duality to turn a convex MDP into a saddle-point problem.”
- “Shervin will cover related work and explain our method, Dual–SPMA.”
- “Ahmed will present our experiments and results.”
- “Finally we’ll wrap up with takeaways and future work.”

Slide 3 — *What is Reinforcement Learning?*

Speaker: Pegah

- “Let’s start with a quick refresh on RL.”
- “In reinforcement learning, an **agent** interacts with an **environment** repeatedly over time.”
- “At each time step t [t]:
  1. the agent observes a state s_t [s-sub-t],
  2. chooses an action a_t [a-sub-t] according to some policy,
  3. receives a reward r_t [r-sub-t],
  4. and the environment transitions to a new state s_{t+1} [s-sub-t-plus-one].”
- “The goal is to learn a policy that chooses actions to maximize long-term reward, based only on this interaction loop, not on a known model of the dynamics.”

Slide 4 — *Markov Decision Process (MDP): Formal Definition*

Speaker: Pegah

- "Formally, we model the environment as a **Markov Decision Process**, or MDP."
- "An MDP is given by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho)$ [script-S, script-A, P, r, gamma, rho]:"
- "$\mathcal{S}$ [script-S] is the state space; $\mathcal{A}$ [script-A] is the action space."
- "$P(s' \mid s, a)$ [P of s-prime given s, a] is the transition probability: if we're in state s and take action a, this is the distribution over next states s' [s-prime]."
- "$r(s, a)$ [r of s, a] is the reward we get for taking action a in state s."
- "γ [gamma] is a discount factor between 0 and 1; it tells us how much we care about future rewards compared to immediate ones."
- "$\rho(s)$ [rho of s] is the initial state distribution at time zero."
- "A **policy** $\pi(a \mid s)$ [pi of a given s] is just a conditional probability distribution: it tells us how likely we are to choose each action in each state."

Slide 5 — *Trajectory and Return*

Speaker: Pegah

- "If we run a policy in an MDP, we get a **trajectory**: a sequence of states, actions, and rewards over time."
- "You can think of it as: start at s_0 [s-zero], take action a_0 [a-zero], get reward r_0 [r-zero], move to s_1 [s-one], and repeat."
- "The quantity we care about is the **discounted return** of a policy."
- "We write:
  $$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]$$
  [J of pi equals E-sub-pi of the sum from t equals zero to infinity of gamma-to-the-t times r of s-t, a-t]."
- "So the classical RL objective is: find a policy π^* [pi-star] that maximizes this expected discounted return."

Slide 6 — *Occupancy Measure: Where the Policy Spends Time*

Speaker: Pegah

- "Now we introduce a slightly different way to look at a policy, which will be crucial later: the **discounted occupancy measure**."

- "For a policy π [pi], we define:

$$d_\pi(s,a) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \Pr_\pi(s_t = s, a_t = a)$$

[d-sub-pi of s, a equals one-minus-gamma times the sum from t equals zero to infinity of gamma-to-the-t times the probability under pi that s-t equals s and a-t equals a]."
- "Intuitively, $d_\pi(s,a)$ [d-pi of s, a] is the **normalized discounted frequency** of visiting state–action pair (s,a) under π [pi]."
- "The factor $(1-\gamma)$ [one-minus-gamma] just normalizes things so that $\sum_{s,a} d_\pi(s,a) = 1$ [the sum over all s, a of d-pi equals one]; so d_π [d-pi] is actually a probability distribution over state–action pairs."
- "You can think of d_π [d-pi] as a heatmap over the grid of states and actions: bright cells are visited often; dark cells are rarely visited."

Slide 7 — *Key Identity: RL is Linear in Occupancy*

Speaker: Pegah

- "The reason occupancy measures are so nice is this **fundamental identity**."
- "If we take the inner product between rewards and occupancy, we get:

$$\langle r, d_\pi\rangle = \sum_{s,a} r(s,a)\, d_\pi(s,a)$$

[angle-bracket r, d-pi angle-bracket equals the sum over s, a of r of s, a times d-pi of s, a]."
- "Using the definition of d_π [d-pi], this equals $(1-\gamma)J(\pi)$ [one-minus-gamma times J of pi]."
- "So, **up to the constant factor** $(1-\gamma)$ [one-minus-gamma], the RL objective $J(\pi)$ [J of pi] is simply a linear function of the occupancy measure d_π [d-pi]."
- "This means standard RL is basically: choose a policy whose occupancy puts more weight on high-reward state–action pairs."
- "However, many interesting objectives are **not** linear in d_π [d-pi]:"
  - "Safety constraints like $\langle c, d_\pi\rangle \le \tau$ [angle-bracket c, d-pi angle-bracket less-than-or-equal-to tau],"
  - "Imitation learning where we try to match an expert occupancy,"
  - "Exploration bonuses like reward plus entropy of d_π [d-pi]."
- "That's where **convex MDPs** come in."

Slide 8 — *Why Convex MDPs?*

Speaker: Pegah

- "Linear RL is great if all we care about is maximizing reward."

- "But in practice we often care about more structured objectives:"
  - "We might want to **enforce safety** constraints."
  - "We might want to **match expert behaviour** in imitation learning."
  - "We might want to **encourage exploration** or control risk."
- "Convex MDPs give a clean way to express these goals."
- "The high-level idea is: instead of maximizing a linear function of d_π [d-pi], we want to **minimize a convex function** $f(d_\pi)$ [f of d-pi]."
- "So the problem becomes: $\min_\pi f(d_\pi)$ [min over pi of f of d-pi], where f [f] could include rewards, costs, entropy, and so on."
- "The challenge is that optimizing over occupancy measures is hard: the dimension is $|\mathcal{S}||\mathcal{A}|$ [size-of-S times size-of-A], and the valid occupancies lie in a complicated constrained polytope."
- "We can't just feed this to a standard RL algorithm."
- "Our approach will be to use **Fenchel duality** to turn this into a **min–max game** that looks like ordinary shaped-reward RL."

Slide 9 — *Convex MDP: Examples*

Speaker: Pegah

- "Here are a few examples of convex MDP objectives, all expressed as functions of d [d]."
- "First, standard RL is actually a very simple special case:
  $$f(d) = -\langle r, d\rangle$$
  [f of d equals negative angle-bracket r, d angle-bracket]. This is linear, hence convex."
- "Second, **entropy-regularized RL**:
  $$f(d) = -\langle r, d\rangle + \alpha \sum_{s,a} d(s,a)\log d(s,a)$$
  [f of d equals negative angle-bracket r, d angle-bracket plus alpha times the sum over s, a of d of s, a times log of d of s, a]."
  - "The extra term encourages spread-out occupancies—softer, more exploratory policies."
- "Third, a **constrained safety** objective:
  $$f(d) = -\langle r, d\rangle + \mu \max\{0, \langle c, d\rangle - \tau\}$$
  [f of d equals negative angle-bracket r, d angle-bracket plus mu times max of zero and angle-bracket c, d angle-bracket minus tau]."
  - "Here c [c] is a cost function, τ [tau] is a safety threshold, and μ [mu] is a fixed penalty weight."
- "All three share the same structure: they're **convex functions of the occupancy measure**."
- "Next, Danielle will show how we apply **Fenchel duality** to these convex objectives and turn the whole problem into a saddle-point formulation."

2 Problem Formulation & Fenchel Duality (Speaker: Danielle)

Slide 10 — *Roadmap (checkpoint)*

Speaker: Danielle

- “Thanks, Pegah. So far we’ve set up the RL basics and motivated convex MDPs.”
- “Next, I’ll show how we apply **Fenchel duality** to these convex objectives and turn the whole problem into a saddle-point formulation.”

Slide 11 — *Fenchel Conjugate: Definition*

Speaker: Danielle

- “We start with the concept of the **Fenchel conjugate** of a convex function.”
- “Given a convex function $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ [f from R-n to R union infinity], its conjugate is defined as:
$$f^*(y) = \sup_{x \in \mathbb{R}^n} \{\langle y, x\rangle - f(x)\}$$
[f-star of y equals the supremum over x in R-n of angle-bracket y, x angle-bracket minus f of x].”
- “Here, y [y] is a dual variable in the same dimension, $\langle y, x\rangle$ [angle-bracket y, x angle-bracket] is the usual dot product, and ‘sup’ [soup / supremum] is the supremum—the least upper bound.”
- “Intuitively, $f^*(y)$ [f-star of y] measures how large the linear function $\langle y, x\rangle$ [angle-bracket y, x angle-bracket] can be relative to $f(x)$ [f of x].”
- “We’ll use this conjugate to rewrite our convex objective in a way that exposes a min-max structure.”

Slide 12 — *Fenchel–Moreau Theorem*

Speaker: Danielle

- “The key result we rely on is the **Fenchel–Moreau theorem**.”
- “It says that for any proper, closed, convex function f [f], you can recover f from its conjugate via:
$$f(d) = \sup_{y}\{\langle y, d\rangle - f^*(y)\}$$
[f of d equals the supremum over y of angle-bracket y, d angle-bracket minus f-star of y].”
- “So f is literally the conjugate of its own conjugate: $f = f^{**}$ [f equals f-double-star].”
- “Why is this useful for us?”
- “Because it allows us to turn **minimizing** $f(d)$ [f of d] into a **min–max optimization** over d [d] and y [y], where y lives in the dual space.”

Slide 13 — *Applying Fenchel Duality to Convex MDPs*

Speaker: Danielle

- "Let's apply this to our convex MDP."
- "We start with the problem: $\min_{d\in\mathcal{D}} f(d)$ [min over d in script-D of f of d], where $\mathcal{D}$ [script-D] is the set of feasible occupancy measures."
- "Using the Fenchel–Moreau identity, we can write:

$$f(d) = \sup_y\{\langle y, d\rangle - f^*(y)\}$$

[f of d equals sup over y of angle-bracket y, d angle-bracket minus f-star of y]."
- "Plugging that in, we get:

$$\min_{d\in\mathcal{D}} f(d) = \min_{d\in\mathcal{D}} \sup_y\{\langle y, d\rangle - f^*(y)\}$$

[min over d in script-D of f of d equals min over d in script-D sup over y of angle-bracket y, d angle-bracket minus f-star of y]."
- "This gives us a **convex-concave saddle-point problem**:

$$\min_{d\in\mathcal{D}} \max_y\{\langle y, d\rangle - f^*(y)\}$$

[min over d in script-D max over y of angle-bracket y, d angle-bracket minus f-star of y]."
- "Under standard convexity and closedness assumptions, solving this saddle-point is equivalent to solving the original convex MDP."
- "Finally, we replace d by the occupancy induced by a policy, d_π [d-pi], and obtain the saddle-point formulation we'll actually work with:

$$\min_\pi \max_y L(\pi, y), \quad \text{where } L(\pi, y) = \langle y, d_\pi\rangle - f^*(y)$$

[min over pi max over y of L of pi, y, where L of pi, y equals angle-bracket y, d-pi angle-bracket minus f-star of y]."

Slide 14 — *Two-Player Game Interpretation*

Speaker: Danielle

- "This saddle-point problem can be viewed as a **two-player game**."
- "The objective is:

$$\min_\pi \max_y L(\pi, y), \quad \text{with } L(\pi, y) = \langle y, d_\pi\rangle - f^*(y)$$

[min over pi max over y of L of pi, y, with L of pi, y equals angle-bracket y, d-pi angle-bracket minus f-star of y]."
- "We interpret this as:"

  - "The **policy player** chooses a policy π [pi] and wants to **minimize** L [L]. This is our RL agent."
  - "The **dual player** chooses a dual variable y [y] and wants to **maximize** L [L]. This player shapes the reward signal."
- "At an equilibrium—or saddle point—neither player can unilaterally improve: the policy player has found an optimal policy π^* [pi-star], and the dual player has found an optimal certificate y^* [y-star]."
- "This is exactly what we mean by 'solving the min–max problem'."

Slide 15 — *Visualizing the Saddle Point*

Speaker: Danielle

- "To build some geometric intuition, here's a simple 3D example."
- "The surface shows a function $L(d, y)$ [L of d, y] of two variables: think of d [d] on the horizontal axis as the policy-side variable, and y [y] on the other axis as the dual variable."
- "The **red point** is a saddle point."
- "If we move along the d-direction [d-direction], keeping y [y] fixed, the red point is a **minimum** for the policy player."
- "If we move along the y-direction [y-direction], keeping d [d] fixed, the red point is a **maximum** for the dual player."
- "So that point is simultaneously 'best response' for both players, and that's why it's the solution of the min–max game."

Slide 16 — *From Saddle Point to Shaped Reward*

Speaker: Danielle

- "Now, how do we actually minimize over policies in this saddle-point objective?"
- "For a fixed dual variable y [y], the policy player's subproblem is:

$$\min_{\pi} \langle y, d_\pi \rangle$$

[min over pi of angle-bracket y, d-pi angle-bracket]."
- "Using the occupancy definition, we can rewrite this inner product as:

$$\langle y, d_\pi \rangle = (1-\gamma)\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t y(s_t, a_t) \right]$$

[angle-bracket y, d-pi angle-bracket equals one-minus-gamma times E-sub-pi of the sum from t equals zero to infinity of gamma-to-the-t times y of s-t, a-t]."

- "So minimizing $\langle y, d_\pi \rangle$ [angle-bracket y, d-pi angle-bracket] is equivalent to **maximizing** expected return with a **shaped reward**:

  $$r_y(s, a) = -y(s, a)$$

  [r-sub-y of s, a equals negative y of s, a]."
- "In other words, from the policy's perspective, the dual variable y [y] just defines a new reward function."
- "This is our main operational insight: the policy player can use **standard RL techniques** on a shaped reward $r_y = -y$ [r-sub-y equals negative y]."

Slide 17 — *Summary: The Fenchel Dual Reduction*

Speaker: Danielle

- "Let me summarize the reduction so far."
- "We started with a convex MDP: $\min_\pi f(d_\pi)$ [min over pi of f of d-pi]."
- "Using Fenchel duality, we rewrote it as a saddle-point problem:

  $$\min_\pi \max_y L(\pi, y)$$

  [min over pi max over y of L of pi, y]."
- "By fixing y [y], we observed that minimizing over π [pi] is equivalent to RL with a shaped reward $r_y = -y$ [r-sub-y equals negative y]."
- "This leads to a very simple algorithmic structure:"
  1. "A **dual step**: update y [y] using the gradient of L [L] with respect to y [y]:

     $$y_{k+1} = y_k + \alpha(\hat{d}_{\pi_k} - \nabla f^*(y_k))$$

     [y-sub-k-plus-one equals y-sub-k plus alpha times open-paren d-hat-sub-pi-k minus nabla f-star of y-k close-paren]."
  2. "A **policy step**: run an RL algorithm on the shaped reward $r_{y_k} = -y_k$ [r-sub-y-k equals negative y-k] to get the next policy π_{k+1} [pi-sub-k-plus-one]."
- "So we've reduced the convex MDP to **alternating between convex optimization in the dual variable and standard RL in the policy**."

3 Related Work & Our Method (Speaker: Shervin)

Slide 18 — *Roadmap (checkpoint)*

Speaker: Shervin

- "We've now completed the core mathematical formulation using Fenchel duality."
- "Next, I'll talk about the related work that inspired this framework and the specific policy optimizer we use, SPMA."

Slide 19 — *Related Work: "Reward Is Enough"*

Speaker: Shervin

- "The first key paper is '*Reward Is Enough for Convex MDPs*' by Zahavy and co-authors."
- "This is the foundational work that introduced the convex-MDP perspective we just summarized."
- "They made three main contributions:"
  1. "They gave the **Fenchel dual reduction** from convex MDPs to a saddle-point problem—the exact reduction we use."
  2. "They proposed a **meta-algorithm**: alternate between updating the policy and updating the dual variable. Any RL algorithm can play the policy role; any online convex optimization algorithm can play the dual role."
  3. "They showed that many existing RL paradigms—imitation learning, constrained RL, entropy-regularized RL—are special cases of convex MDPs."
- "Our contribution is to **implement this framework concretely using SPMA** as the policy optimizer, and to compare it against a strong primal–dual baseline."

Slide 20 — *Related Work: Softmax Policy Mirror Ascent (SPMA)*

Speaker: Shervin

- "The second key ingredient is **Softmax Policy Mirror Ascent**, or SPMA, from a recent paper by Asad et al."
- "SPMA is a policy optimization algorithm that performs **mirror ascent in logit space** using the log-sum-exp mirror map."
- "In tabular MDPs, they show that SPMA achieves **linear convergence**, which is faster than the usual sublinear rates of vanilla policy gradient."
- "The tabular update rule has the form:

  $$\pi_{t+1}(a \mid s) = \pi_t(a \mid s)\,[1 + \eta A_{\pi_t}(s, a)]$$

  [pi-sub-t-plus-one of a given s equals pi-sub-t of a given s times open-bracket one plus eta times A-sub-pi-t of s, a close-bracket],
  where $A_{\pi_t}(s, a)$ [A-sub-pi-t of s, a] is the advantage function."

- "In practice, we use a numerically stable approximation of this update."
- "SPMA has a few attractive properties:"
  - "It's **geometry-aware**: the updates are designed to respect the probability simplex."
  - "There is no explicit per-state normalization step like in some other methods."
  - "And the paper provides a clean extension to function approximation as a convex classification problem."
- "This makes SPMA a good candidate for the policy player in our convex-MDP framework."

Slide 21 — *Related Work: NPG–PD (Our Baseline)*

Speaker: Shervin

- "Our main baseline is the **Natural Policy Gradient Primal–Dual** method, or NPG–PD, from Ding et al. 2020."
- "NPG–PD is designed for constrained MDPs with a Lagrangian of the form:

$$L(\pi, \lambda) = J_r(\pi) + \lambda(J_c(\pi) - \tau), \quad \text{with } \lambda \geq 0$$

[L of pi, lambda equals J-r of pi plus lambda times open-paren J-c of pi minus tau close-paren, with lambda greater-than-or-equal-to zero]."
- "On the **primal side**, it performs **natural policy gradient ascent** on the policy, using the Fisher information matrix as a geometry."
- "On the **dual side**, it updates the Lagrange multiplier using:

$$\lambda_{k+1} = [\lambda_k + \beta(J_c(\pi_k) - \tau)]_+$$

[lambda-sub-k-plus-one equals the positive part of lambda-sub-k plus beta times open-paren J-c of pi-k minus tau close-paren],
which is projected gradient ascent on the constraint violation."
- "NPG–PD comes with theoretical guarantees: they show an $\mathcal{O}(1/\sqrt{T})$ [O of one over square-root-T] bound on optimality gap and constraint violation."
- "Because it's both principled and relatively strong empirically, we use NPG–PD as our **comparison point** for Dual–SPMA."

Slide 22 — *Roadmap (checkpoint)*

Speaker: Shervin

- "So far, we've seen the convex-MDP theory and the two main building blocks: the Fenchel dual framework from Zahavy et al., and the SPMA policy optimizer."
- "Next I'll describe what we actually built: our **Dual–SPMA method**, occupancy estimators, and the NPG–PD baseline."

Slide 23 — *Our Contributions*

Speaker: Shervin

- “Here is a summary of what we contributed in this project.”
- “First, we implemented a **complete Dual–SPMA framework**: the outer dual loop plus an SPMA-based policy oracle.”
  - “Our implementation supports both entropy-regularized RL and constrained safety CMDPs.”
- “Second, we developed **three different occupancy estimators**:”
  - “a simple tabular Monte Carlo estimator,”
  - “a feature-based Monte Carlo estimator,”
  - “and an MLE-style estimator inspired by recent work by Barakat et al.”
- “Third, we implemented a **faithful NPG–PD baseline** so that we could compare on equal footing.”
- “Finally, we performed an **empirical comparison** of SPMA vs NPG–PD on constrained safety tasks, focusing on convergence and constraint satisfaction.”

Slide 24 — *Dual–SPMA Loop: High-Level View*

Speaker: Shervin

- “Here’s the high-level structure of our Dual–SPMA algorithm.”
- “Recall the saddle-point objective:

  $$L(\pi, y) = \langle y, d_\pi \rangle - f^*(y)$$

  [L of pi, y equals angle-bracket y, d-pi angle-bracket minus f-star of y].”
- “The **outer loop** updates the dual variable:

  $$y_{k+1} = y_k + \alpha(\hat{d}_{\pi_k} - \nabla f^*(y_k))$$

  [y-sub-k-plus-one equals y-sub-k plus alpha times open-paren d-hat-sub-pi-k minus nabla f-star of y-k close-paren].
  We use an estimated occupancy $\hat{d}_{\pi_k}$ [d-hat-sub-pi-k] from rollouts.”
- “The **inner loop** updates the policy: we run K_{inner} [K-inner] SPMA steps using the shaped reward $r_y = -y$ [r-sub-y equals negative y].”
- “Algorithmically, each outer iteration looks like this:”
  1. “Run SPMA steps on the current shaped reward to improve the policy.”
  2. “Estimate the occupancy of the new policy via Monte Carlo.”
  3. “Update the dual variable using this occupancy.”
- “After a fixed number of outer iterations, we return the final policy π_K [pi-sub-K].”

Slide 25 — *Policy Player: SPMA Actor Loss*

Speaker: Shervin

- "In practice, our SPMA policy update is implemented via a custom **actor loss**."
- "Standard policy gradient uses the loss:

$$L_{\mathrm{PG}} = -\mathbb{E}[\log \pi(a \mid s)\, A(s,a)]$$

[L-sub-PG equals negative E of log pi of a given s times A of s, a]."
- "SPMA adds a 'stay close to the old policy' regularizer in logit space."
- "Our SPMA loss is:

$$L_{\mathrm{SPMA}} = \mathbb{E}\left[-\Delta \log \pi \cdot A + \frac{1}{\eta}\Big(\exp(\Delta \log \pi) - 1 - \Delta \log \pi\Big)\right]$$

[L-sub-SPMA equals E of negative delta-log-pi times A plus one-over-eta times open-paren exp of delta-log-pi minus one minus delta-log-pi close-paren]."
- "Here $\Delta \log \pi = \log \pi_{\mathrm{new}}(a \mid s) - \log \pi_{\mathrm{old}}(a \mid s)$ [delta-log-pi equals log-pi-new of a given s minus log-pi-old of a given s] is the change in log-probability, and η [eta] is a step-size parameter we select via Armijo line search."
- "The exponential term is a kind of KL-like regularizer that penalizes large changes in log-probabilities, giving us a stable, geometry-aware update on the simplex."

Slide 26 — *Occupancy Estimation: MC vs MLE*

Speaker: Shervin

- "A critical ingredient in the dual update is the occupancy estimate $\hat{d}_\pi$ [d-hat-sub-pi]."
- "Our **default** is a Monte Carlo estimator in the tabular case:

$$\hat{d}_\pi(s,a) = \frac{1-\gamma}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \gamma^t \mathbf{1}\{s_t^{(i)} = s,\, a_t^{(i)} = a\}$$

[d-hat-sub-pi of s, a equals one-minus-gamma over N times the sum over i from 1 to N of the sum over t from 0 to T of gamma-to-the-t times the indicator that s-t-i equals s and a-t-i equals a]."
- "This is simple: we just count discounted visits across trajectories."
- "The downside is that its variance grows with the number of state–action pairs $|\mathcal{S}||\mathcal{A}|$ [size-of-S times size-of-A]."
- "To address this, we also explored an MLE-based estimator inspired by Barakat et al."
- "We parametrize a log-linear distribution:

$$\lambda_\omega(s,a) \propto \exp(\omega^\top \phi(s,a))$$

[lambda-sub-omega of s, a is proportional to exp of omega-transpose phi of s, a], and fit ω [omega] by maximizing likelihood on our samples."

- "This estimator has the nice property that its error depends on the feature dimension, not directly on $|\mathcal{S}||\mathcal{A}|$ [size-of-S times size-of-A]."
- "As a sanity check, in all our experiments we verified that $\sum_{s,a} \hat{d}_\pi(s,a)$ [sum over s, a of d-hat-pi of s, a] stayed close to 1."

Slide 27 — *Example: Constrained Safety CMDP*

Speaker: Shervin

- "Let me show how a constrained MDP fits our framework."
- "The problem is: maximize reward $J_r(\pi)$ [J-r of pi] subject to $J_c(\pi) \le \tau$ [J-c of pi less-than-or-equal-to tau], where:

$$J_r(\pi) = \mathbb{E}_\pi\left[\sum_t \gamma^t r(s_t, a_t)\right], \quad J_c(\pi) = \mathbb{E}_\pi\left[\sum_t \gamma^t c(s_t, a_t)\right]$$

  [J-r of pi equals E-sub-pi of the sum over t of gamma-to-the-t times r of s-t, a-t; J-c of pi equals E-sub-pi of the sum over t of gamma-to-the-t times c of s-t, a-t]."
- "In Dual–SPMA:"
  1. "Build dual variable: $y_\lambda(s,a) = \lambda c(s,a) - r(s,a)$ [y-sub-lambda of s, a equals lambda times c of s, a minus r of s, a]."
  2. "Policy sees shaped reward: $r_y = -y = r - \lambda c$ [r-sub-y equals negative y equals r minus lambda c]."
  3. "Run SPMA inner loop on r_y [r-sub-y]."
  4. "Update dual: $\lambda_{k+1} = [\lambda_k + \beta(J_c(\pi_k) - \tau)]_+$ [lambda-sub-k-plus-one equals the positive part of lambda-sub-k plus beta times J-c of pi-k minus tau]."
- "SPMA and NPG–PD have the **same dual update**—the only difference is Step 3: SPMA loss vs. natural gradient."
- "Now Ahmed will present our experimental results."

4 Experiments & Conclusion (Speaker: Ahmed)

Slide 28 — *Example: Constrained Safety CMDP*

Speaker: Ahmed

- “Let me show how this looks concretely on a constrained safety problem.”
- “The CMDP objective is: maximize reward subject to a cost constraint:

$$\max_{\pi} J_r(\pi) \quad \text{such that} \quad J_c(\pi) \le \tau$$

[max over pi of J-r of pi, such that J-c of pi is less-than-or-equal-to tau].”
- “Here J_r [J-r] and J_c [J-c] are discounted returns of reward and cost respectively.”
- “In the Dual–SPMA view, we build a dual variable:

$$y_\lambda(s,a) = \lambda c(s,a) - r(s,a)$$

[y-sub-lambda of s, a equals lambda times c of s, a minus r of s, a].”
- “The policy sees the shaped reward:

$$r_y = -y_\lambda = r(s,a) - \lambda c(s,a)$$

[r-sub-y equals negative y-lambda equals r of s, a minus lambda times c of s, a].”
- “We then run the SPMA inner loop on this shaped reward to update the policy.”
- “Finally, we update the dual variable:

$$\lambda_{k+1} = [\lambda_k + \beta(J_c(\pi_k) - \tau)]_+$$

[lambda-sub-k-plus-one equals the positive part of lambda-sub-k plus beta times open-paren J-c of pi-k minus tau close-paren].”
- “Notice that this dual update is the same as in NPG–PD; **the main difference is the policy optimizer**: SPMA vs natural gradient.”

Slide 29 — *Roadmap (checkpoint)*

Speaker: Ahmed

- “We’ve now described the Dual–SPMA method, the occupancy estimators, and the NPG–PD baseline.”
- “Next, I’ll show you our **experimental setup and results** on a simple constrained safety task.”

Slide 30 — *Experimental Setup*

Speaker: Ahmed

- "Our experiments focus on a tabular setting where we can directly visualize occupancies."
- "The environment is the classic **FrozenLake 4×4** gridworld."
- "We use deterministic transitions for simplicity."
- "Certain states—holes in the lake—are marked as **unsafe**; taking an action in those states incurs cost $c = 1$ [c equals one]."
- "We compare two methods: our **Dual–SPMA** and the **NPG–PD** baseline."
- "We track several metrics over outer iterations:"
  - "$J_r(\pi)$ [J-r of pi]: reward return,"
  - "$J_c(\pi)$ [J-c of pi]: cost return,"
  - "$J_c - \tau$ [J-c minus tau]: constraint violation,"
  - "and the sum of the occupancy estimate, to check that $\sum_{s,a} \hat{d}_\pi(s,a) \approx 1$ [sum over s, a of d-hat-pi of s, a is approximately one]."
- "We set the discount factor to $\gamma = 0.99$ [gamma equals zero-point-nine-nine], the safety threshold to $\tau = 0.1$ [tau equals zero-point-one], and run 30 outer iterations with 2048 environment steps per rollout."

Slide 31 — *Results: Entropy-Regularized RL*

Speaker: Ahmed

- "First, we tested on entropy-regularized RL—no constraints, just an entropy bonus."
- "The left plot shows L [L], the Lagrangian objective, versus iteration. Both methods decrease it, confirming the dual loop is working."
- "The right plot shows the occupancy sum $\sum_{s,a} \hat{d}_\pi(s,a)$ [sum over s, a of d-hat-pi of s, a]. It stays close to 1, which is a sanity check on our estimator."
- "Both SPMA and NPG–PD behave similarly here; they're both optimizing the same entropy-regularized objective."

Slide 32 — *Results: Constrained Safety CMDP*

Speaker: Ahmed

- "Next, we tested on the constrained safety CMDP—maximize reward while staying below the cost threshold."
- "The left plot shows the reward return $J_r(\pi_k)$ [J-r of pi-k] over iterations. Both methods increase reward as the policy improves."

- "The right plot shows the constraint violation $J_c(\pi_k) - \tau$ [J-c of pi-k minus tau]. Both methods eventually bring this below zero, meaning the constraint is satisfied."
- "We see that Dual–SPMA and NPG–PD reach similar final performance, though their learning curves differ slightly."

Slide 33 — *Results: Dual–SPMA vs NPG–PD*

Speaker: Ahmed

- "Now let's compare Dual–SPMA against NPG–PD on the same constrained task."
- "The left plot shows the reward return $J_r(\pi_k)$ [J-r of pi-k] versus outer iterations for both methods."
- "We find that both ultimately achieve similar reward once the constraint is satisfied."
- "The right plot shows the constraint violation $J_c(\pi_k) - \tau$ [J-c of pi-k minus tau] for each method."
- "Both methods eventually bring the violation close to zero, but they behave a bit differently."
- "Empirically, we find that **SPMA** tends to take larger, geometry-aware steps—sometimes converging faster but being slightly more sensitive to hyperparameters."
- "NPG–PD is often smoother but requires careful tuning of the step size β [beta]."
- "Overall, Dual–SPMA is competitive with NPG–PD on these tasks, which supports our claim that SPMA is a viable policy optimizer in the convex-MDP framework."

Slide 34 — *Results: Occupancy Heatmaps*

Speaker: Ahmed

- "To make the policies more interpretable, we visualize their occupancies as heatmaps over the grid."
- "On the **left**, we show the occupancy of an **unconstrained** policy trained only for reward."
- "You can see that it explores the map more broadly, including states that correspond to holes or unsafe positions."
- "On the **right**, we show the occupancy of a **safety-constrained** policy from Dual–SPMA."
- "This policy clearly avoids unsafe states: their occupancy is very low or zero."
- "These pictures confirm that the constraint is not just satisfied numerically, but also reflected in the agent's behaviour."

Slide 35 — *Takeaways & Limitations*

Speaker: Ahmed

- "Let me summarize the main takeaways and limitations."
- "Our high-level **recipe** is:

  $$\text{Convex MDP} \Rightarrow \text{Fenchel dual saddle-point game} \Rightarrow \text{SPMA-based shaped-reward RL}$$

  "
- "On top of that, we built:"
  - "Dual–SPMA loops for entropy-regularized RL and constrained safety,"
  - "a strong NPG–PD baseline,"
  - "and several occupancy estimators."
- "In our experiments, Dual–SPMA is stable, learns to satisfy constraints, and performs competitively with NPG–PD on a tabular CMDP."
- "In terms of **limitations**:"
  - "All experiments are in low-dimensional tabular settings; we haven't pushed to large continuous environments yet."
  - "Our MLE estimator still needs more stress-testing on high-dimensional tasks."
  - "And we haven't fully explored hyperparameter sensitivity, especially for SPMA's step sizes."

Slide 36 — *Future Work*

Speaker: Ahmed

- "There are several interesting directions for future work."
- "First, **scaling up**: we'd like to test Dual–SPMA on larger CMDPs and safety benchmarks, such as MuJoCo tasks with continuous states and actions."
- "Second, **better occupancy estimation**: in particular, evaluating the MLE estimator more thoroughly in high-dimensional settings and comparing its variance to Monte Carlo."
- "Third, exploring **more convex objectives** beyond safety and entropy: for example, risk-sensitive criteria or imitation learning objectives defined as convex functions of occupancy."
- "Finally, on the theory side, deriving **convergence rates** and **sample complexity bounds** for Dual–SPMA, and comparing them directly to methods like NPG–PD."

Slide 37 — *Q&A*

Speaker: Whoever is fielding questions

- “Thanks for listening—that concludes our presentation.”
- “We’re happy to take questions.”
- (If needed, you can remind the audience: “Pegah covers background and motivation; Danielle covers the Fenchel dual formulation; Shervin covers related work and the method; Ahmed covers experiments and conclusions.”)

Slide 38 — *References*

Speaker: Only if someone asks for a specific citation

- “These are the main references we relied on: Zahavy et al. for convex MDPs, Asad et al. for SPMA, Ding et al. for NPG–PD, and Barakat et al. for the MLE-style occupancy estimator.”

5 Backup Slides

Backup Slide 39 — Flow Constraints (Occupancy Polytope)

Use only if asked "what is $\mathcal{D}$?" or "how do you define valid occupancy measures?"

- "This slide shows the **Bellman flow constraints** that define the occupancy set $\mathcal{D}$ [script-D]."
- "For each state s [s], the total occupancy flowing into s must equal the initial distribution plus the discounted flow from other states."
- "This defines a convex polytope in $\mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ [R to the size-of-S times size-of-A]."
- "So $\mathcal{D}$ [script-D] is convex, which is what we need for the convex MDP formulation."

Backup Slide 40 — Entropy-Regularized Conjugate

Use if someone asks "where does $f^*(y)$ come from in the entropy case?"

- "Here we show the explicit conjugate for the entropy-regularized objective."
- "If $f(d) = -\langle r, d\rangle + \alpha \sum d \log d$ [f of d equals negative angle-bracket r, d angle-bracket plus alpha times sum of d log d], then the conjugate is:

$$f^*(y) = \alpha \log \sum_{s,a} \exp\left(\frac{y(s,a) + r(s,a)}{\alpha}\right)$$

  [f-star of y equals alpha times log of the sum over s, a of exp of open-paren y of s, a plus r of s, a close-paren over alpha]."
- "The gradient of this conjugate is simply a softmax over $y+r$ [y plus r] scaled by α [alpha]."
- "This is very convenient because it makes $\nabla f^*(y)$ [nabla f-star of y] easy to compute in the dual update."

Backup Slide 41 — SPMA with Function Approximation

Use if asked "how does SPMA work with neural networks?"

- "In function approximation, SPMA can be interpreted as solving a **convex KL minimization** problem in the policy space."
- "Given an intermediate policy $\pi_{t+1/2}$ [pi-sub-t-plus-one-half], the next parameters θ [theta] minimize a weighted KL divergence between $\pi_{t+1/2}$ [pi-sub-t-plus-one-half] and π_θ [pi-sub-theta], with weights given by state occupancies."
- "This is equivalent to softmax classification with states as inputs and actions as labels."
- "So SPMA extends naturally to neural policies as a sequence of convex classification problems."

Backup Slide 42 — Algorithm Pseudocode

Use if someone asks for the precise algorithm

- "This is the full pseudocode of Dual–SPMA."
- "We initialize a dual variable y_1 [y-sub-one] and a policy π_1 [pi-sub-one]."
- "For each outer iteration k [k]:"
  1. "We run K_{inner} [K-inner] SPMA updates on shaped reward r_{y_k} [r-sub-y-k] to generate π_{k+1} [pi-sub-k-plus-one]."
  2. "We estimate the occupancy of π_{k+1} [pi-sub-k-plus-one] from trajectories."
  3. "We update the dual variable using the gradient step:

     $$y_{k+1} = y_k + \alpha(\hat{d}_{\pi_{k+1}} - \nabla f^*(y_k))$$

     [y-sub-k-plus-one equals y-sub-k plus alpha times open-paren d-hat-sub-pi-k-plus-one minus nabla f-star of y-k close-paren]."
- "After K [K] outer iterations we return the final policy."

Backup Slide 43 — Notation Summary

Use if someone is confused about symbols

- "This slide is just a notation table."
- "We list the state and action sets, policy $\pi(a \mid s)$ [pi of a given s], reward r [r], cost c [c], discount γ [gamma], occupancy $d_\pi(s, a)$ [d-pi of s, a], the return $J(\pi)$ [J of pi], the convex objective f [f], its conjugate f^* [f-star], the dual variable y [y], the CMDP multiplier λ [lambda], the safety threshold τ [tau], and the step sizes η [eta] and α [alpha]."
- "It's here purely as a reference if any symbol in the earlier slides is unclear."