BRIDGING REWARD REGRESSION AND PROJECTED BELLMAN EQUATION WITH SUCCESSOR FEATURES

Anonymous authors
Paper under double-blind review

ABSTRACT

Successor features (SFs) decompose the Q-function into a successor feature map and a task vector, enabling efficient transfer in continual and goal-conditioned reinforcement learning. The Q-function estimate depends critically on how the task vector is estimated. We formalize the two estimators used in the literature—reward regression and the projected Bellman equation—prove that they generally yield different solutions, and characterize when they coincide. We unify them through an interpolation parameter and establish a stability–approximation tradeoff: reward regression stabilizes off-policy updates, while moving toward the solution of projected Bellman equation tightens the worst-case approximation guarantee. Building on this, we develop *SF-Bellman*, a scalable algorithm that improves performance and transfer speed over reward regression based SF baselines and standard baseline algorithms in continual-learning and goal-conditioned control tasks.

1 INTRODUCTION

Successor features (SFs) provide a structured approach to Q-function estimation in reinforcement learning (RL) (Barreto et al., 2017). Extending the successor representation of Dayan (1993), SFs capture the expected discounted sum of features under a policy, separating the dynamics of the environment from the reward. Leveraging this structure, SFs enable efficient transfer across tasks, and have proven to be effective in settings ranging from continual learning to goal-conditioned tasks (Ma et al., 2020; McLeod et al., 2021).

In this framework, the Q-function is represented as an inner product between the SF and a coefficient vector, referred to as the *task vector* in the SF literature. The quality of the resulting Q-function depends on how the task vector is estimated. SF methods often estimate the task vector by regressing the rewards (Barreto et al., 2017). However, minimizing reward-prediction error does not in general minimize the Q-function approximation error; therefore, the method does not scale to complex sparse-reward and online learning scenarios where most transitions carry zero reward (Chua et al., 2024). Other methods instead fit the task vector through a Bellman-style objective (Ma et al., 2020), which avoids this limitation but can incur instability under off-policy updates (Sutton et al., 2008).

The differing approximation and stability properties of these approaches motivate a unified analysis of task vector estimation with SFs. We formulate two Q-function estimators based on SFs: one derived from the reward regression which we cast as a projected reward equation (PRE), and one derived from the projected Bellman equation (PBE) of classical linear function approximation (Sutton et al., 2008). In particular, **i)** We show that the two estimators are generally different. Moreover, we characterize a necessary and sufficient condition when the estimators can be equivalent; **ii)** We provide a unified view of both equations through defining an interpolation matrix between the feature map and successor feature map; **iii)** We analyze a stability–approximation error tradeoff: reward regression acts as a stabilizer, while solving PBE can decrease the approximation error.

Based on these observations, we develop a scalable online framework called *SF-Bellman* to solve the sparse-reward and online learning scenarios for continual and goal-conditioned control settings. The algorithm couples the proposed unified view on the task vector estimator with the deep SF architecture of Chua et al. (2024). In continual learning and goal-conditioned tasks with sparse-reward and online learning scenarios, it improves learning performance and enables faster transfer

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

across tasks. We show an average relative improvement of about $20-30\%$ on both continual learning tasks and goal-conditioned tasks compared to regression based method.

2 PRELIMINARIES

Notations. $[\boldsymbol{v}]_i$ for $\boldsymbol{v} \in \mathbb{R}^n$: i-th element of $\boldsymbol{v}$; $[\boldsymbol{A}]_{i,j}$ for $\boldsymbol{A} \in \mathbb{R}^{m\times n}$: (i,j)-th entry; For a matrix $\boldsymbol{D} \succeq 0$, $\|\boldsymbol{x}\|_{\boldsymbol{D}} := \sqrt{\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x}}$ for $\boldsymbol{x} \in \mathbb{R}^n$, and $||\boldsymbol{A}||_{\boldsymbol{D}} = \sup_{\boldsymbol{x}\neq 0} \frac{||\boldsymbol{A}\boldsymbol{x}||_{\boldsymbol{D}}}{||\boldsymbol{x}||_{\boldsymbol{D}}}$ for $\boldsymbol{A} \in \mathbb{R}^{n\times n}$; $\Delta(\mathcal{A})$: a probability simplex over a set $\mathcal{A}$; $\lambda_{\min}(\boldsymbol{A})$: the minimum eigenvalue of $\boldsymbol{A}$; $\mathrm{col}(\boldsymbol{A}) := \{\boldsymbol{A}\boldsymbol{w} : \boldsymbol{w} \in \mathbb{R}^n\}$; $\boldsymbol{I}$: the identity matrix with proper dimensions; $\mathrm{spec}(\boldsymbol{A})$: the set of eigenvalues of $\boldsymbol{A} \in \mathbb{R}^{n\times n}$

Markov Decision Process. A Markov decision process (MDP) is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S} := \{1, 2, \ldots, |\mathcal{S}|\}$ and $\mathcal{A} := \{1, 2, \ldots, |\mathcal{A}|\}$ denote the state and action spaces, respectively. $\gamma \in (0, 1)$ is the discount factor, $\mathcal{P}(\cdot \mid s, a) \in \Delta(\mathcal{S})$ is the transition probability function, and $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function. An agent at state $s \in \mathcal{S}$ chooses an action $a \in \mathcal{A}$ following a policy $\pi(\cdot \mid s) \in \Delta(\mathcal{A})$, transits to next state $s' \sim \mathcal{P}(\cdot \mid s, a)$, and receives a reward $r(s, a, s')$. The expected discounted sum of the rewards $Q^\pi(s, a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r(s_k, a_k, s_{k+1}) \mid (s_0, a_0) = (s, a), \pi\right]$ is called the Q-function. In matrix and vector notations, let $\boldsymbol{P}^\pi \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times|\mathcal{S}||\mathcal{A}|}$ denote the transition matrix such that $[\boldsymbol{P}^\pi]_{(s-1)|\mathcal{A}|+a,(s'-1)|\mathcal{A}|+a'} = \mathcal{P}(s' \mid s, a)\pi(a' \mid s')$. Define the reward vector by $\boldsymbol{R} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ such that $[\boldsymbol{R}]_{(s-1)|\mathcal{A}|+a} = \mathbb{E}[r(s, a, s') \mid s, a]$, and $\boldsymbol{D}_\mu$ be a diagonal matrix with diagonal entries $\mu(s, a)$ for $\mu \in \Delta(\mathcal{S} \times \mathcal{A})$. Throughout the paper, we assume that $\mu(s, a) > 0$. Given a feature map $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$, let $\boldsymbol{\Phi} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times d}$ be a feature matrix such that its $(s-1)|\mathcal{A}| + a$-th row is $\phi(s, a)^\top$, where $d \in \mathbb{N}$ denotes the feature dimension. We assume that $\boldsymbol{\Phi}$ has full column rank. Then, $\boldsymbol{\Phi}^\top \boldsymbol{D}_\mu \boldsymbol{\Phi}$ is invertible and the weighted Euclidean projection matrix is defined as

$$\boldsymbol{\Gamma}_{\boldsymbol{\Phi}} = \boldsymbol{\Phi}(\boldsymbol{\Phi}^\top \boldsymbol{D}_\mu \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{D}_\mu.$$

Successor features. In the successor feature framework (Barreto et al., 2017), the goal is to approximate the Q-function under the assumption that the expected reward function is linear in the features, i.e., $\mathbb{E}[r(s, a, s') \mid s, a] \approx \phi(s, a)^\top \boldsymbol{w}$ for some $\boldsymbol{w} \in \mathbb{R}^d$. The expected discounted sum of features is captured by the successor feature matrix following a policy π, expressed as follows:$\boldsymbol{\Psi}_\pi = (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)^{-1}\boldsymbol{\Phi} = \sum_{t=0}^{\infty} \gamma^t (\boldsymbol{P}^\pi)^t \boldsymbol{\Phi}$ where the row vector defines the successor feature map $\psi^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$. Then, the Q-function can be represented as $Q^\pi(s, a) \approx [\boldsymbol{\Psi}_\pi \boldsymbol{w}]_{(s-1)|\mathcal{A}|+a,1}$ with equality if $\boldsymbol{R} = \boldsymbol{\Phi}\boldsymbol{w} = (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{\Psi}_\pi \boldsymbol{w}$ holds.

Related Works. SFs support transfer by separating the prediction of future features from the rewards. This decomposition allows agents to reuse learned predictions when rewards change while the environment dynamics remain fixed (Barreto et al., 2017; Lehnert et al., 2017; Barreto et al., 2018; 2020; Borsa et al., 2019). Beyond transfer, SFs have also been used to learn exploratory behaviors and support reward-free pretraining (Machado et al., 2018; Liu and Abbeel, 2021).

Learning SFs directly from observations in online manner requires learning jointly the SF, a feature map, and a task vector. Prior SF methods generally relied on hand-designed feature maps, prespecified task vectors, or solving relatively small MDP problems. To resolve these challenge, Kulkarni et al. (2016); Zhang et al. (2017) combined reward regression and SF learning with an auxiliary reconstruction loss to preserve information in the learned features. Nonetheless, learning the full vector-valued SF remained inefficient, and Chua et al. (2024) proposed learning SFs along the task vector direction to avoid such optimization challenge.

Beyond learning the SF representation, SF methods require estimating the task vector. Kulkarni et al. (2016); Zhang et al. (2017); Chua et al. (2024) use reward regression, whereas Ma et al. (2020) used a Bellman loss objective for goal-conditioned transfer. Janz et al. (2019) combined reward-prediction and Bellman loss objectives to address exploration under uncertainty. These methods illustrate different choices of task vector objective, motivating a theoretical analysis of their relationship. We characterize when reward regression and the projected Bellman equation yield the same task vector and analyze the stability–approximation tradeoff of interpolating between them. Building on the task-directional SF architecture of Chua et al. (2024), we develop an algorithm that incorporates this interpolation and demonstrate its benefits in online continual-learning and goal-conditioned control with sparse rewards. Table 1 summarizes SF methods by task vector objective and SF learning scheme.

Beyond the SF-specific literature, our analysis connects to classical work on linear value-function approximation. Tsitsiklis and Van Roy (1996) analyzed the convergence of temporal-difference (TD) learning to the PBE solution, while Scherrer (2010) compared solution of PBE with Bellman residual minimization (BRM). Baird (1995) introduced residual algorithms that interpolate between TD and BRM updates, and Zhang et al. (2019) revisited this approach with deep neural networks. With exact SFs, PRE and PBE correspond to these TD and BRM update directions, respectively, over $\mathrm{col}(\boldsymbol{\Psi}_\pi)$. Our novel contribution is to interpret this interpolation as SF task-vector estimation and characterize its equivalence conditions and stability–approximation tradeoff which can also be interpreted for general function class beyond $\mathrm{col}(\boldsymbol{\Psi}_\pi)$.

Table 1: Classification by task vector objective, and SF learning scheme. The interpolation parameter η ($\eta = 0$: PRE; $\eta = 1$: PBE) indicate the corresponding interpolation parameters under our theoretical model. RR stands for reward regression.

Method	task vector objective	η	SF learning
Barreto et al. (2017; 2018; 2020)	RR	0	Full
Kulkarni et al. (2016)	RR	0	Full
Janz et al. (2019)	RR + Bellman loss	$1/2$	Full
Ma et al. (2020)	Bellman loss	1	Full
Chua et al. (2024)	RR	0	$\boldsymbol{w}$-direction
Ours	Interpolated PRE–PBE	$[0, 1]$	$\boldsymbol{w}$-direction

3 Projected Reward and Projected Bellman Equation

In this section, we characterize the equations whose solutions yield estimates of the Q-function via the successor feature $\boldsymbol{\Psi}_\pi$. We introduce two different approaches to estimate the Q-function, namely, solving the projected reward equation and the projected Bellman equation. We characterize the solution of each approach and show that they generally yield different estimates that only coincide under specific conditions. We then provide a unified perspective by showing that both approaches can be derived from a common objective function.

3.1 The Projected Reward and Projected Bellman Equation

Projected Reward Equation. We begin with the projected reward approach–solving a linear regression problem for the reward. The goal is to find a task vector $\boldsymbol{w} \in \mathbb{R}^d$ that approximates the reward vector, i.e., $\boldsymbol{R} \approx \boldsymbol{\Phi}\boldsymbol{w}$. Then, we can estimate the Q-function by $\boldsymbol{Q}^\pi \approx \boldsymbol{\Psi}_\pi \boldsymbol{w}$. From a projection perspective, the estimated task vector is obtained by solving the following equation:

$$\boldsymbol{\Psi}_\pi \boldsymbol{w}_{\mathrm{PRE}} = \boldsymbol{\Psi}_\pi \arg\min_{\boldsymbol{w}' \in \mathbb{R}^d} \|\boldsymbol{\Gamma}_{\boldsymbol{\Phi}} \boldsymbol{R} - \boldsymbol{\Phi}\boldsymbol{w}'\|^2_{\boldsymbol{D}_\mu} = \boldsymbol{\Psi}_\pi (\boldsymbol{\Phi}^\top \boldsymbol{D}_\mu \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{D}_\mu \boldsymbol{R} \tag{1}$$

where $\boldsymbol{w}_{\mathrm{PRE}}$ denotes the solution of the above equation. The solution can be seen as computing the discounted sum of projected reward, $\sum_{k=0}^{\infty} \gamma^k (\boldsymbol{P}^\pi)^k \boldsymbol{\Gamma}_{\boldsymbol{\Phi}} \boldsymbol{R}$.

Projected Bellman Equation. In contrast, the projected Bellman equation approach estimates the Q-function by projecting the Bellman target onto the column space of $\boldsymbol{\Psi}_\pi$:

$$\begin{aligned} \boldsymbol{\Psi}_\pi \boldsymbol{w}_{\mathrm{PBE}} &= \boldsymbol{\Psi}_\pi \cdot \arg\min_{\boldsymbol{w}' \in \mathbb{R}^d} \|\boldsymbol{\Gamma}_{\boldsymbol{\Psi}_\pi} (\boldsymbol{R} + \gamma \boldsymbol{P}^\pi \boldsymbol{\Psi}_\pi \boldsymbol{w}_{\mathrm{PBE}}) - \boldsymbol{\Psi}_\pi \boldsymbol{w}'\|^2_{\boldsymbol{D}_\mu} \\ &= \boldsymbol{\Psi}_\pi (\boldsymbol{\Psi}_\pi^\top \boldsymbol{D}_\mu (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi) \boldsymbol{\Psi}_\pi)^{-1} \boldsymbol{\Psi}^{\pi\top} \boldsymbol{D}_\mu \boldsymbol{R} \end{aligned} \tag{2}$$

where $\boldsymbol{w}_{\mathrm{PBE}}$ is the solution of the above equation. Note that this is equivalent to the projected Bellman equation (Tsitsiklis and Van Roy, 1996) with $\boldsymbol{\Psi}_\pi$ as the feature matrix. Different from (1), we are projecting the Bellman target $\boldsymbol{R} + \gamma \boldsymbol{P}^\pi \boldsymbol{\Psi}_\pi \boldsymbol{w}_{\mathrm{PBE}}$ to the column space of $\boldsymbol{\Psi}_\pi$.

Remark 3.1 (Connection to Bellman residual minimization and PBE)**.** *Scherrer (2010) compares Bellman residual minimization (BRM) and the solution of PBE over a common linear value-function class. In our setting, this class is* $\mathrm{col}(\boldsymbol{\Psi}_\pi)$*. For every* $\boldsymbol{w} \in \mathbb{R}^d$ *we have* $\boldsymbol{R} - \boldsymbol{\Phi}\boldsymbol{w} = \boldsymbol{R} + \gamma \boldsymbol{P}^\pi \boldsymbol{\Psi}_\pi \boldsymbol{w} - \boldsymbol{\Psi}_\pi \boldsymbol{w}$*. Thus, the PRE objective is exactly the Bellman residual minimization objective over* $\mathrm{col}(\boldsymbol{\Psi}_\pi)$*. Therefore,* $\boldsymbol{w}_{\mathrm{PRE}}$ *correspond to the solution of BRM studied by Scherrer (2010).*

3.2 Equivalence Conditions and Counterexamples

Although both approaches use the same SF representation $\boldsymbol{\Psi}_\pi$, they generally produce different task vectors because they apply projection to different quantities. We therefore first characterize the condition under which these two approaches nevertheless yield the same solution.

Lemma 3.2. *Suppose $\boldsymbol{\Psi}_\pi^\top \boldsymbol{D}_\mu(\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{\Psi}_\pi$ is invertible. Then, $\boldsymbol{w}_{\mathrm{PRE}} = \boldsymbol{w}_{\mathrm{PBE}}$ if and only if $\boldsymbol{\Psi}_\pi^\top \boldsymbol{D}_\mu(\boldsymbol{I} - \boldsymbol{\Gamma}_{\boldsymbol{\Phi}})\boldsymbol{R} = \boldsymbol{0}$.*

The proof is given in Appendix A.1. The above result characterizes the discrepancy between the two approaches through the reward-approximation residual $(\boldsymbol{I} - \boldsymbol{\Gamma}_{\boldsymbol{\Phi}})\boldsymbol{R}$. The lemma shows that the PRE and PBE approaches yield the same task vector if and only if the reward residual is also orthogonal to $\mathrm{col}(\boldsymbol{\Psi}_\pi)$. Thus, the two approaches may coincide even when the reward is not exactly representable by $\boldsymbol{\Phi}$. One sufficient condition for this is that $\mathrm{col}(\boldsymbol{\Phi})$ is invariant under $\boldsymbol{P}^\pi$.

Corollary 3.3. *Under the assumptions of Lemma 3.2, $\boldsymbol{w}_{\mathrm{PRE}} = \boldsymbol{w}_{\mathrm{PBE}}$ if either $\boldsymbol{R} \in \mathrm{col}(\boldsymbol{\Phi})$ or $\mathrm{col}(\boldsymbol{\Phi})$ is invariant under $\boldsymbol{P}^\pi$, i.e., $\boldsymbol{P}^\pi\boldsymbol{\Phi} = \boldsymbol{\Phi}\boldsymbol{N}$ for some $\boldsymbol{N} \in \mathbb{R}^{d\times d}$.*

The proof is given in Appendix A.2. The above results give two sufficient conditions for the equivalence condition in Lemma 3.2. In general, these conditions need not hold, and the PRE and PBE solutions can differ. The following example illustrates the difference.

Failure of Equivalence. Consider the Cafe–Bakery example in Figure 1. The agent receives a reward of one upon reaching the Restaurant (s_3) and zero otherwise. At s_0, it chooses between the Cafe route through s_1 and the Bakery route through s_2. From the Cafe, the Restaurant action reaches s_3 immediately. From the Bakery, the corresponding action remains in s_2 with probability 0.9, thereby delaying arrival at s_3. Although these two Restaurant actions have different expected immediate rewards, the feature map assigns them the same feature. Reward regression must therefore assign them a common reward estimate. This error is amplified by the successor features. The Bakery action may be repeated many times and its shared feature is accumulated repeatedly, causing PRE to overvalue the Bakery route. The distinct features associated with the initial Cafe and Bakery choices cannot correct this error under PRE because both choices have zero immediate reward and therefore receive zero weight. In contrast, PBE estimates the task vector using Bellman targets, which propagate subsequent rewards back to the initial choices. It can therefore assign non-zero weights to the features distinguishing the two routes. Consequently, under the ϵ-greedy policy considered here, PRE incorrectly ranks Bakery above Cafe, whereas PBE recovers the correct ordering and selects Cafe. The full construction is provided in Appendix C.

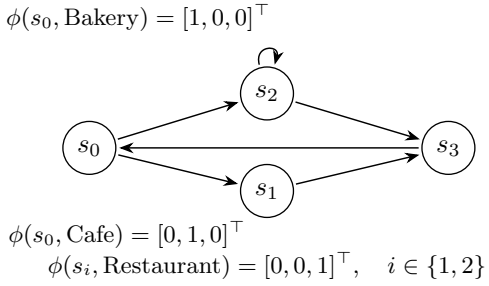


Figure 1: Cafe–Bakery example. The agent receives reward $+1$ upon reaching the restaurant (s_3) and zero elsewhere.

3.3 Unified Perspective

In general, however, it is not clear how to satisfy the conditions in Lemma 3.2. The two solutions need not coincide. Nonetheless, despite this difference, the two equations can be derived from a single unified objective. To this end, we define an interpolating matrix between $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}_\pi$ and consider the projection induced by it. For $\eta \in [0,1]$, let

$$\boldsymbol{B}_\eta^\pi = \eta\,\boldsymbol{\Psi}_\pi + (1-\eta)\,\boldsymbol{\Phi}, \qquad \boldsymbol{\Gamma}_{\boldsymbol{B}_\eta^\pi} = \boldsymbol{B}_\eta^\pi\big(\boldsymbol{B}_\eta^{\pi\top}\boldsymbol{D}_\mu\boldsymbol{B}_\eta^\pi\big)^{-1}\boldsymbol{B}_\eta^{\pi\top}\boldsymbol{D}_\mu \tag{3}$$

denote this interpolating matrix and its corresponding projection matrix.

Lemma 3.4. *Suppose $\boldsymbol{\Phi}^\top\boldsymbol{D}_\mu\boldsymbol{B}_\eta^\pi$ is invertible. The following problem admits a unique minimizer*

$$\boldsymbol{w}_\eta := \arg\min_{\boldsymbol{w}\in\mathbb{R}^d} \|\boldsymbol{\Gamma}_{\boldsymbol{B}_\eta^\pi}(\boldsymbol{R} + \gamma\boldsymbol{P}^\pi\boldsymbol{\Psi}_\pi\boldsymbol{w} - \boldsymbol{\Psi}_\pi\boldsymbol{w})\|_{\boldsymbol{D}_\mu}^2 = (\boldsymbol{B}_\eta^{\pi\top}\boldsymbol{D}_\mu\boldsymbol{\Phi})^{-1}(\boldsymbol{B}_\eta^{\pi\top}\boldsymbol{D}_\mu\boldsymbol{R}).$$

The proof is given in Appendix A.3. Note that the minimizer in the above lemma corresponds to $\boldsymbol{w}_{\mathrm{PRE}}$ when $\eta = 0$, and for $\eta = 1$, it corresponds to $\boldsymbol{w}_{\mathrm{PBE}}$. Let us define the interpolated equation from Lemma 3.4:

$$\boldsymbol{M}_\eta^\pi\boldsymbol{w}_\eta = \boldsymbol{b}_\eta^\pi, \quad \text{where} \quad \boldsymbol{M}_\eta^\pi := \boldsymbol{B}_\eta^{\pi\top}\boldsymbol{D}_\mu\boldsymbol{\Phi}, \boldsymbol{b}_\eta^\pi := \boldsymbol{B}_\eta^{\pi\top}\boldsymbol{D}_\mu\boldsymbol{R}. \tag{4}$$

4 APPROXIMATION GEOMETRY AND STABILITY

The interpolation parameter η in (3) determines the task vector solution between PRE and PBE. In this section, we study the approximation properties of the resulting solution and the stability of the corresponding deterministic and stochastic iterative updates used to compute it.

4.1 APPROXIMATION GEOMETRY: SOLUTION PATH AND APPROXIMATION ERROR

We first characterize the path traced by the interpolated solution as η varies between the PRE and PBE. This characterization reveals that $\boldsymbol{w}_\eta$ is generally not a linear interpolation between $\boldsymbol{w}_{\mathrm{PRE}}$ and $\boldsymbol{w}_{\mathrm{PBE}}$.

Proposition 4.1. *Suppose that $\boldsymbol{M}_1^\pi$ is invertible and $\boldsymbol{K} := (\boldsymbol{M}_0^\pi)^{-1}\boldsymbol{M}_1^\pi$ is diagonalizable with eigenpairs $\{(\lambda_i, \boldsymbol{v}_i)\}_{i=1}^d$, i.e., $\boldsymbol{K}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$. Then, there exists some $\{c_i\}_{i=1}^d$ such that $\boldsymbol{v} := \boldsymbol{w}_{\mathrm{PBE}} - \boldsymbol{w}_{\mathrm{PRE}} = \sum_{i=1}^d c_i \boldsymbol{v}_i$. Moreover, for every $\eta \in [0,1]$ such that $\boldsymbol{M}_\eta^\pi$ is non-singular, the following holds:*

$$\boldsymbol{w}_\eta = \boldsymbol{w}_{\mathrm{PRE}} + \sum\nolimits_{i=1}^d \beta_\eta(\lambda_i)\, c_i \boldsymbol{v}_i, \qquad \beta_\eta(\lambda) := \eta\lambda/(1-\eta+\eta\lambda). \tag{5}$$

The proof is given in Appendix B.1. The above result shows that $\boldsymbol{w}_\eta$ is generally not a linear interpolation between $\boldsymbol{w}_{\mathrm{PRE}}$ and $\boldsymbol{w}_{\mathrm{PBE}}$. Rather, each eigenvector direction follows its own path, determined by $\beta_\eta(\lambda_i)$. Thus, η changes not only the overall amount of movement toward the PBE solution, but also the relative contributions of the eigenvector. While the above result gives an exact instance-dependent expression, we next derive uniform approximation guarantees. Let

$$\boldsymbol{w}^* := \arg\min_{\boldsymbol{w}\in\mathbb{R}^d} \|\boldsymbol{Q}^\pi - \boldsymbol{\Psi}_\pi \boldsymbol{w}\|_{\boldsymbol{D}_\mu}^2$$

denote the best approximation of $\boldsymbol{Q}^\pi$ within the linear function class. In the following, the first result holds under stationarity, while the second provides a sharper guarantee under reversibility.

Proposition 4.2 (Approximation guarantees for the interpolated equation)**.** *Let $\boldsymbol{\mu} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ be a vector such that $[\boldsymbol{\mu}]_{(s-1)|\mathcal{A}|+a} = \mu(s,a)$ for $\mu \in \Delta(\mathcal{S}\times\mathcal{A})$. The following holds:*

(i) ***(Stationary case.)*** *Suppose μ is stationary for $\boldsymbol{P}^\pi$, i.e. $\boldsymbol{\mu}^\top \boldsymbol{P}^\pi = \boldsymbol{\mu}^\top$. Then the interpolated equation admits a unique solution $\boldsymbol{w}_\eta$ and*

$$\left\|\boldsymbol{Q}^\pi - \boldsymbol{\Psi}_\pi \boldsymbol{w}_\eta\right\|_{\boldsymbol{D}_\mu}^2 \;\le\; \kappa_\eta^{\mathrm{stat}} \left\|\boldsymbol{Q}^\pi - \boldsymbol{\Psi}_\pi \boldsymbol{w}^*\right\|_{\boldsymbol{D}_\mu}^2, \quad \kappa_\eta^{\mathrm{stat}} := \frac{\left[1+(1-\eta)\gamma^2\right]^2}{(1-\gamma^2)\left[1-(1-\eta)^2\gamma^2\right]}. \tag{6}$$

The constant $\kappa_\eta^{\mathrm{stat}}$ is strictly decreasing in η.

(ii) ***(Reversible case.)*** *Suppose in addition that the chain induced by π is reversible with respect to $\boldsymbol{D}_\mu$, i.e. $\boldsymbol{D}_\mu \boldsymbol{P}^\pi = \boldsymbol{P}^{\pi\top} \boldsymbol{D}_\mu$. Let $\nu_{\max} := \max \mathrm{spec}(\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)$, which satisfies $1-\gamma \le \nu_{\max} \le 1+\gamma$. Set $g_\eta(\nu) := (1-\eta)\nu^2 + \eta\nu$ and $\xi_\eta := g_\eta(\nu_{\max})/g_\eta(1-\gamma) \ge 1$. Then, the inequality in (6) holds with the smaller constant $\kappa_\eta := (\xi_\eta+1)^2/(4\xi_\eta) \le \kappa_\eta^{\mathrm{stat}}$. The constant κ_η is strictly decreasing in η unless $\nu_{\max} = 1-\gamma$.*

The proof is given in Appendix B.2. In both the stationary and reversible settings, the worst-case approximation factor decreases monotonically with η. That is, the worst case approximation guarantee improves as the interpolation moves from PRE toward PBE.

Remark 4.3. *At $\eta = 1$, the bound in Proposition 4.2(i) reduces to $\kappa_1^{\mathrm{stat}} = \frac{1}{1-\gamma^2}$, which recovers the classical squared-error approximation guarantee for the PBE (Tsitsiklis and Van Roy, 1996). For $\eta = 0$, we have $\kappa_0^{\mathrm{stat}} = \left(\frac{1+\gamma^2}{1-\gamma^2}\right)^2$. Thus, under stationarity, the worst-case approximation guarantee improves monotonically as η moves from PRE toward PBE. Moreover, in asymptotic sense, note that κ_1^{stat} improves dependency on $\frac{1}{1-\gamma}$ by one over κ_0^{stat}. Under reversibility, Proposition 4.2(ii) sharpens this uniform guarantee while preserving the same monotonic improvement with η.*

4.2 STABILITY OF THE INTERPOLATED UPDATE

While the previous section characterizes the approximation guarantees of the solution $\boldsymbol{w}_\eta$, computing $\boldsymbol{w}_\eta$ in practice requires an iterative, sample-based algorithm. To solve the interpolated equation in (4), we consider the iterative update, for $k \in \mathbb{N}$ and $\boldsymbol{w}_0 \in \mathbb{R}^d$, described in Algorithm 1.

Algorithm 1 Interpolated task vector update

Input: $\eta \in [0,1]$, step sizes $\alpha_k \in [0,1]$, $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$, $\psi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$
for $k = 0, 1, 2, \ldots$ **do**
  Sample $(s_k, a_k) \overset{\text{i.i.d.}}{\sim} \mu$, $s'_k \sim \mathcal{P}(\cdot \mid s_k, a_k)$, $a'_k \sim \pi(\cdot \mid s'_k)$, and observe $r_k := r(s_k, a_k, s'_k)$
  $\boldsymbol{g}_k^{\text{PRE}} \leftarrow \phi(s_k, a_k)\big(r_k - \phi(s_k, a_k)^\top \boldsymbol{w}_k\big)$
  $\boldsymbol{g}_k^{\text{PBE}} \leftarrow \psi(s_k, a_k)\big(r_k + \gamma \psi(s'_k, a'_k)^\top \boldsymbol{w}_k - \psi(s_k, a_k)^\top \boldsymbol{w}_k\big)$
  $\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k + \alpha_k\big[(1-\eta)\boldsymbol{g}_k^{\text{PRE}} + \eta \boldsymbol{g}_k^{\text{PBE}}\big]$
end for

Algorithm 2 SF-Bellman (Simplified pseudo-code)

Input: $\eta \in [0,1]$, step sizes $\alpha_k, \beta_k \in [0,1]$, $\phi_{\boldsymbol{v}_0} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$, $\psi_{\boldsymbol{u}_0} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$
for $k = 0, 1, 2, \ldots$ **do**
  Select $a_k \in \mathcal{A}$ following an ϵ-greedy policy over $\psi_{\boldsymbol{u}_k}(s_k, \cdot)^\top \boldsymbol{w}_k$
  Observe $s'_k \sim \mathcal{P}(\cdot \mid s_k, a_k)$, and $r_k := r(s_k, a_k, s'_k)$

$$\mathcal{L}_k^{\text{SF}} = \frac{1}{2}\big(r_k + \gamma\, \text{SG}\big(\max_{a \in \mathcal{A}} \psi_{\boldsymbol{u}_k}(s'_k, a)^\top \boldsymbol{w}_k\big) - \psi_{\boldsymbol{u}_k}(s_k, a_k)^\top \boldsymbol{w}_k\big)^2 \tag{7}$$

  Normalize $\phi_{\boldsymbol{v}_k}(s_k, a_k) \leftarrow \phi_{\boldsymbol{v}_k}(s_k, a_k) / ||\phi_{\boldsymbol{v}_k}(s_k, a_k)||_2$
  Update $\boldsymbol{u}_{k+1} \leftarrow \boldsymbol{u}_k - \alpha_k \nabla_{\boldsymbol{u}} \mathcal{L}_k^{\text{SF}}$
  Update $\boldsymbol{w}_k$ following Algorithm 1 using step-size β_k
end for

Under standard stochastic approximation theory, the asymptotic convergence of Algorithm 1 depends on the real parts of the eigenvalues of $\boldsymbol{M}_\eta^\pi$—specifically, a sufficient condition is $\boldsymbol{x}^\top \boldsymbol{M}_\eta^\pi \boldsymbol{x} > 0$ for all non-zero $\boldsymbol{x}$. The precise threshold on η is characterized in the following lemma:

Lemma 4.4 (Off-policy stability). *Let* $\mathcal{S} = \{\boldsymbol{s} \in \text{col}(\boldsymbol{\Psi}_\pi) : \boldsymbol{s}^\top \boldsymbol{D}_\mu (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{s} < 0\}$ *and*

$$c^* = \inf_{\boldsymbol{y} \in \mathcal{S}} ||(\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{y}||_{\boldsymbol{D}_\mu}^2 / \big(-\boldsymbol{y}^\top \boldsymbol{D}_\mu (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{y}\big).$$

Then $\boldsymbol{x}^\top \boldsymbol{M}_\eta^\pi \boldsymbol{x} \geq 0$ *if and only if* $\eta \leq \frac{c^*}{1+c^*}$.

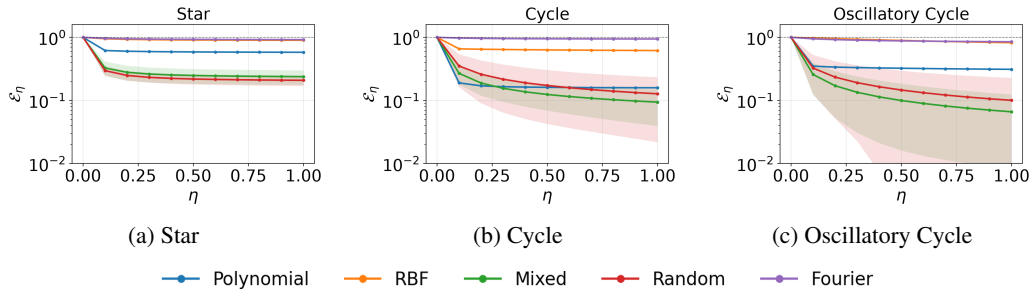
The proof is given in Appendix B.3. The preceding result shows that the reward regression term can stabilize the update in the off-policy setting. When $\eta = 0$, the stability of Algorithm 1 follows from the positive semi-definiteness of $\boldsymbol{\Phi}^\top \boldsymbol{D}_\mu \boldsymbol{\Phi}$. As η increases, weight shifts toward off-policy update, in which the update can be unstable (Tsitsiklis and Van Roy, 1996). However, this is only a sufficient condition and stability may also hold beyond the certified regime. The following theorem characterizes the convergence of the iterate. The proof is given in Appendix B.4.

Theorem 4.5 (Almost-sure convergence). *Let* $\eta < c^*/(1+c^*)$ *as in Lemma 4.4, and the step sizes satisfy* $\sum_k \alpha_k = \infty$, $\sum_k \alpha_k^2 < \infty$. *Then,* $\boldsymbol{w}_k \to \boldsymbol{w}_\eta$ *for the iterate in Algorithm 1.*

5 Online Control with SF-Bellman

The analysis of Section 3 and 4 assumes a fixed feature map ϕ and successor-feature map ψ_π for a given policy π. In practice, the features and successor features are instead learned online, and the policy changes during learning. In this section, we develop SF-Bellman, an online-control algorithm that extends the interpolated update to this setting.

SF-Bellman learns two objects: the successor features and the task vector. The successor features are trained following the $\boldsymbol{w}$-directional scheme of Chua et al. (2024), rather than regressing the full d-dimensional ψ_π. The successor feature is parameterized by $\psi_{\boldsymbol{u}}(s, a)$ where $\boldsymbol{u}$ is a learnable parameter in Algorithm 2. Then, $\boldsymbol{u}$ is updated with taking gradient on $\mathcal{L}_k^{\text{SF}}$ in (7) where $\text{SG}(\cdot)$ denotes the stop-gradient operator. The task vector is updated by the interpolated rule of Algorithm 1, applied unchanged with the learned maps in place of the fixed ones. The feature map $\phi_{\boldsymbol{v}_k}$ are normalized so that the overall update is stabilized. At $\eta = 0$ the algorithm reduces to that of Chua et al. (2024). The full algorithm with mini-batches, replay, and target networks is given in Appendix D.



(a) Star (b) Cycle (c) Oscillatory Cycle

Figure 2: Results on the synthetic MDPs. Normalized function approximation error $\mathcal{E}_\eta$ as a function of η. Curves and shaded regions report the mean and standard error over 20 seeds.

6 Experiments

We evaluate the interpolated task vector estimator and our corresponding algorithm by analyzing the approximation behavior of the solution $\boldsymbol{w}_\eta$, and the performance of online control with *SF-Bellman*. We first study the interpolated estimator in policy-evaluation problems to understand how η changes the solution and its approximation error. We then evaluate *SF-Bellman* in continual-learning and goal-conditioned control tasks, comparing the algorithm with other baselines.

6.1 Policy Evaluation : Comparison of solutions on synthetic MDPs

We evaluate the approximation properties of the interpolated task vector estimator on small synthetic Markov chains for which the transition matrix, stationary distribution, and SFs can be computed exactly. This removes sampling and optimization error from the comparison. We consider three reversible chains: an n-cycle with sparse rewards, a star graph with a non-uniform stationary distribution, and an oscillatory n-cycle with alternating rewards. For each environment, we use five $d = 3$ feature classes: a mixed eigenbasis of $\boldsymbol{P}^\pi$, polynomial features, Gaussian radial-basis, Fourier, and random features. Full environment and feature definitions are given in Appendix E.1.

For each η, we compute the interpolated solution $\boldsymbol{w}_\eta$ and report the normalized approximation error $\mathcal{E}_\eta = \frac{\|\boldsymbol{Q}^\pi - \boldsymbol{\Psi}_\pi \boldsymbol{w}_\eta\|_{\boldsymbol{D}_\mu}}{\|\boldsymbol{Q}^\pi - \boldsymbol{\Psi}_\pi \boldsymbol{w}_{\mathrm{PRE}}\|_{\boldsymbol{D}_\mu}}$. Figure 2 reports $\mathcal{E}_\eta$ for each environment and feature class, averaged over 20 seeds. Across all environments and feature classes, $\mathcal{E}_\eta$ is non-increasing in η, consistent with the monotone guarantee of Proposition 4.2 (ii), with most of the improvement occurring at small η.

6.2 Online Control with SF-Bellman

Evaluation settings. We consider both continual learning and goal-conditioned continuous-control tasks. These scenarios capture two forms of task variation central to SF methods: adaptation to recurring task changes and generalization across a continuous family of goals. The former tests whether *SF-Bellman* can reuse its learned representation when previously encountered tasks return, whereas the latter tests whether it generalizes to goals not observed during training. In both settings, rewards are sparse, making them challenging for reward regression and therefore providing a meaningful test of whether the interpolated update improves learning. We also compare against baseline algorithms including deep Q-network (DQN, Mnih et al. (2015)), deep deterministic policy gradient algorithm (DDPG, Lillicrap et al. (2015)), soft actor critic (SAC, Haarnoja et al. (2018)) and algorithms that take gradient steps on the full successor feature loss (Kulkarni et al., 2016; Ma et al., 2020). We call the algorithms that use the full successor feature loss *SF-full*.

Continual-learning protocol. In the continual-learning experiments, we alternate between two tasks according to an $A \to B \to A \to B$ sequence. Tasks A and B share the same observation and action spaces and differ only in the goal location. This recurring sequence evaluates both adaptation following a task switch and reuse of previously acquired knowledge when a task reappears. To efficiently evaluate adaptation and the reuse of previously acquired knowledge when a task reappears, our architecture employs a universal value function approximation scheme following Schaul et al. (2015); Chua et al. (2024). The detailed implementations are provided in Appendix F.1.

Metric	Algorithm	FourRoomS9	DoorKey8x8	KeyCorridorS3R1	PointMaze	Reacher	PandaReach
Score	SF-SIMPLE ($\eta = 0$)	0.931 ± 0.013	0.829 ± 0.061	0.660 ± 0.084	$\mathbf{97.263 \pm 23.087}$	35.823 ± 2.744	25.333 ± 3.546
	SF-BELLMAN ($\eta = 0.99$)	0.967 ± 0.010	$\mathbf{0.982 \pm 0.008}$	$\mathbf{0.924 \pm 0.022}$	$\mathbf{115.207 \pm 23.569}$	$\mathbf{41.577 \pm 1.507}$	$\mathbf{33.800 \pm 2.737}$
	SF-BELLMAN ($\eta = 1.0$)	0.941 ± 0.017	0.923 ± 0.066	0.772 ± 0.104	$\mathbf{101.610 \pm 23.547}$	38.317 ± 2.250	30.400 ± 3.681
	SF-FULL ($\eta = 0$)	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	1.770 ± 0.818	10.000 ± 2.683	16.000 ± 2.735
	DQN / DDPG	$\mathbf{0.990 \pm 0.005}$	0.739 ± 0.090	0.776 ± 0.078	36.310 ± 16.659	31.033 ± 3.679	27.167 ± 3.382
Steps to 80% ($\times 10^3$)	SF-SIMPLE ($\eta = 0$)	18.7 ± 1.7	11.2 ± 3.0	23.7 ± 4.0	$\mathbf{48.6 \pm 4.0}$	24.1 ± 3.9	38.6 ± 4.6
	SF-BELLMAN ($\eta = 0.99$)	16.0 ± 1.9	$\mathbf{5.7 \pm 1.7}$	$\mathbf{10.3 \pm 2.3}$	$\mathbf{48.2 \pm 4.0}$	$\mathbf{18.1 \pm 3.3}$	$\mathbf{23.9 \pm 4.6}$
	SF-BELLMAN ($\eta = 1.0$)	12.8 ± 2.2	$\mathbf{6.5 \pm 3.1}$	15.1 ± 4.7	$\mathbf{45.2 \pm 4.4}$	$\mathbf{17.1 \pm 4.0}$	34.1 ± 5.2
	SF-FULL ($\eta = 0$)	50.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0	62.7 ± 1.0	44.2 ± 4.9	43.1 ± 4.3
	DQN / DDPG	$\mathbf{7.6 \pm 0.7}$	20.6 ± 4.4	15.8 ± 3.9	61.0 ± 1.9	24.1 ± 4.8	41.8 ± 4.2
AUC diff	SF-SIMPLE ($\eta = 0$)	0.085 ± 0.022	0.429 ± 0.035	0.344 ± 0.049	16.739 ± 15.501	10.172 ± 3.434	10.875 ± 2.134
	SF-BELLMAN ($\eta = 0.99$)	0.149 ± 0.023	$\mathbf{0.567 \pm 0.035}$	$\mathbf{0.506 \pm 0.017}$	$\mathbf{39.503 \pm 9.574}$	$\mathbf{12.818 \pm 2.183}$	$\mathbf{13.567 \pm 2.452}$
	SF-BELLMAN ($\eta = 1.0$)	0.106 ± 0.031	0.498 ± 0.040	0.472 ± 0.065	24.619 ± 11.271	$\mathbf{11.908 \pm 2.271}$	$\mathbf{11.967 \pm 2.675}$
	SF-FULL ($\eta = 0$)	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	2.055 ± 1.165	0.202 ± 1.290	5.363 ± 1.012
	DQN / DDPG	$\mathbf{0.179 \pm 0.028}$	0.413 ± 0.065	$\mathbf{0.496 \pm 0.055}$	0.329 ± 5.540	$\mathbf{11.502 \pm 2.463}$	3.183 ± 2.616

Table 2: DQN is used for MiniGrid and DDPG for continuous control. Bold indicates results within one standard error of the best mean. Metrics include the **Final Score** (the average score over the last 10% of evaluation in the final cycle), **Steps to 80%** (the environment steps required to reach 80% of that maximum score in the final cycle), and **AUC Difference** (the difference in area under the curve between the first and last cycles). Results are mean $\pm$ standard error across over 15 seeds.

We consider both discrete and continuous-action environments. For discrete-control, we evaluate *SF-Bellman* on FourRoomS9, DoorKey8x8 and KeyCorridorS3N1, three image-based MiniGrid environments. FourRoomS9 is a simple navigation task whereas DoorKey8x8 and KeyCorridorS3N1 requiring multi-step navigation and object interaction: DoorKey8x8 involves collecting a key and opening a door, while KeyCorridorS3N1 requires searching multiple locations to find the key. The MiniGrid environments are based on Pignatelli et al. (2026). For continuous control, we consider sparse-reward adaptations of Reacher from the DeepMind Control Suite (Tassa et al., 2018), PointMaze, derived from the Maze2D task introduced in D4RL (Fu et al., 2020), and PandaReach, adapted from panda-gym (Gallouédec et al., 2021). The reward is $+1$ when the agent reaches goal and otherwise zero. In MiniGrid envs, the episode terminates when reaching the goal, whereas continuous control tasks operate under a finite horizon, requiring the agent to stabilize around the goal.

As shown in Table 2 and Figure 3, *SF-Bellman* with $\eta = 0.99$ improves both final performance and learning efficiency across discrete and continuous control tasks. In MiniGrid, it outperforms SF-Simple and DQN in all three mean metrics on DoorKey8x8 and KeyCorridorS3R1. In continuous control, it achieves higher mean final scores and AUC differences than SF-Simple and DDPG across all three tasks, with earlier threshold achievement on Reacher and PandaReach and comparable to SF-Simple on PointMaze. Relative to the PBE endpoint ($\eta = 1$), it consistently achieves higher mean final scores and AUC differences. These findings suggest that a small reward-regression component can benefit Bellman-based task-vector learning, consistent with the stability–approximation tradeoff in Proposition 4.2 and Lemma 4.4. Overall it gives about $+22\%$ improvement over the reward-regression baseline and $+54.2\%$ over DQN/DDPG.

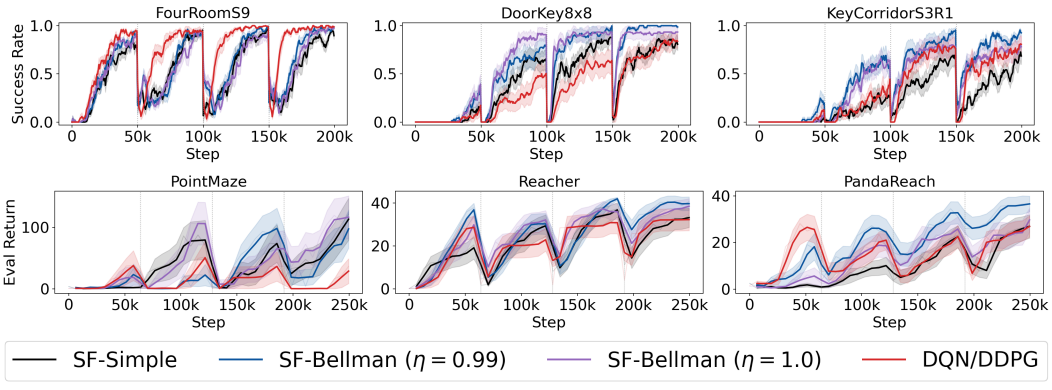


Figure 3: Learning curves on continual-learning environments

Goal-conditioned control. We further evaluate *SF-Bellman* on four goal-conditioned continuous-control tasks: Reacher, PointMaze, PandaReach, and Pick-and-Place. At the beginning of each training episode, a goal is sampled from the prescribed training distribution, whereas evaluation is conducted on held-out goals excluded from this distribution to test the generalization performance. For Pick-and-Place, we adopt a curriculum-based learning scheme while keeping the same training and evaluation protocol. The details are provided in Appendix H.2 and I. We compare SF-Simple ($\eta = 0$), *SF-Bellman* with $\eta \in \{0.99, 1\}$, SF-Full and DDPG. Following Ma et al. (2020), we implemented an additional network to output a goal embedding as the task vector. As shown in Table 3, both *SF-Bellman* variants achieve higher held-out and train scores than SF-Simple and DDPG, and reach the success threshold earlier across all three environments. For $\eta \in \{0.99, 1\}$, their performance is comparable across the tasks, indicating that the gains are robust near the PBE endpoint. Overall, the results demonstrate that shifting the task vector update from reward regression toward Bellman learning improves held-out-goal performance. The overall normalized scores (SF-Bellman score divided by baseline score) gives $+31\%$ improvement over SF-Simple and $+15\%$ improvement over DDPG.

Table 3: Performance on goal-conditioned continuous-control tasks. We report the final evaluation score and the Steps to 80% metric described in Table 2. For Pick-and-Place, we use curriculum learning and report the number of steps required to reach Level 5 of the curriculum and reports average held-out score of level 5 and 6. SF-Full failed to reach the level 5.

	SF-Simple	SF-Bellman ($\eta = 0.99$)	SF-Bellman ($\eta = 1$)	SF-Full ($\eta = 1$)	DDPG/SAC
Reacher					
Held-out Score	34.89 ± 1.53	$\mathbf{39.94 \pm 1.12}$	37.57 ± 1.15	3.04 ± 0.53	38.19 ± 1.29
Train Score	42.63 ± 0.43	42.97 ± 0.58	$\mathbf{43.89 \pm 0.09}$	4.59 ± 0.52	43.63 ± 0.10
PointMaze					
Held-out Score	167.92 ± 17.33	$\mathbf{207.63 \pm 7.41}$	$\mathbf{200.33 \pm 9.32}$	0.32 ± 0.09	$\mathbf{201.19 \pm 17.80}$
Train Score	190.77 ± 15.72	$\mathbf{223.54 \pm 3.08}$	$\mathbf{225.69 \pm 4.06}$	0.88 ± 0.74	221.48 ± 10.45
PandaReach					
Held-out Score	17.90 ± 2.83	$\mathbf{27.80 \pm 2.00}$	$\mathbf{28.64 \pm 1.69}$	0.20 ± 0.04	20.22 ± 2.65
Train Score	26.81 ± 2.08	33.34 ± 1.24	$\mathbf{34.76 \pm 1.26}$	0.09 ± 0.04	28.42 ± 1.59
Pick-and-Place					
Held-out Score	10.55 ± 6.68	$\mathbf{256.78 \pm 7.67}$	227.85 ± 13.40	–	192.13 ± 24.28
Train Score	13.51 ± 8.37	$\mathbf{300.52 \pm 7.04}$	280.87 ± 11.89	–	221.28 ± 29.81

7 Conclusion

This work bridges two task vector estimators within the SF framework: reward regression, cast as PRE, and the PBE. We establish that they generally yield distinct solutions and characterize their necessary and sufficient equivalence conditions. By formulating a unified objective through an interpolating matrix governed by a parameter $\eta \in [0, 1]$, we characterize a fundamental stability-approximation tradeoff: reward regression acts as a stabilizer in off-policy regimes, whereas solving the PBE reduces the worst-case approximation error. Building on this theoretical insight, we introduce *SF-Bellman*, a scalable online algorithm that incorporates the interpolated task vector update into the task vector-directional architecture. Future works include extending the framework to behavioral foundation models based on the successor measure.

AI use statement

In this work, we used generative AI tools for assist in the writing of proofs and to implement methods. We have not used generative AI tools for generating synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat dataset and support qualitative and thematic data analysis, interpret results are not applicable to this work. Additionally, we used generative AI tools for create or modify scientific figures or images. LLM-generated code and sentences were verified or tested for correctness by at least 2 authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine learning proceedings 1995*, pages 30–37. Elsevier, 1995.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P Van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, pages 501–510. PMLR, 2018.

André Barreto, Shaobo Hou, Diana Borsa, David Silver, and Doina Precup. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117 (48):30079–30087, 2020.

Vivek S Borkar and Sean P Meyn. The ODE method for convergence of stochastic approximation and reinforcement learning. *SIAM Journal on Control and Optimization*, 38(2):447–469, 2000.

Diana Borsa, Andre Barreto, John Quan, Daniel J. Mankowitz, Hado van Hasselt, Remi Munos, David Silver, and Tom Schaul. Universal successor features approximators. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1VWjiRcKX.

Raymond Chua, Arna Ghosh, Christos Kaplanis, Blake Aaron Richards, and Doina Precup. Learning successor features the simple way. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=rI7oZj1WMc.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624, 1993.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pages 482–495. PMLR, 2017.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Quentin Gallouédec, Nicolas Cazin, Emmanuel Dellandréa, and Liming Chen. panda-gym: Open-source goal-conditioned environments for robotic learning. *arXiv preprint arXiv:2106.13687*, 2021.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

David Janz, Jiri Hron, Przemysław Mazur, Katja Hofmann, José Miguel Hernández-Lobato, and Sebastian Tschiatschek. Successor uncertainties: exploration and uncertainty in temporal difference learning. *Advances in neural information processing systems*, 32, 2019.

Tejas D Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J Gershman. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*, 2016.

Lucas Lehnert, Stefanie Tellex, and Michael L Littman. Advantages and limitations of using successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1708.00102*, 2017.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pages 6736–6747. PMLR, 2021.

Chen Ma, Dylan R Ashley, Junfeng Wen, and Yoshua Bengio. Universal successor features for transfer reinforcement learning. *arXiv preprint arXiv:2001.04025*, 2020.

Marlos C Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. In *International Conference on Learning Representations*, 2018.

Matthew McLeod, Chunlok Lo, Matthew Schlegel, Andrew Jacobsen, Raksha Kumaraswamy, Martha White, and Adam White. Continual auxiliary task learning. *Advances in neural information processing systems*, 34:12549–12562, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Eduardo Pignatelli, Jarek Liesen, Robert Lange, Chris Lu, Pablo Samuel Castro, and Laura Toni. Navix: Scaling minigrid environments with jax. *Advances in Neural Information Processing Systems*, 38, 2026.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015.

Bruno Scherrer. Should one compute the Temporal Difference fix point or minimize the Bellman Residual? The unified oblique projection view. In *ICML*, Haïfa, Israel, June 2010. URL `https://inria.hal.science/inria-00537403`.

Richard S Sutton, Csaba Szepesvári, and Hamid Reza Maei. A convergent $O(n)$ algorithm for off-policy temporal-difference learning with linear function approximation. *Advances in Neural Information Processing Systems*, 21(21):1609–1616, 2008.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

John Tsitsiklis and Benjamin Van Roy. Analysis of temporal-diffference learning with function approximation. In M.C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996. URL `https://proceedings.neurips.cc/paper_files/paper/1996/file/e00406144c1e7e35240afed70f34166a-Paper.pdf`.

Charles W Wampler. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1): 93–101, 1986.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A Kahrs, et al. Mujoco playground. *arXiv preprint arXiv:2502.08844*, 2025.

Jingwei Zhang, Jost Tobias Springenberg, Joschka Boedecker, and Wolfram Burgard. Deep reinforcement learning with successor features for navigation across similar environments. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 2371–2378. IEEE, 2017.

Shangtong Zhang, Wendelin Boehmer, and Shimon Whiteson. Deep residual reinforcement learning. *arXiv preprint arXiv:1905.01072*, 2019.

A Omitted proofs in Section 3

A.1 Proof of Lemma 3.2

Proof. Let $\boldsymbol{M} := \boldsymbol{\Psi}_\pi^\top \boldsymbol{D}_\mu \boldsymbol{\Phi}$. By the successor-feature identity $(\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{\Psi}_\pi = \boldsymbol{\Phi}$,

$$\boldsymbol{\Psi}_\pi^\top \boldsymbol{D}_\mu (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{\Psi}_\pi = \boldsymbol{\Psi}_\pi^\top \boldsymbol{D}_\mu \boldsymbol{\Phi} = \boldsymbol{M},$$

so the hypothesis states precisely that $\boldsymbol{M}$ is invertible. From (1) and (2),

$$\boldsymbol{\Phi}^\top \boldsymbol{D}_\mu \boldsymbol{\Phi}\, \boldsymbol{w}_{\mathrm{PRE}} = \boldsymbol{\Phi}^\top \boldsymbol{D}_\mu \boldsymbol{R}, \qquad \boldsymbol{M}\boldsymbol{w}_{\mathrm{PBE}} = \boldsymbol{\Psi}_\pi^\top \boldsymbol{D}_\mu \boldsymbol{R}.$$

Since $\boldsymbol{\Gamma}_{\boldsymbol{\Phi}} \boldsymbol{R} = \boldsymbol{\Phi}\boldsymbol{w}_{\mathrm{PRE}}$,

$$\begin{aligned}\boldsymbol{\Psi}_\pi^\top \boldsymbol{D}_\mu (\boldsymbol{I} - \boldsymbol{\Gamma}_{\boldsymbol{\Phi}})\boldsymbol{R} &= \boldsymbol{\Psi}_\pi^\top \boldsymbol{D}_\mu \boldsymbol{R} - \boldsymbol{\Psi}_\pi^\top \boldsymbol{D}_\mu \boldsymbol{\Phi}\boldsymbol{w}_{\mathrm{PRE}} \\ &= \boldsymbol{M}\big(\boldsymbol{w}_{\mathrm{PBE}} - \boldsymbol{w}_{\mathrm{PRE}}\big).\end{aligned}$$

As $\boldsymbol{M}$ is invertible, the left-hand side vanishes if and only if $\boldsymbol{w}_{\mathrm{PRE}} = \boldsymbol{w}_{\mathrm{PBE}}$. □

A.2 Proof of Corollary 3.3

Proof. Since $\boldsymbol{\Phi}$ is $\boldsymbol{P}^\pi$-invariant,

$$\begin{aligned}\boldsymbol{\Psi}_\pi =& (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)^{-1}\boldsymbol{\Phi} \\ =& \sum_{k=0}^{\infty} \gamma^k (\boldsymbol{P}^\pi)^k \boldsymbol{\Phi} \\ =& \boldsymbol{\Phi} \sum_{k=0}^{\infty} \gamma^k \boldsymbol{N}^k \\ =& \boldsymbol{\Phi}(\boldsymbol{I} - \gamma \boldsymbol{N})^{-1}.\end{aligned}$$

Therefore, $\mathrm{col}(\boldsymbol{\Psi}_\pi) = \mathrm{col}(\boldsymbol{\Phi})$, which concludes the proof. □

A.3 Proof of Lemma 3.4

Proof. We have

$$\boldsymbol{B}_\eta^{\pi\top} \boldsymbol{D}_\mu (\boldsymbol{R} + \gamma \boldsymbol{P}^\pi \boldsymbol{\Psi}_\pi \boldsymbol{w} - \boldsymbol{\Psi}_\pi \boldsymbol{w}) = \boldsymbol{B}_\eta^{\pi\top} \boldsymbol{D}_\mu \boldsymbol{R} - \boldsymbol{B}_\eta^{\pi\top} \boldsymbol{D}_\mu \boldsymbol{\Phi}\boldsymbol{w}.$$

Therefore, the solution becomes $\boldsymbol{w} = (\boldsymbol{B}_\eta^{\pi\top} \boldsymbol{D}_\mu \boldsymbol{\Phi})^{-1}(\boldsymbol{B}_\eta^{\pi\top} \boldsymbol{D}_\mu \boldsymbol{R})$.

Since $B_\eta^{\pi\top} D_\mu \Phi$ is invertible, the minimizer is satisfied if and only if $w = (\boldsymbol{B}_\eta^{\pi\top} \boldsymbol{D}_\mu \Phi)^{-1} \boldsymbol{B}_\eta^{\pi\top} \boldsymbol{D}_\mu \boldsymbol{R}$, establishing uniqueness. □

A.4 n-step counterpart of PRE and PBE

For the n-step counterparts of PRE and PBE, however, the corresponding value estimates converge to the same limit as $n \to \infty$.. Concretely, we replace the reward $\boldsymbol{R}$ with its n-step return $\boldsymbol{R}^{(n)} := \sum_{k=0}^{n} \gamma^k (\boldsymbol{P}^\pi)^k \boldsymbol{R}$, and the feature map $\boldsymbol{\Phi}$ with the n-step successor features $\boldsymbol{\Psi}^{(n)} := \sum_{k=0}^{n} \gamma^k (\boldsymbol{P}^\pi)^k \boldsymbol{\Phi}$, the natural truncation of $\boldsymbol{\Psi}_\pi$. At $n = 0$, nothing changes and we recover the PRE in (1). As n grows, however, the two estimators become equivalent.

Lemma A.1 (Equivalence in the n-step limit)**.** *For $n \in \mathbb{N}$, define the n-step projected reward solution*

$$\boldsymbol{w}_{\mathrm{PRE}}^{(n)} \;=\; \operatorname*{arg\,min}_{\boldsymbol{w}' \in \mathbb{R}^d} \big\| \boldsymbol{R}^{(n)} - \boldsymbol{\Psi}^{(n)} \boldsymbol{w}' \big\|_{\boldsymbol{D}_\mu}^2,$$

and the n-step projected Bellman solution $\boldsymbol{w}_{\mathrm{PBE}}^{(n)}$ as the fixed point of

$$\boldsymbol{\Psi}_\pi \boldsymbol{w} \;=\; \boldsymbol{\Gamma}_{\boldsymbol{\Psi}_\pi}\big(\boldsymbol{R}^{(n)} + \gamma^{n+1} (\boldsymbol{P}^\pi)^{n+1} \boldsymbol{\Psi}_\pi \boldsymbol{w}\big).$$

Suppose $\boldsymbol{\Psi}_\pi$ has full column rank. Then the two estimators are asymptotically equivalent:

$$\lim_{n\to\infty} \boldsymbol{\Psi}^{(n)} \boldsymbol{w}_{\mathrm{PRE}}^{(n)} \;=\; \lim_{n\to\infty} \boldsymbol{\Psi}_\pi \boldsymbol{w}_{\mathrm{PBE}}^{(n)},$$

and the common limit is $\boldsymbol{\Gamma}_{\boldsymbol{\Psi}_\pi} \boldsymbol{Q}^\pi$, the orthogonal projection of the true Q-function onto $\mathrm{col}(\boldsymbol{\Psi}_\pi)$.

Proof. Since $\rho(\gamma \boldsymbol{P}^\pi) \le \gamma < 1$, the partial sums converge $\boldsymbol{R}^{(n)} \to \boldsymbol{Q}^\pi = (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)^{-1} \boldsymbol{R}$ as $n \to \infty$, and $\boldsymbol{\Psi}^{(n)} = \boldsymbol{\Psi}_\pi - \gamma^{n+1} (\boldsymbol{P}^\pi)^{n+1} \boldsymbol{\Psi}_\pi \to \boldsymbol{\Psi}_\pi$ at rate γ^{n+1}.

Projected reward. The n-step estimate is the orthogonal projection $\boldsymbol{\Psi}^{(n)} \boldsymbol{w}_{\mathrm{PRE}}^{(n)} = \boldsymbol{\Gamma}_{\boldsymbol{\Psi}^{(n)}} \boldsymbol{R}^{(n)}$, where $\boldsymbol{\Gamma}_{\boldsymbol{\Psi}^{(n)}} = \boldsymbol{\Psi}^{(n)} (\boldsymbol{\Psi}^{(n)\top} \boldsymbol{D}_\mu \boldsymbol{\Psi}^{(n)})^{-1} \boldsymbol{\Psi}^{(n)\top} \boldsymbol{D}_\mu$. Since $\boldsymbol{\Psi}_\pi$ has full column rank and $\boldsymbol{\Psi}^{(n)} \to \boldsymbol{\Psi}_\pi$, the Gram matrix $\boldsymbol{\Psi}^{(n)\top} \boldsymbol{D}_\mu \boldsymbol{\Psi}^{(n)}$ is invertible for all sufficiently large n, and by continuity of matrix inversion, $\boldsymbol{\Gamma}_{\boldsymbol{\Psi}^{(n)}} \to \boldsymbol{\Gamma}_{\boldsymbol{\Psi}_\pi}$. Hence $\boldsymbol{\Psi}^{(n)} \boldsymbol{w}_{\mathrm{PRE}}^{(n)} \to \boldsymbol{\Gamma}_{\boldsymbol{\Psi}_\pi} \boldsymbol{Q}^\pi$.

Projected Bellman. The fixed-point equation yields

$$\big(\boldsymbol{\Psi}_\pi^\top \boldsymbol{D}_\mu \boldsymbol{\Psi}_\pi - \gamma^{n+1} \boldsymbol{\Psi}_\pi^\top \boldsymbol{D}_\mu (\boldsymbol{P}^\pi)^{n+1} \boldsymbol{\Psi}_\pi\big) \boldsymbol{w}_{\mathrm{PBE}}^{(n)} = \boldsymbol{\Psi}_\pi^\top \boldsymbol{D}_\mu \boldsymbol{R}^{(n)}.$$

Since $\boldsymbol{P}^\pi$ is row-stochastic, $\|(\boldsymbol{P}^\pi)^{n+1}\|_\infty \le 1$, so the second term vanishes as $n \to \infty$. The bracketed matrix therefore converges to $\boldsymbol{\Psi}_\pi^\top \boldsymbol{D}_\mu \boldsymbol{\Psi}_\pi$, which is invertible, and

$$\boldsymbol{w}_{\mathrm{PBE}}^{(n)} \to (\boldsymbol{\Psi}_\pi^\top \boldsymbol{D}_\mu \boldsymbol{\Psi}_\pi)^{-1} \boldsymbol{\Psi}_\pi^\top \boldsymbol{D}_\mu \boldsymbol{Q}^\pi, \qquad \text{i.e.} \qquad \boldsymbol{\Psi}_\pi \boldsymbol{w}_{\mathrm{PBE}}^{(n)} \to \boldsymbol{\Gamma}_{\boldsymbol{\Psi}_\pi} \boldsymbol{Q}^\pi. \qquad \square$$

B Omitted proofs in Section 4

B.1 Proof of Proposition 4.1

Proof. Subtracting $\boldsymbol{M}_\eta \boldsymbol{w}_{\mathrm{PRE}}$ from $\boldsymbol{b}_\eta$ defined in (4) and using $\boldsymbol{M}_0 \boldsymbol{w}_{\mathrm{PRE}} = \boldsymbol{b}_0$, $\boldsymbol{M}_1 \boldsymbol{w}_{\mathrm{PBE}} = \boldsymbol{b}_1$,

$$\boldsymbol{b}_\eta - \boldsymbol{M}_\eta \boldsymbol{w}_{\mathrm{PRE}} = \eta \left(\boldsymbol{b}_1 - \boldsymbol{M}_1 \boldsymbol{w}_{\mathrm{PRE}}\right) = \eta \boldsymbol{M}_1 \boldsymbol{v},$$

so that

$$\boldsymbol{M}_\eta \left(\boldsymbol{w}_\eta - \boldsymbol{w}_{\mathrm{PRE}}\right) = \eta \boldsymbol{M}_1 \boldsymbol{v}.$$

Factoring $\boldsymbol{M}_\eta = \boldsymbol{M}_0 \left((1-\eta)\boldsymbol{I} + \eta \boldsymbol{K}\right)$ and $\boldsymbol{M}_1 = \boldsymbol{M}_0 \boldsymbol{K}$, and applying both sides to the expansion $\boldsymbol{v} = \sum_{i=1}^d c_i \boldsymbol{v}_i$, the matrix $(1-\eta)\boldsymbol{I} + \eta \boldsymbol{K}$ acts on $\boldsymbol{v}_i$ as the scalar $1 - \eta + \eta \lambda_i$, which is nonzero by the nonsingularity of $\boldsymbol{M}_\eta$. Hence

$$\boldsymbol{w}_\eta - \boldsymbol{w}_{\mathrm{PRE}} = \left((1-\eta)\boldsymbol{I} + \eta \boldsymbol{K}\right)^{-1} \boldsymbol{K} \sum_{i=1}^d \eta c_i \boldsymbol{v}_i = \sum_{i=1}^d \frac{\eta \lambda_i}{1 - \eta + \eta \lambda_i} \, c_i \boldsymbol{v}_i,$$

which is (5). This completes the proof.

$\square$

Furthermore, we can prove an error bound. Let $\boldsymbol{H} := \boldsymbol{\Psi}_\pi^\top \boldsymbol{D}_\mu \boldsymbol{\Psi}_\pi$. Then

$$\|\boldsymbol{Q}^\pi - \boldsymbol{\Psi}_\pi \boldsymbol{w}_\eta\|_{\boldsymbol{D}_\mu}^2 = \|\boldsymbol{Q}^\pi - \boldsymbol{\Psi}_\pi \boldsymbol{w}^*\|_{\boldsymbol{D}_\mu}^2 + \left\| \boldsymbol{w}_{\mathrm{PRE}} - \boldsymbol{w}_* + \sum_{i=1}^d \beta_\eta(\lambda_i) c_i \boldsymbol{v}_i \right\|_{\boldsymbol{H}}^2. \tag{8}$$

The error bound can be proven in the following manner. Since $\boldsymbol{w}^*$ is the $\boldsymbol{D}_\mu$-orthogonal projection of $\boldsymbol{Q}^\pi$ onto $\mathrm{col}(\boldsymbol{\Psi}_\pi)$,

$$\|\boldsymbol{Q}^\pi - \boldsymbol{\Psi}_\pi \boldsymbol{w}\|_{\boldsymbol{D}_\mu}^2 = \mathcal{E}_* + \|\boldsymbol{w} - \boldsymbol{w}^*\|_{\boldsymbol{H}}^2.$$

The result follows from Proposition 4.1.

B.2 Proof of approximation error bounds

Throughout this subsection we use the following notations

$$\boldsymbol{A} := \boldsymbol{I} - \gamma \boldsymbol{P}^{\pi}, \qquad \boldsymbol{A}_{\eta} := \boldsymbol{I} - (1-\eta)\gamma \boldsymbol{P}^{\pi}, \qquad \boldsymbol{T}_{\eta} := \boldsymbol{A}^{*}\boldsymbol{A}_{\eta},$$

where $\boldsymbol{M}^{*} := \boldsymbol{D}_{\mu}^{-1}\boldsymbol{M}^{\top}\boldsymbol{D}_{\mu}$ denotes the adjoint with respect to the inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{D}_{\mu}} := \boldsymbol{x}^{\top}\boldsymbol{D}_{\mu}\boldsymbol{y}$. Since $\boldsymbol{\Phi} = \boldsymbol{A}\boldsymbol{\Psi}^{\pi}$, the interpolating matrix factors as $\boldsymbol{B}_{\eta}^{\pi} = \boldsymbol{A}_{\eta}\boldsymbol{\Psi}^{\pi}$.

Lemma B.1 (Unified stability and approximation bound)**.** *Suppose $\boldsymbol{\Psi}^{\pi}$ has full column rank. Let*

$$\boldsymbol{A} := \boldsymbol{I} - \gamma \boldsymbol{P}^{\pi}, \qquad \boldsymbol{A}_{\eta} := \boldsymbol{I} - (1-\eta)\gamma \boldsymbol{P}^{\pi}, \qquad \boldsymbol{T}_{\eta} := \boldsymbol{A}^{*}\boldsymbol{A}_{\eta},$$

where $\boldsymbol{M}^{*} := \boldsymbol{D}_{\mu}^{-1}\boldsymbol{M}^{\top}\boldsymbol{D}_{\mu}$ denotes the adjoint with respect to

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{D}_{\mu}} := \boldsymbol{x}^{\top}\boldsymbol{D}_{\mu}\boldsymbol{y}.$$

If there exist $c > 0$ and $\zeta \in [0, 1)$ such that

$$\|\boldsymbol{I} - c\,\boldsymbol{T}_{\eta}\|_{\boldsymbol{D}_{\mu}} \leq \zeta, \tag{9}$$

then the interpolated equation admits a unique solution $\boldsymbol{w}_{\eta}$, the mean-update matrix $\boldsymbol{M}_{\eta}^{\pi}$ has positive-definite symmetric part (and hence $-\boldsymbol{M}_{\eta}^{\pi}$ is Hurwitz), and

$$\|\boldsymbol{Q}^{\pi} - \boldsymbol{\Psi}^{\pi}\boldsymbol{w}_{\eta}\|_{\boldsymbol{D}_{\mu}}^{2} \leq \frac{1}{1-\zeta^{2}} \min_{\boldsymbol{w} \in \mathbb{R}^{d}} \|\boldsymbol{Q}^{\pi} - \boldsymbol{\Psi}^{\pi}\boldsymbol{w}\|_{\boldsymbol{D}_{\mu}}^{2}. \tag{10}$$

Proof. Write

$$\boldsymbol{\Psi} := \boldsymbol{\Psi}^{\pi}, \qquad \boldsymbol{D} := \boldsymbol{D}_{\mu},$$

and let $\boldsymbol{\Gamma}_{\boldsymbol{\Psi}}$ denote the $\boldsymbol{D}$-orthogonal projection onto $\mathrm{col}(\boldsymbol{\Psi})$.

Since $\boldsymbol{\Phi} = \boldsymbol{A}\boldsymbol{\Psi}$, the interpolating matrix satisfies

$$\begin{aligned} \boldsymbol{B}_{\eta}^{\pi} &= (1-\eta)\boldsymbol{\Phi} + \eta\boldsymbol{\Psi} \\ &= \big[(1-\eta)\boldsymbol{A} + \eta\boldsymbol{I}\big]\boldsymbol{\Psi} \\ &= \boldsymbol{A}_{\eta}\boldsymbol{\Psi}. \end{aligned}$$

Hence the mean-update matrix can be written as

$$\begin{aligned} \boldsymbol{M}_{\eta}^{\pi} &:= \boldsymbol{B}_{\eta}^{\pi\top}\boldsymbol{D}\boldsymbol{\Phi} \\ &= \boldsymbol{\Psi}^{\top}\boldsymbol{A}_{\eta}^{\top}\boldsymbol{D}\boldsymbol{A}\boldsymbol{\Psi} \\ &= (\boldsymbol{T}_{\eta}\boldsymbol{\Psi})^{\top}\boldsymbol{D}\boldsymbol{\Psi}. \end{aligned} \tag{11}$$

We first show that the hypothesis implies coercivity of $\boldsymbol{T}_{\eta}$. For any $\boldsymbol{x} \neq \boldsymbol{0}$,

$$\begin{aligned} c\langle \boldsymbol{x}, \boldsymbol{T}_{\eta}\boldsymbol{x} \rangle_{\boldsymbol{D}} &= \|\boldsymbol{x}\|_{\boldsymbol{D}}^{2} - \langle \boldsymbol{x}, (\boldsymbol{I} - c\boldsymbol{T}_{\eta})\boldsymbol{x} \rangle_{\boldsymbol{D}} \\ &\geq \|\boldsymbol{x}\|_{\boldsymbol{D}}^{2} - \|\boldsymbol{x}\|_{\boldsymbol{D}}\|(\boldsymbol{I} - c\boldsymbol{T}_{\eta})\boldsymbol{x}\|_{\boldsymbol{D}} \\ &\geq (1-\zeta)\|\boldsymbol{x}\|_{\boldsymbol{D}}^{2}. \end{aligned}$$

Therefore,

$$\langle \boldsymbol{x}, \boldsymbol{T}_{\eta}\boldsymbol{x} \rangle_{\boldsymbol{D}} \geq \frac{1-\zeta}{c}\|\boldsymbol{x}\|_{\boldsymbol{D}}^{2} > 0. \tag{12}$$

For any $\boldsymbol{x} \neq \boldsymbol{0}$, let $\boldsymbol{y} := \boldsymbol{\Psi}\boldsymbol{x}$. Since $\boldsymbol{\Psi}$ has full column rank, $\boldsymbol{y} \neq \boldsymbol{0}$. Using equation 11,

$$\begin{aligned} \boldsymbol{x}^{\top}\boldsymbol{M}_{\eta}^{\pi}\boldsymbol{x} &= \langle \boldsymbol{y}, \boldsymbol{T}_{\eta}\boldsymbol{y} \rangle_{\boldsymbol{D}} \\ &> 0. \end{aligned}$$

Hence the symmetric part of $\boldsymbol{M}_{\eta}^{\pi}$ is positive definite. In particular, $\boldsymbol{M}_{\eta}^{\pi}$ is invertible, every eigenvalue of $\boldsymbol{M}_{\eta}^{\pi}$ has positive real part, and $-\boldsymbol{M}_{\eta}^{\pi}$ is Hurwitz. Thus, the interpolated equation admits a unique solution $\boldsymbol{w}_{\eta}$.

Now define

$$\boldsymbol{e}_\eta := \boldsymbol{Q}^\pi - \boldsymbol{\Psi}\boldsymbol{w}_\eta.$$

Since $\boldsymbol{R} = \boldsymbol{A}\boldsymbol{Q}^\pi$, we have

$$\boldsymbol{R} - \boldsymbol{\Phi}\boldsymbol{w}_\eta = \boldsymbol{A}\boldsymbol{e}_\eta.$$

Therefore, the interpolated equation

$$\boldsymbol{B}_\eta^{\pi\top}\boldsymbol{D}(\boldsymbol{R} - \boldsymbol{\Phi}\boldsymbol{w}_\eta) = \boldsymbol{0}$$

becomes

$$\boldsymbol{\Psi}^\top \boldsymbol{A}_\eta^\top \boldsymbol{D}\boldsymbol{A}\boldsymbol{e}_\eta = \boldsymbol{0}.$$

Using

$$\boldsymbol{A}_\eta^\top \boldsymbol{D} = \boldsymbol{D}\boldsymbol{A}_\eta^*, \quad \boldsymbol{T}_\eta^* = \boldsymbol{A}_\eta^*\boldsymbol{A},$$

we obtain

$$\boldsymbol{\Psi}^\top \boldsymbol{D}\boldsymbol{T}_\eta^*\boldsymbol{e}_\eta = \boldsymbol{0} \quad \Longleftrightarrow \quad \boldsymbol{\Gamma}_{\boldsymbol{\Psi}}\boldsymbol{T}_\eta^*\boldsymbol{e}_\eta = \boldsymbol{0}. \tag{13}$$

It follows from (13) that

$$\boldsymbol{\Gamma}_{\boldsymbol{\Psi}}\boldsymbol{e}_\eta = \boldsymbol{\Gamma}_{\boldsymbol{\Psi}}(\boldsymbol{I} - c\boldsymbol{T}_\eta^*)\boldsymbol{e}_\eta.$$

Since $\boldsymbol{\Gamma}_{\boldsymbol{\Psi}}$ is a $\boldsymbol{D}$-orthogonal projection, it is non-expansive. Moreover, $\|\boldsymbol{M}^*\|_{\boldsymbol{D}} = \|\boldsymbol{M}\|_{\boldsymbol{D}}$. Hence

$$\begin{aligned}
\|\boldsymbol{\Gamma}_{\boldsymbol{\Psi}}\boldsymbol{e}_\eta\|_{\boldsymbol{D}} &\le \|\boldsymbol{I} - c\boldsymbol{T}_\eta^*\|_{\boldsymbol{D}}\|\boldsymbol{e}_\eta\|_{\boldsymbol{D}} &(14)\\
&= \|\boldsymbol{I} - c\boldsymbol{T}_\eta\|_{\boldsymbol{D}}\|\boldsymbol{e}_\eta\|_{\boldsymbol{D}} &(15)\\
&\le \zeta\|\boldsymbol{e}_\eta\|_{\boldsymbol{D}}. &(16)
\end{aligned}$$

Since $\boldsymbol{\Psi}\boldsymbol{w}_\eta \in \mathrm{col}(\boldsymbol{\Psi})$,

$$(\boldsymbol{I} - \boldsymbol{\Gamma}_{\boldsymbol{\Psi}})\boldsymbol{e}_\eta = (\boldsymbol{I} - \boldsymbol{\Gamma}_{\boldsymbol{\Psi}})\boldsymbol{Q}^\pi.$$

The two components $\boldsymbol{\Gamma}_{\boldsymbol{\Psi}}\boldsymbol{e}_\eta$ and $(\boldsymbol{I} - \boldsymbol{\Gamma}_{\boldsymbol{\Psi}})\boldsymbol{e}_\eta$ are $\boldsymbol{D}$-orthogonal, so

$$\begin{aligned}
\|\boldsymbol{e}_\eta\|_{\boldsymbol{D}}^2 &= \|(\boldsymbol{I} - \boldsymbol{\Gamma}_{\boldsymbol{\Psi}})\boldsymbol{Q}^\pi\|_{\boldsymbol{D}}^2 + \|\boldsymbol{\Gamma}_{\boldsymbol{\Psi}}\boldsymbol{e}_\eta\|_{\boldsymbol{D}}^2\\
&\le \|(\boldsymbol{I} - \boldsymbol{\Gamma}_{\boldsymbol{\Psi}})\boldsymbol{Q}^\pi\|_{\boldsymbol{D}}^2 + \zeta^2\|\boldsymbol{e}_\eta\|_{\boldsymbol{D}}^2.
\end{aligned}$$

Since $\zeta < 1$, rearranging gives

$$\|\boldsymbol{e}_\eta\|_{\boldsymbol{D}}^2 \le \frac{1}{1-\zeta^2}\|(\boldsymbol{I} - \boldsymbol{\Gamma}_{\boldsymbol{\Psi}})\boldsymbol{Q}^\pi\|_{\boldsymbol{D}}^2.$$

Finally,

$$\|(\boldsymbol{I} - \boldsymbol{\Gamma}_{\boldsymbol{\Psi}})\boldsymbol{Q}^\pi\|_{\boldsymbol{D}}^2 = \min_{\boldsymbol{w}\in\mathbb{R}^d}\|\boldsymbol{Q}^\pi - \boldsymbol{\Psi}\boldsymbol{w}\|_{\boldsymbol{D}}^2,$$

which proves (10). □

Proof of Proposition 4.2(i). Write

$$\boldsymbol{P} := \boldsymbol{P}^\pi, \qquad \boldsymbol{D} := \boldsymbol{D}_\mu.$$

By Lemma B.1, it suffices to exhibit $c > 0$ and $\zeta \in [0, 1)$ satisfying (9).

Since $\boldsymbol{\mu}^\top\boldsymbol{P} = \boldsymbol{\mu}^\top$ and $\boldsymbol{P}$ is row stochastic, Jensen's inequality gives, for any $\boldsymbol{x}$,

$$\begin{aligned}
\|\boldsymbol{P}\boldsymbol{x}\|_{\boldsymbol{D}}^2 &= \sum_i [\boldsymbol{\mu}]_i \left(\sum_j [\boldsymbol{P}]_{ij}[\boldsymbol{x}]_j\right)^2\\
&\le \sum_i [\boldsymbol{\mu}]_i \sum_j [\boldsymbol{P}]_{ij}[\boldsymbol{x}]_j^2\\
&= \sum_j \left(\sum_i [\boldsymbol{\mu}]_i[\boldsymbol{P}]_{ij}\right)[\boldsymbol{x}]_j^2\\
&= \|\boldsymbol{x}\|_{\boldsymbol{D}}^2.
\end{aligned}$$

Hence

$$\|\boldsymbol{P}\|_{\boldsymbol{D}} \le 1, \quad \|\boldsymbol{P}^*\|_{\boldsymbol{D}} = \|\boldsymbol{P}\|_{\boldsymbol{D}} \le 1.$$

Define

$$c_\eta := 1 + (1-\eta)\gamma^2, \qquad E_\eta := \gamma(2-\eta),$$

and let

$$\boldsymbol{H} := \boldsymbol{P}^* + \boldsymbol{P} + \gamma(\boldsymbol{I} - \boldsymbol{P}^*\boldsymbol{P}).$$

Expanding

$$\boldsymbol{T}_\eta = (\boldsymbol{I} - \gamma\boldsymbol{P}^*)\big(\boldsymbol{I} - (1-\eta)\gamma\boldsymbol{P}\big)$$

gives

$$\begin{aligned} c_\eta\boldsymbol{I} - \boldsymbol{T}_\eta &= \gamma\left[\boldsymbol{P}^* + (1-\eta)\boldsymbol{P} + (1-\eta)\gamma(\boldsymbol{I} - \boldsymbol{P}^*\boldsymbol{P})\right] \\ &= \gamma\left[\eta\boldsymbol{P}^* + (1-\eta)\boldsymbol{H}\right]. \end{aligned}$$

For every $\boldsymbol{x}$,

$$\begin{aligned} \langle \boldsymbol{x}, (2\boldsymbol{I} - \boldsymbol{H})\boldsymbol{x}\rangle_{\boldsymbol{D}} &= \|(\boldsymbol{I} - \boldsymbol{P})\boldsymbol{x}\|_{\boldsymbol{D}}^2 \\ &\quad + (1-\gamma)\left(\|\boldsymbol{x}\|_{\boldsymbol{D}}^2 - \|\boldsymbol{P}\boldsymbol{x}\|_{\boldsymbol{D}}^2\right) \\ &\ge 0, \end{aligned}$$

and

$$\begin{aligned} \langle \boldsymbol{x}, (2\boldsymbol{I} + \boldsymbol{H})\boldsymbol{x}\rangle_{\boldsymbol{D}} &= \|(\boldsymbol{I} + \boldsymbol{P})\boldsymbol{x}\|_{\boldsymbol{D}}^2 \\ &\quad + (1+\gamma)\left(\|\boldsymbol{x}\|_{\boldsymbol{D}}^2 - \|\boldsymbol{P}\boldsymbol{x}\|_{\boldsymbol{D}}^2\right) \\ &\ge 0. \end{aligned}$$

Since $\boldsymbol{H}$ is self-adjoint with respect to $\langle\cdot,\cdot\rangle_{\boldsymbol{D}}$, it follows that

$$\|\boldsymbol{H}\|_{\boldsymbol{D}} \le 2.$$

Therefore,

$$\begin{aligned} \|\boldsymbol{T}_\eta - c_\eta\boldsymbol{I}\|_{\boldsymbol{D}} &\le \gamma\left(\eta\|\boldsymbol{P}^*\|_{\boldsymbol{D}} + (1-\eta)\|\boldsymbol{H}\|_{\boldsymbol{D}}\right) \\ &\le \gamma\left(\eta + 2(1-\eta)\right) \\ &= E_\eta. \end{aligned}$$

Moreover,

$$\begin{aligned} c_\eta - E_\eta &= 1 + (1-\eta)\gamma^2 - \gamma(2-\eta) \\ &= (1-\gamma)\big(1 - (1-\eta)\gamma\big) \\ &> 0. \end{aligned}$$

Hence, defining

$$\beta_\eta := \frac{E_\eta}{c_\eta},$$

we have $\beta_\eta < 1$. Taking

$$c := \frac{1}{c_\eta}, \quad \zeta := \beta_\eta,$$

gives

$$\begin{aligned} \|\boldsymbol{I} - c\boldsymbol{T}_\eta\|_{\boldsymbol{D}} &= \frac{1}{c_\eta}\|c_\eta\boldsymbol{I} - \boldsymbol{T}_\eta\|_{\boldsymbol{D}} \\ &\le \beta_\eta. \end{aligned}$$

Thus, Lemma B.1 yields

$$\|\boldsymbol{Q}^\pi - \boldsymbol{\Psi}^\pi \boldsymbol{w}_\eta\|_{\boldsymbol{D}_\mu}^2 \le \kappa_\eta^{\mathrm{stat}} \min_{\boldsymbol{w}\in\mathbb{R}^d} \|\boldsymbol{Q}^\pi - \boldsymbol{\Psi}^\pi \boldsymbol{w}\|_{\boldsymbol{D}_\mu}^2,$$

where

$$\begin{aligned}\kappa_\eta^{\mathrm{stat}} &:= \frac{1}{1-\beta_\eta^2} \\ &= \frac{c_\eta^2}{c_\eta^2 - E_\eta^2} \\ &= \frac{\left[1+(1-\eta)\gamma^2\right]^2}{(1-\gamma^2)\left[1-(1-\eta)^2\gamma^2\right]}.\end{aligned}$$

Here we used

$$\begin{aligned}c_\eta^2 - E_\eta^2 &= \left(1+(1-\eta)\gamma^2\right)^2 - \gamma^2(2-\eta)^2 \\ &= (1-\gamma^2)\left[1-(1-\eta)^2\gamma^2\right].\end{aligned}$$

Finally,

$$\begin{aligned}\frac{d}{d\eta}\log\kappa_\eta^{\mathrm{stat}} &= -\frac{2\gamma^2}{1+(1-\eta)\gamma^2} - \frac{2(1-\eta)\gamma^2}{1-(1-\eta)^2\gamma^2} \\ &< 0\end{aligned}$$

for $\gamma \in (0,1)$. Hence, $\kappa_\eta^{\mathrm{stat}}$ is strictly decreasing on $[0,1]$. □

Proof of Proposition 4.2(ii). Write

$$\boldsymbol{A} := \boldsymbol{I} - \gamma \boldsymbol{P}^\pi.$$

Reversibility means

$$\boldsymbol{D}_\mu \boldsymbol{P}^\pi = \boldsymbol{P}^{\pi\top} \boldsymbol{D}_\mu,$$

or equivalently

$$(\boldsymbol{P}^\pi)^* = \boldsymbol{P}^\pi.$$

Summing the detailed-balance relation over rows gives $\boldsymbol{\mu}^\top \boldsymbol{P}^\pi = \boldsymbol{\mu}^\top$, so the stationary-case assumptions also hold.

Under reversibility, $\boldsymbol{A}^* = \boldsymbol{A}$, and

$$\begin{aligned}\boldsymbol{A}_\eta &= \boldsymbol{I} - (1-\eta)\gamma\boldsymbol{P}^\pi \\ &= \eta\boldsymbol{I} + (1-\eta)\boldsymbol{A}.\end{aligned}$$

Since $\boldsymbol{A}_\eta$ is a polynomial in $\boldsymbol{A}$, it commutes with $\boldsymbol{A}$. Therefore,

$$\begin{aligned}\boldsymbol{T}_\eta &= \boldsymbol{A}^*\boldsymbol{A}_\eta \\ &= \boldsymbol{A}_\eta\boldsymbol{A} \\ &= (1-\eta)\boldsymbol{A}^2 + \eta\boldsymbol{A} \\ &= g_\eta(\boldsymbol{A}),\end{aligned}$$

where

$$g_\eta(\nu) := (1-\eta)\nu^2 + \eta\nu.$$

Hence, $\boldsymbol{T}_\eta$ is self-adjoint with respect to $\langle\cdot,\cdot\rangle_{\boldsymbol{D}_\mu}$.

Let

$$\nu_{\min} := \min \mathrm{spec}(\boldsymbol{A}), \quad \nu_{\max} := \max \mathrm{spec}(\boldsymbol{A})$$

where $\nu_{\min} = 1 - \gamma$. Since

$$1 - \gamma \le \nu_{\min} \le \nu_{\max} \le 1 + \gamma$$

and g_η is strictly increasing on $(0, \infty)$,

$$\lambda_{\min}(\boldsymbol{T}_\eta) = g_\eta(\nu_{\min}), \quad \lambda_{\max}(\boldsymbol{T}_\eta) = g_\eta(\nu_{\max}).$$

Define

$$g_{\min} := g_\eta(\nu_{\min}), \quad g_{\max} := g_\eta(\nu_{\max}), \quad \xi_\eta := \frac{g_{\max}}{g_{\min}}.$$

Both $g_{\min}$ and $g_{\max}$ are positive.

Choose

$$c := \frac{2}{g_{\min} + g_{\max}}.$$

Since $\boldsymbol{I} - c\boldsymbol{T}_\eta$ is self-adjoint,

$$\begin{aligned}
\|\boldsymbol{I} - c\boldsymbol{T}_\eta\|_{\boldsymbol{D}_\mu} &= \max_{\lambda \in \mathrm{spec}(\boldsymbol{T}_\eta)} |1 - c\lambda| \\
&= \max\{|1 - cg_{\min}|, |1 - cg_{\max}|\} \\
&= \frac{g_{\max} - g_{\min}}{g_{\max} + g_{\min}} \\
&= \frac{\xi_\eta - 1}{\xi_\eta + 1} =: \zeta_\eta < 1.
\end{aligned}$$

Lemma B.1 therefore gives

$$\|\boldsymbol{Q}^\pi - \boldsymbol{\Psi}^\pi \boldsymbol{w}_\eta\|^2_{\boldsymbol{D}_\mu} \le \kappa_\eta \min_{\boldsymbol{w} \in \mathbb{R}^d} \|\boldsymbol{Q}^\pi - \boldsymbol{\Psi}^\pi \boldsymbol{w}\|^2_{\boldsymbol{D}_\mu},$$

where

$$\begin{aligned}
\kappa_\eta &:= \frac{1}{1 - \zeta_\eta^2} \\
&= \frac{(\xi_\eta + 1)^2}{4\xi_\eta}.
\end{aligned}$$

To establish monotonicity, note that the map

$$f(u) := \frac{(u+1)^2}{4u}$$

is nondecreasing on $[1, \infty)$, since

$$f'(u) = \frac{(u+1)(u-1)}{4u^2} \ge 0.$$

Moreover,

$$\begin{aligned}
\frac{d}{d\eta}\xi_\eta &= \frac{\nu_{\max}\nu_{\min}(\nu_{\min} - \nu_{\max})}{g_\eta(\nu_{\min})^2} \\
&\le 0,
\end{aligned}$$

with strict inequality whenever $\nu_{\min} < \nu_{\max}$. Therefore, $\kappa_\eta = f(\xi_\eta)$ is strictly decreasing in η whenever $\nu_{\min} < \nu_{\max}$, and is constant when $\nu_{\min} = \nu_{\max}$. Plugging in $\nu_{\min} = 1 - \gamma$ gives the desired result.

Finally, we compare this bound with the stationary bound. Let

$$\lambda := 1 - \eta.$$

For the extremal spectral interval

$$\nu_{\min} = 1 - \gamma, \quad \nu_{\max} = 1 + \gamma,$$

we have

$$g_\eta(1 \pm \gamma) = (1 \pm \gamma)(1 \pm \lambda\gamma).$$

Hence,

$$g_{\max} + g_{\min} = 2\left(1 + \lambda\gamma^2\right) = 2c_\eta,$$
$$g_{\max} - g_{\min} = 2\gamma(1 + \lambda) = 2E_\eta.$$

Therefore,

$$\zeta_\eta = \beta_\eta, \quad \kappa_\eta = \kappa_\eta^{\text{stat}}.$$

For a general reversible chain,

$$1 - \gamma \le \nu_{\min} \le \nu_{\max} \le 1 + \gamma.$$

Since g_η is strictly increasing and positive,

$$\xi_\eta = \frac{g_\eta(\nu_{\max})}{g_\eta(\nu_{\min})} \le \frac{g_\eta(1+\gamma)}{g_\eta(1-\gamma)}.$$

Because $f(u) = (u+1)^2/(4u)$ is nondecreasing on $[1, \infty)$,

$$\kappa_\eta \le \kappa_\eta^{\text{stat}}.$$

For $\gamma \in (0, 1)$, equality holds if and only if

$$\nu_{\min} = 1 - \gamma, \quad \nu_{\max} = 1 + \gamma.$$

□

B.3 Proof of Lemma 4.4

Proof. We have

$$\boldsymbol{M}_\eta^\pi = (1-\eta)\boldsymbol{\Phi}^\top \boldsymbol{D}_\mu \boldsymbol{\Phi} + \eta \boldsymbol{\Psi}_\pi^\top \boldsymbol{D}_\mu (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{\Psi}_\pi, \qquad \eta \in [0, 1]. \tag{17}$$

Since $\boldsymbol{\Phi} = (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{\Psi}_\pi$, the quadratic form associated with $\boldsymbol{M}_\eta^\pi$ can be written as

$$\boldsymbol{x}^\top \boldsymbol{M}_\eta^\pi \boldsymbol{x} = (1-\eta)||(\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{y}||_{\boldsymbol{D}_\mu}^2 + \eta \boldsymbol{y}^\top \boldsymbol{D}_\mu (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{y}, \tag{18}$$

where $\boldsymbol{y} = \boldsymbol{\Psi}_\pi \boldsymbol{x}$. If $\boldsymbol{y}^\top \boldsymbol{D}_\mu (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{y} \ge 0$, then both terms in (18) are nonnegative. It therefore suffices to consider vectors satisfying $\boldsymbol{y}^\top \boldsymbol{D}_\mu (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{y} < 0$. For such a vector, the condition $\boldsymbol{x}^\top \boldsymbol{M}_\eta^\pi \boldsymbol{x} \ge 0$ is equivalent to

$$(1-\eta)||(\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{y}||_{\boldsymbol{D}_\mu}^2 \ge -\eta \boldsymbol{y}^\top \boldsymbol{D}_\mu (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{y}.$$

Equivalently,

$$\frac{\eta}{1-\eta} \le \frac{||(\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{y}||_{\boldsymbol{D}_\mu}^2}{-\boldsymbol{y}^\top \boldsymbol{D}_\mu (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{y}}.$$

Taking the infimum over all such $\boldsymbol{y}$ gives $\frac{\eta}{1-\eta} \le c^*$, which is equivalent to $\eta \le \frac{c^*}{1+c^*}$.

For the strict claim, let $\boldsymbol{x} \neq 0$, so $\boldsymbol{y} = \boldsymbol{\Psi}_\pi \boldsymbol{x} \neq 0$ by the full-rank assumption. If $\boldsymbol{y}^\top \boldsymbol{D}_\mu (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{y} \ge 0$, then $\boldsymbol{x}^\top \boldsymbol{M}_\eta^\pi \boldsymbol{x} \ge (1-\eta)\|(\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{y}\|_{\boldsymbol{D}_\mu}^2 > 0$, since $\eta < 1$ and $\boldsymbol{I} - \gamma \boldsymbol{P}^\pi$ is invertible. Otherwise $\eta/(1-\eta) < c^* \le \|(\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{y}\|_{\boldsymbol{D}_\mu}^2 / (-\boldsymbol{y}^\top \boldsymbol{D}_\mu (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{y})$, and the strict inequality gives $\boldsymbol{x}^\top \boldsymbol{M}_\eta^\pi \boldsymbol{x} > 0$.

□

B.4 Proof of Theorem 4.5

Proof. Conditioned on $\mathcal{F}_k = \sigma(\boldsymbol{w}_0, \ldots, \boldsymbol{w}_k)$ where $\sigma(\cdot)$ denotes the σ-algebra, using $(\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)\boldsymbol{\Psi}_\pi = \boldsymbol{\Phi}$, the expected update is $\boldsymbol{B}_\eta^{\pi\top} \boldsymbol{D}_\mu (\boldsymbol{R} - \boldsymbol{\Phi}\boldsymbol{w}_k) = \boldsymbol{b}_\eta - \boldsymbol{M}_\eta^\pi \boldsymbol{w}_k$ with $\boldsymbol{b}_\eta := \boldsymbol{B}_\eta^{\pi\top} \boldsymbol{D}_\mu \boldsymbol{R}$, so the iteration is the linear stochastic approximation $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \alpha_k(\boldsymbol{b}_\eta - \boldsymbol{M}_\eta^\pi \boldsymbol{w}_k + \boldsymbol{\xi}_k)$ with martingale-difference noise satisfying $\mathbb{E}[\|\boldsymbol{\xi}_k\|^2 \mid \mathcal{F}_k] \le C(1 + \|\boldsymbol{w}_k\|^2)$ for some constant C. By Lemma 4.4, $\boldsymbol{x}^\top \boldsymbol{M}_\eta^\pi \boldsymbol{x} > 0$ for $\boldsymbol{x} \neq 0$, hence every eigenvalue of $\boldsymbol{M}_\eta^\pi$ has positive real part and $-\boldsymbol{M}_\eta^\pi$ is Hurwitz. The ODE $\dot{\boldsymbol{w}}_t = \boldsymbol{b}_\eta - \boldsymbol{M}_\eta^\pi \boldsymbol{w}_t$ therefore has $\boldsymbol{w}_\eta$ as its globally asymptotically stable equilibrium, and the claim follows from Theorem 2.2 of Borkar and Meyn (2000). □

B.5 Interpretation under reversible chain case

Under the reversibility assumption, the best possible solution can be re-written as

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w} \in \mathbb{R}^d} \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{R}\|^2_{\boldsymbol{D}_\mu (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)^{-2}}.$$

Meanwhile, the interpolated equation in (4) gives the solution

$$\boldsymbol{w}_\eta = \arg\min_{\boldsymbol{w} \in \mathbb{R}^d} \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{R}\|^2_{\boldsymbol{D}_\mu \left((1-\eta)\boldsymbol{I} + \eta (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)^{-1}\right)}. \tag{19}$$

At $\eta = 0$, the weighting matrix in (19) reduces to $\boldsymbol{D}_\mu$, and hence

$$\boldsymbol{w}_{\mathrm{PRE}} = \arg\min_{\boldsymbol{w} \in \mathbb{R}^d} \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{R}\|^2_{\boldsymbol{D}_\mu}. \tag{20}$$

At $\eta = 1$, it reduces to $\boldsymbol{D}_\mu (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)^{-1}$, yielding

$$\boldsymbol{w}_{\mathrm{PBE}} = \arg\min_{\boldsymbol{w} \in \mathbb{R}^d} \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{R}\|^2_{\boldsymbol{D}_\mu (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)^{-1}}. \tag{21}$$

Thus, the interpolated objective continuously connects the reward-regression and PBE weighting matrices.

C Examples

To proceed, let us introduce matrix and vector notations. Let $\boldsymbol{P} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}|}$ denote the transition matrix such that the $[\boldsymbol{P}]_{(s-1)|\mathcal{A}|+a,s'} = \mathcal{P}(s' \mid s, a)$, $\boldsymbol{\Pi}_\pi \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}||\mathcal{A}|}$ denote the policy matrix of which the row corresponds to $(\boldsymbol{e}_s \otimes \vec{\pi}(\cdot \mid s))^\top$ where $\boldsymbol{e}_s \in \mathbb{R}^{|\mathcal{S}|}$ is a unit vector with s-the element being only one, $\otimes$ is the Kronecker product and $\vec{\pi}(\cdot \mid s) \in \mathbb{R}^{|\mathcal{A}|}$ is a vectorized form of the policy. For simplicity, we denote $\boldsymbol{P}^\pi = \boldsymbol{P}\boldsymbol{\Pi}_\pi$. The reward vector is defined by $\boldsymbol{R} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ such that $[\boldsymbol{R}]_{(s-1)|\mathcal{A}|+a} = \mathbb{E}[r(s, a, s') \mid s, a]$. Let $\boldsymbol{D}_\mu$ be a diagonal matrix with diagonal entries $\mu(s, a)$ and $\sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} \mu(s, a) = 1$. Throughout the paper, let us assume that $\mu(s, a) > 0$.

Example C.1 (Failure of reward regression)**.** *Let us consider the example in Figure 1. There are two actions, at state s_0, going to Bakery or Cafe, and there is only one action at other states s_1, s_2 and s_3. The feature, transition matrix, and the reward vector is defined as follows where redundant state-action pairs are omitted, that is $|\mathcal{S}| = 4$ and $|\mathcal{A}_0| = 2$ where $\mathcal{A}_i$ stands for the number of actions possible at state s_i and $|\mathcal{A}_i| = 1$ for $i \in \{1, 2, \ldots, 3\}$. The reward is one if the next state is s_3 otherwise zero:*

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \ \boldsymbol{P} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0.9 & 0.1 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \ \boldsymbol{\Pi}^\pi = \begin{bmatrix} \epsilon & 1-\epsilon & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \ \boldsymbol{R} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0.1 \\ 0 \end{bmatrix}$$

and we use $\gamma = 0.9$. Considering an ϵ-greedy behavior policy with $\epsilon = 0.1$ and its stationary distribution, solving PRE gives $\boldsymbol{w}_{\mathrm{PRE}} \approx \begin{bmatrix} 0 \\ 0 \\ 0.5 \end{bmatrix}$ and solving PBE gives $\boldsymbol{w}_{\mathrm{PBE}} \approx \begin{bmatrix} 0.5 \\ 0.5 \\ 0.2 \end{bmatrix}$. This leads

to the value function estimate $\boldsymbol{\Psi}_\pi \boldsymbol{w}_{\mathrm{PRE}}(s_0, \mathrm{Cafe}) \approx 2.08$, $\boldsymbol{\Psi}_\pi \boldsymbol{w}_{\mathrm{PRE}}(s_0, \mathrm{Bakery}) \approx 3.34$ *and* $\boldsymbol{\Psi}_\pi(s_0, \mathrm{Cafe}) \boldsymbol{w}_{\mathrm{PBE}} \approx 2.62$, $\boldsymbol{\Psi}_\pi \boldsymbol{w}_{\mathrm{PBE}}(s_0, \mathrm{Bakery}) \approx 2.61$. *The agent should select going to cafe but it selects to going to bakery if we follow the PRE solution whereas following PBE solution, we follow going to cafe.*

D PSEUDO-CODE

Algorithm 3 SF-Bellman (Universal value function scheme)

1: $\eta \in [0,1]$, $\gamma \in (0,1)$, mini-batch size N, target-update period C, step sizes $\{\alpha_k^w, \beta_k^\psi\}_{k \geq 0}$, replay buffer $\mathcal{D}_{\mathrm{old}}$ and $\mathcal{D}_{\mathrm{fresh}}$, networks $\phi_{\boldsymbol{u}_0}(\cdot,\cdot) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ and $\psi_{\boldsymbol{v}_0}(\cdot,\cdot) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$, and task vector $\boldsymbol{w}_0 \in \mathbb{R}^d$, task vector update period U

2: **for** each iteration k **do**

3: Select a_k using ϵ-greedy policy on $\boldsymbol{\psi}_{\boldsymbol{u}_k}(s_k, \cdot, \boldsymbol{w}_k)^\top \boldsymbol{w}_k$

4: Observe $s'_k \sim \mathcal{P}(\cdot \mid s_k, a_k)$ and reward r_k

5: Store $(s_k, a_k, r_k, s'_k, \boldsymbol{w}_k)$ in $\mathcal{D}^{\mathrm{old}}$ and $\mathcal{D}^{\mathrm{fresh}}$

6: Sample $B_k^{\mathrm{old}} = \{(s_{k_j^{\mathrm{old}}}, a_{k_j^{\mathrm{old}}}, r_{k_j^{\mathrm{old}}}, s'_{k_j^{\mathrm{old}}}, \boldsymbol{w}_{k_j^{\mathrm{old}}})\}_{j=1}^N$ from $\mathcal{D}^{\mathrm{old}}$

7: Compute the ψ network loss:

$$L_\psi = \frac{1}{2} \sum_{j=1}^{N} \left(\left(r_j + \gamma \mathrm{SG}\big(\max_{a \in \mathcal{A}} \psi_{\boldsymbol{u}_k}(s'_{k_j^{\mathrm{old}}}, a, \boldsymbol{w}_{k_j^{\mathrm{old}}})^\top \boldsymbol{w}_{k_j^{\mathrm{old}}}\big) - \psi_{\boldsymbol{u}_k}(s_{k_j^{\mathrm{old}}}, a_{k_j^{\mathrm{old}}}, \boldsymbol{w}_{k_j^{\mathrm{old}}})^\top \boldsymbol{w}_{k_j^{\mathrm{old}}} \right) \right)^2.$$

8: Sample $B_k^{\mathrm{fresh}} = \{(s_{k_j^{\mathrm{fresh}}}, a_{k_j^{\mathrm{fresh}}}, r_{k_j^{\mathrm{fresh}}}, s'_{k_j^{\mathrm{fresh}}})\}_{j=1}^N$ from $\mathcal{D}^{\mathrm{fresh}}$

9: Compute the task vector loss:

$$L_{\mathrm{PRE}} = \sum_{j=1}^{N} \left(\left(r_{k_j^{\mathrm{fresh}}} - \phi_{\boldsymbol{u}_k}(s_{k_j^{\mathrm{fresh}}}, a_{k_j^{\mathrm{fresh}}}, \boldsymbol{w}_k)^\top \boldsymbol{w}_k \right) \right)^2$$

$$L_{\mathrm{PBE}} = \sum_{l=1}^{N} \left(\left(r_l + \gamma \mathrm{SG}\big(\max_{a \in \mathcal{A}} \psi_{\bar{\boldsymbol{u}}_k}(s'_l, a, \bar{\boldsymbol{w}}_k)^\top \bar{\boldsymbol{w}}_k\big) - \psi_{\boldsymbol{u}_k}(s_l, a_l, \boldsymbol{w}_k)^\top \boldsymbol{w}_k \right) \right)^2$$

10: Update $\boldsymbol{v}_k$ by takin gradient on L_ψ.

11: Every U steps, update

$$\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k + \alpha_k^w \nabla_{\boldsymbol{w}} \left(\frac{1-\eta}{2} L_{\mathrm{PRE}} + \frac{\eta}{2} L_{\mathrm{PBE}} \right).$$

12: Periodically copy $\boldsymbol{w} \to \bar{\boldsymbol{w}}$ and $\boldsymbol{\theta} \to \bar{\boldsymbol{\theta}}$ (frequency C)

13: **end for**

Algorithm 4 Actor–Critic SF-Bellman (Universal value function scheme)

Input: $\eta \in [0,1]$, $\gamma \in (0,1)$, minibatch size N, replay buffer $\mathcal{D}_{\text{old}}$ and $\mathcal{D}_{\text{fresh}}$, target-update period C, step sizes $\{\alpha_k, \beta_k^{\psi}, \beta_k^{\pi}\}_{k\geq 0}$, feature network $\phi_{\boldsymbol{u}_0} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}^d$, successor-feature network $\psi_{\boldsymbol{v}_0} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}^d$, actor $\pi_{\boldsymbol{\theta}_0} : \mathcal{S} \times \mathbb{R}^d \to \mathcal{A}$, task vector $\boldsymbol{w}_0 \in \mathbb{R}^d$, and exploration noise ϵ_k, task vector update period U

Initialize $\bar{\boldsymbol{u}}_0 \leftarrow \boldsymbol{u}_0$, $\bar{\boldsymbol{v}}_0 \leftarrow \boldsymbol{v}_0$, $\bar{\boldsymbol{w}}_0 \leftarrow \boldsymbol{w}_0$, and $\bar{\boldsymbol{\theta}}_0 \leftarrow \boldsymbol{\theta}_0$

for $k = 0, 1, 2, \ldots$ **do**

  Select $a_k \leftarrow \pi_{\boldsymbol{\theta}_k}(s_k, \boldsymbol{w}_k) + \epsilon_k$

  Observe $s'_k \sim \mathcal{P}(\cdot \mid s_k, a_k)$ and reward r_k

  Store $(s_k, a_k, r_k, s'_k, \boldsymbol{w}_k)$ in $\mathcal{D}_{\text{old}}$ and $\mathcal{D}_{\text{fresh}}$

  Sample $B_k = \{(s_{k_j^{\text{old}}}, a_{k_j^{\text{old}}}, r_{k_j^{\text{old}}}, s'_{k_j^{\text{old}}}, \boldsymbol{w}_{k_j^{\text{old}}})\}_{j=1}^{N}$ from $\mathcal{D}_{\text{old}}$

  Compute the ψ network loss using stored task vectors and $a'_{k_j^{\text{old}}} \leftarrow \pi_{\boldsymbol{\theta}_k}(s'_{k_j^{\text{old}}}, \bar{\boldsymbol{w}}_k)$:

$$L_{\psi} = \frac{1}{2} \sum_{j=1}^{N} \left(\left(r_{k_j^{\text{old}}} + \gamma \text{SG}\big(\psi_{\boldsymbol{u}_k}(s'_{k_j^{\text{old}}}, a'_{k_j^{\text{old}}}, \boldsymbol{w}_{k_j^{\text{old}}})^{\top} \boldsymbol{w}_{k_j^{\text{old}}}\big) - \psi_{\boldsymbol{u}_k}(s_{k_j^{\text{old}}}, a_{k_j^{\text{old}}}, \boldsymbol{w}_{k_j^{\text{old}}})^{\top} \boldsymbol{w}_{k_j^{\text{old}}} \right) \right)^2.$$

  Update the successor-feature parameters: $\boldsymbol{v}_{k+1} \leftarrow \boldsymbol{v}_k - \beta_k^{\psi} \nabla_{\boldsymbol{v}} \mathcal{L}_{\psi}(\boldsymbol{v}_k)$

  Update the task vector $\boldsymbol{w}_k$ as in Algorithm 3

  Compute the actor loss:

$$\mathcal{L}_{\pi}(\boldsymbol{\theta}_k) \leftarrow -\frac{1}{N} \sum_{j=1}^{N} \psi_{\boldsymbol{v}_k}\Big(s_{k_j^{\text{old}}}, \pi_{\boldsymbol{\theta}_k}(s_{k_j^{\text{old}}}, \boldsymbol{w}_{k_j^{\text{old}}}), \boldsymbol{w}_{k_j^{\text{old}}}\Big)^{\top} \boldsymbol{w}_j.$$

  Every every U steps, update $\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k - \beta_k^{\pi} \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\pi}(\boldsymbol{\theta}_k)$, treating $(\boldsymbol{v}_{k+1}, \boldsymbol{w}_{k+1})$ as fixed.

  if $(k+1) \bmod C = 0$ **then**

    Update target parameters:

$$(\bar{\boldsymbol{u}}_{k+1}, \bar{\boldsymbol{v}}_{k+1}, \bar{\boldsymbol{w}}_{k+1}, \bar{\boldsymbol{\theta}}_{k+1}) \leftarrow (\boldsymbol{u}_{k+1}, \boldsymbol{v}_{k+1}, \boldsymbol{w}_{k+1}, \boldsymbol{\theta}_{k+1}).$$

  else

    Keep $(\bar{\boldsymbol{u}}_{k+1}, \bar{\boldsymbol{v}}_{k+1}, \bar{\boldsymbol{w}}_{k+1}, \bar{\boldsymbol{\theta}}_{k+1}) \leftarrow (\bar{\boldsymbol{u}}_k, \bar{\boldsymbol{v}}_k, \bar{\boldsymbol{w}}_k, \bar{\boldsymbol{\theta}}_k)$

  end if

end for

Algorithm 5 SF-Bellman (Goal-Conditioned)

1: $\eta \in [0,1]$, $\gamma \in (0,1)$, minibatch size N, target-update period C, step sizes $\{\alpha_k, \beta_k^{\phi}, \beta_k^{\psi}\}_{k\geq 0}$, replay buffer $\mathcal{D}$, networks $\phi_{\boldsymbol{u}_0} : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}^d$ and $\psi_{\boldsymbol{v}_0} : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}^d$, and task vector network $\boldsymbol{w}_0 : \mathcal{G} \to \mathbb{R}^d$, task vector update period U
2: **for** each iteration k **do**
3: Sample g_k from $\mathcal{G}$
4: Select a_k using ϵ-greedy policy on $\boldsymbol{\psi}_{\boldsymbol{u}_k}(s_k, \cdot, g_k)^\top \boldsymbol{w}_k(g_k)$
5: Observe $s'_k \sim \mathcal{P}(\cdot \mid s_k, a_k)$ and reward $r_k := (s_k, a_k, s'_k, g_k)$
6: Store $(s_k, a_k, r_k, s'_k, g_k)$ in $\mathcal{D}$
7: Sample $B_k = \{(s_j, a_j, r_j, s'_j, g_j)\}_{j=1}^N$ from $\mathcal{D}$
8: Compute the network loss using stored task vectors:

$$L_\psi = \frac{1}{2}\sum_{j=1}^{N}\left(\left(r_j + \gamma \mathrm{SG}\big(\max_{a\in\mathcal{A}} \psi_{\boldsymbol{u}_k}(s'_j, a, g_j)^\top \boldsymbol{w}_j(g_j)\big) - \psi_{\boldsymbol{u}_k}(s_j, a_j, g_j)^\top \boldsymbol{w}_j(g_j)\right)\right)^2.$$

9: Compute the task vector loss:

$$L_{\mathrm{PRE}} = \sum_{l=1}^{N}\left(r_l - \phi_{\boldsymbol{u}_k}(s_l, a_l, g_l)^\top \boldsymbol{w}_k(g_l)\right)^2$$

$$L_{\mathrm{PBE}} = \sum_{l=1}^{N}\left(r_l + \gamma \mathrm{SG}\big(\max_{a\in\mathcal{A}} \psi_{\bar{\boldsymbol{u}}_k}(s'_l, a, g_l)^\top \bar{\boldsymbol{w}}_k(g_l)\big) - \psi_{\boldsymbol{u}_k}(s_l, a_l, g_l)^\top \boldsymbol{w}_k(g_l)\right)^2$$

10: Update parameters every U steps:

$$\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k + \alpha \nabla_{\boldsymbol{w}} \left(\frac{1-\eta}{2} L_{\mathrm{PRE}} + \frac{\eta}{2} L_{\mathrm{PBE}}\right)$$

11: Periodically copy $\boldsymbol{w} \to \bar{\boldsymbol{w}}$ (frequency C)
12: **end for**

SF-full refers to replacing the loss L_ψ with

$$L_\psi = \frac{1}{2}\sum_{j=1}^{N}\left\|\phi_j + \gamma \mathrm{SG}\left(\psi_{\boldsymbol{u}_k}(s'_j, a'_j)\right) - \psi_{\boldsymbol{u}_k}(s_j, a_j)\right\|^2.$$

where $a'_j = \arg\max_{a\in\mathcal{A}} \psi_{\boldsymbol{u}_k}^\top(s'_j, a)\boldsymbol{w}_k$.

E Experiments : Policy Evaluation

E.1 Environments

For policy evaluation, the action space is a singleton ($|\mathcal{A}| = 1$), so the expected reward vector $\boldsymbol{R} \in \mathbb{R}^{|\mathcal{S}|}$ is indexed directly by state $s \in \mathcal{S}$, denoted $[\boldsymbol{R}]_s = \mathbb{E}[r(s, a, s') \mid s]$.

n-Cycle. States $0, \ldots, n-1$ are arranged on a ring; each state transitions to either neighbour with probability $\frac{1}{2}$. The stationary distribution is uniform distribution. The reward is $[\boldsymbol{R}]_0 = [\boldsymbol{R}]_2 = 1$ and zero elsewhere.

Star graph. State 0 is the hub; states $1, \ldots, n-1$ are leaves. The transition is defined with probability $\varepsilon = 0.1$:

- Hub: stays at state 0 with probability $1 - \varepsilon$; moves to each leaf with probability $\varepsilon/(n-1)$.
- Leaf i: stays at state i with probability $1 - \varepsilon$; returns to hub with probability ε.

The stationary distribution satisfies $\mu(0) = \frac{1}{2}$ and $\mu(i) = \frac{1}{2(n-1)}$ for $i \geq 1$. The reward is $[\boldsymbol{R}]_0 = 1.1$ at the hub and $[\boldsymbol{R}]_i = 1$ at every leaf.

Oscillatory n-Cycle. Same transition dynamics as the n-Cycle (uniform random walk, $\mu = \mathbf{1}/n$), but with an alternating reward $r(s, a, s') = (-1)^s$.

E.2 Feature classes considered in policy evaluation

All experiments use $d = 3$. The following describes how each feature maps are constructed:

Mixed eigenbasis Let $\boldsymbol{U} \in \mathbb{R}^{n\times n}$ be the orthonormal eigenbasis of $\boldsymbol{P}^\pi$ (computed via symmetric eigen-decomposition). Draw $\boldsymbol{C} \in \mathbb{R}^{n\times d}$ with i.i.d. standard normal entries, form $\boldsymbol{UC}$, and set $\boldsymbol{\Phi}$ to be the orthonormal columns obtained from a QR decomposition.

Polynomial. States $s = 0, \ldots, n-1$ are rescaled to $x_s = (s+1)/(n+1)$. The raw features are $[\boldsymbol{\Phi}]_{s,j} = x_s^j, \qquad j = 1, \ldots, k.$

Gaussian radial basis function. Place d centers equally around the cycle of n states at $c_j = (j-1)n/d$, for $j = 1, \ldots, d$. For each state $s \in \{0, \ldots, n-1\}$, define the circular distance

$$\delta(s, c_j) = \min(|s - c_j|,\, n - |s - c_j|)\,.$$

Set the bandwidth to $\sigma = n/d$ and define the feature matrix $\boldsymbol{\Phi} \in \mathbb{R}^{n\times d}$ by

$$[\boldsymbol{\Phi}]_{s,j} = \exp\left(-\frac{\delta(s, c_j)^2}{2\sigma^2}\right), \qquad j = 1, \ldots, d.$$

Random. Each entry $[\boldsymbol{\Phi}]_{s,j}$ is drawn i.i.d. from $\mathcal{N}(0, 1)$.

Fourier. States $s = 0, \ldots, n-1$ are rescaled to $x_s = s/n$. The features are defined as follows:

$$[\boldsymbol{\Phi}]_{s,j} = \cos(\pi j\, x_s), \qquad j = 0, \ldots, d-1,$$

ordered from smoothest ($j = 0$, the constant feature) to most oscillatory.

F Experiment Details

F.1 Implementation Details

Networks and Target Parameters. *SF-Bellman* uses online parameters $(\boldsymbol{u}_k, \boldsymbol{v}_k, \boldsymbol{w}_k)$ for the feature network $\phi_{\boldsymbol{u}}(\cdot, \cdot)$, the successor-feature network $\psi_{\boldsymbol{v}}(\cdot, \cdot)$, and the task vector $\boldsymbol{w}_k$, together with target network parameters $(\bar{\boldsymbol{u}}_k, \bar{\boldsymbol{v}}_k, \bar{\boldsymbol{w}}_k)$, initialized as $(\bar{\boldsymbol{u}}_0, \bar{\boldsymbol{v}}_0, \bar{\boldsymbol{w}}_0) = (\boldsymbol{u}_0, \boldsymbol{v}_0, \boldsymbol{w}_0)$. The target parameters are copied from the online parameters every C iterations or updated with Polyak averaging rule (Lillicrap et al., 2015). For actor-critic implementation, we keep an additional actor network parameter $\boldsymbol{\theta}_k$ and its target parameter $\bar{\boldsymbol{\theta}}_k$.

Shared encoder architecture. The feature map $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ and $\psi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ share the encoder structure where the encoder is a network $h : \mathcal{S} \to \mathbb{R}^d$. This design follows from Chua et al. (2024).

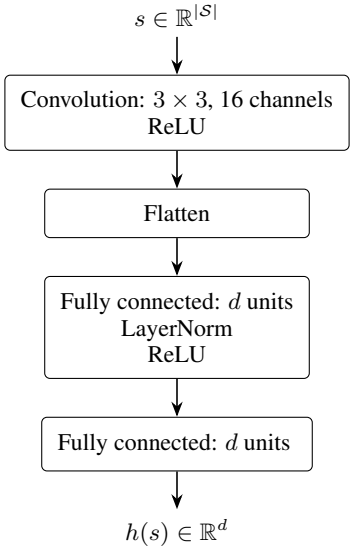


Figure 4: Encoder architecture used by the feature maps ϕ and ψ.

SF-Bellman and DQN architecture DQN shares the same encoder structure in Figure 4.

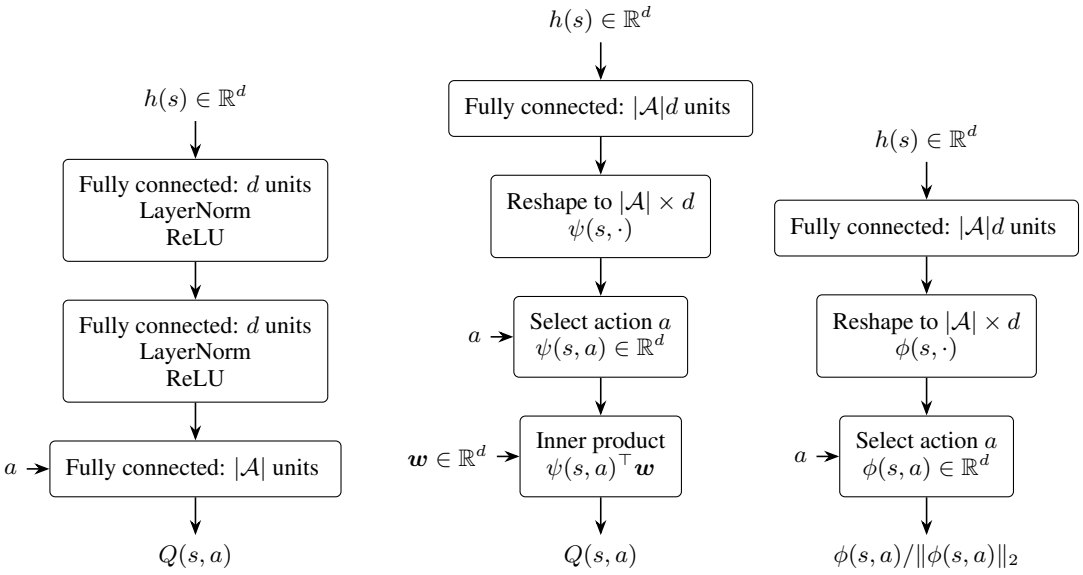
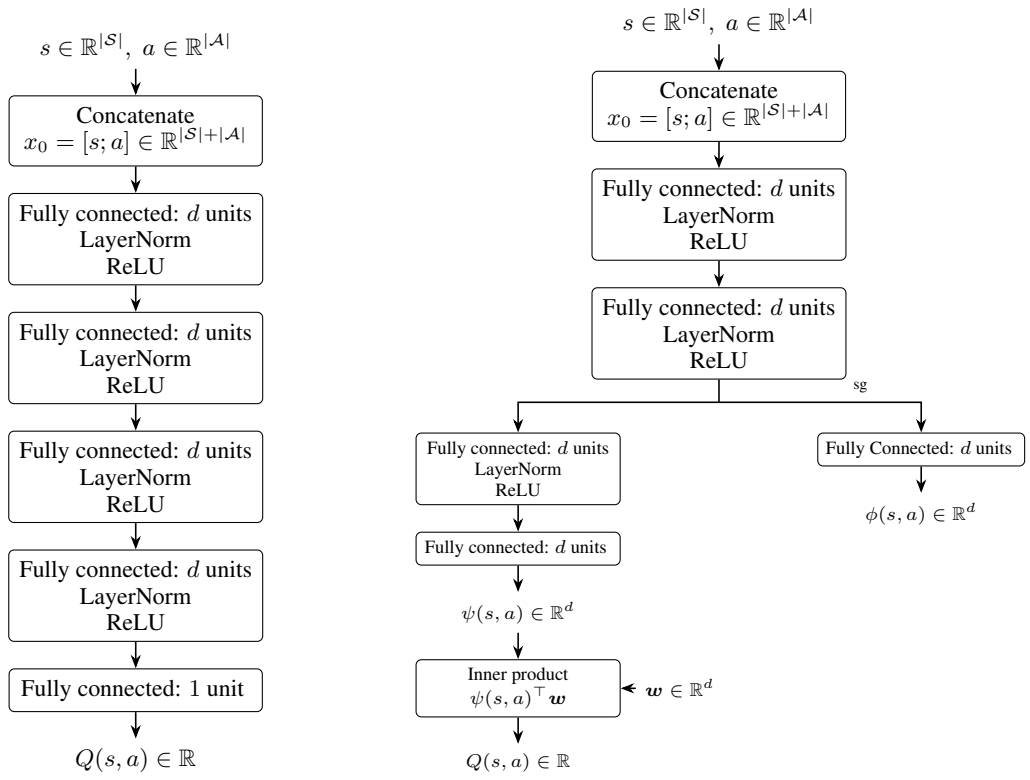


Figure 5: Architectures for (a) the DQN head, (b) SF Q-function computation, and (c) SF feature computation. The DQN head uses two hidden layers of width d to obtain a trainable parameter count comparable to that of the SF-Bellman Q-function head.

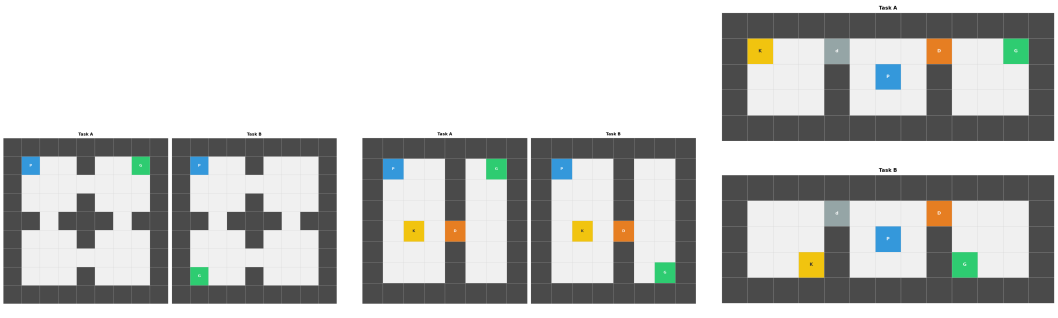
Actor–critic implementation. We parameterize the deterministic actor by $\pi_{\boldsymbol{\theta}_k}$ and maintain a target actor $\pi_{\bar{\boldsymbol{\theta}}_k}$. The critic is represented as $\psi_{\boldsymbol{v}_k}(s,a)^\top \boldsymbol{w}_k$. The target parameters are held fixed when constructing bootstrapped targets and are copied from the online parameters every C iterations.



(a) DDPG critic (b) SF critic

Figure 6: The DDPG critic processes the concatenated observation and action through four nonlinear hidden blocks before the scalar action value. The SF critic uses a two-block shared encoder followed by successor-feature and immediate-feature heads. The task vector is not provided as an input to either head. The depth of DDPG has chosen to closely to match the number of parameters of that of SF-critic while keeping the same width.

G CONTINUAL-LEARNING EXPERIMENTS



(a) FourRoomS9 (b) DoorKey8x8 (c) KeyCorridorS3R1

Figure 7: Continual-learning environments. The blue block is the location of the agent, yellow denotes the key, orange denotes the locked door, grey denotes the unlocked door, green denotes the goal location.

G.1 Environment details

For discrete control environments, we use three MiniGrid tasks implemented in NAVIX (Pignatelli et al., 2026). In both, the agent receives a 7×7 egocentric RGB view, plus one channel indicating whether it carries a key. The action space has five discrete actions: *turn left*, *turn right*, *move forward*, *pick up*, and *toggle* (open or unlock a door). The reward is 1 for reaching the goal and 0 otherwise, and an episode ends at the goal or after 800 steps. Each environment has two tasks, A and B, that share the same layout and differ only in object positions. The tasks alternate every 5×10^4 steps over two cycles ($A \to B \to A \to B$).

FourRoom-S9 There are four rooms and the agent needs to reaach the goal.

DoorKey-8x8. A wall splits the grid into two rooms that are connected by a locked door. The agent must pick up the key, unlock the door, and reach the goal in the second room. The key and door are fixed, and only the goal moves: it is in the top-right corner in task A and in the bottom-right corner in task B.

KeyCorridor-S3R1. The grid consists of three rooms in a row: a key room, a corridor, and a goal room. The agent starts in the corridor. It must open the door to the key room, pick up the key, unlock the door to the goal room, and reach the goal. The door positions are fixed. The key and goal are both in the top corners in task A and both in the bottom corners in task B.

For continuous-control environments, We evaluate the agents on three different levels of dynamical complexity: PointMaze, Reacher, and PandaReach. In every environment, the transition dynamics remain unchanged across tasks, while the reward function changes through a switch in the goal position. The goal is omitted from the observation, so the active task must be inferred from observed rewards. For task $i \in \{1, 2\}$ the reward function is defined as

$$r_k = \mathbf{1}\{\|\boldsymbol{x}_{k+1} - \boldsymbol{g}_i\|_2 < \epsilon\}, \tag{22}$$

where $\mathbf{1}\{\cdot\}$ is the indicator function, $\boldsymbol{x}_{k+1}$ denotes the the position of controlled object after an action, $\boldsymbol{g}_i$ is the goal location for task i, and ϵ is the success threshold.

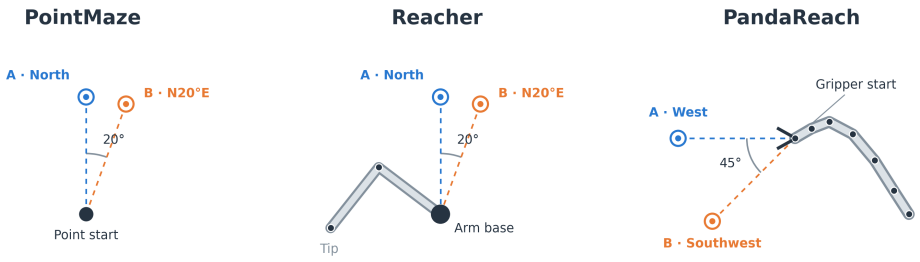


Figure 8: Continual-learning environments on continuous control tasks

PointMaze. PointMaze consists of a two-dimensional point mass moving in an open, wall-free arena. The agent directly controls forces along the horizontal and vertical axes. Its observation is

$$\boldsymbol{s}_k = [p_x \quad p_y \quad \dot{p}_x \quad \dot{p}_y]^\top \in \mathbb{R}^4, \tag{23}$$

and the action lies in $\mathbb{R}^2$. The target switches between North and a target 20° east of North, both at radius 0.30. The success threshold is 0.05, and each episode lasts 300 environment steps. This environment provides the simplest setting, with direct Cartesian control and relatively simple translational dynamics.

Reacher. Reacher is a planar two-link robotic arm controlled through two joint actuators. The agent must account for the nonlinear mapping from joint-space actions to the Cartesian position of the fingertip. With the goal hidden, the observation is

$$\boldsymbol{s}_t = \left[\cos(q_1) \quad \cos(q_2) \quad \sin(q_1) \quad \sin(q_2) \quad \tfrac{d}{dt}q_1 \quad \tfrac{d}{dt}q_2\right]^\top \in \mathbb{R}^6, \tag{24}$$

Table 4: Properties of the goal-hidden switching environments.

Environment	Observation	Action	Goal radius	Horizon
PointMaze	4	2	0.30	300
Reacher	6	2	0.15	50
PandaReach	10	4	0.075	50

and the action lies in $\mathbb{R}^2$. The target switches from North to 20° east of North at radius 0.15. The success threshold is 0.04, and the episode horizon is 50 steps. Reacher tests whether an algorithm remains effective under nonlinear dynamics.

PandaReach. PandaReach uses the seven-joint Franka Emika Panda arm. The policy produces a four-dimensional Cartesian control command. The first three components specify an end-effector displacement, while the fourth component corresponds to the fixed gripper and is ignored. A damped least-squares inverse-kinematics controller (Wampler, 1986) maps the Cartesian command to the Panda's joint actuators. We use the MuJoCo Menagerie Franka Emika Panda model executed with the MJX backend (Zakka et al., 2025). The simulation uses a 5 ms physics timestep.

With the goal hidden, the observation is

$$\boldsymbol{s}_t = \begin{bmatrix} \boldsymbol{p}_{\mathrm{ee}}^\top & \boldsymbol{q}_{\mathrm{finger}}^\top & \dot{\boldsymbol{p}}_{\mathrm{ee}}^\top & \dot{\boldsymbol{q}}_{\mathrm{finger}}^\top \end{bmatrix}^\top \in \mathbb{R}^{10}, \tag{25}$$

where $\boldsymbol{p}_{\mathrm{ee}} \in \mathbb{R}^3$ is the end-effector position, $\boldsymbol{q}_{\mathrm{finger}} \in \mathbb{R}^2$ contains the finger positions, $\dot{\boldsymbol{p}}_{\mathrm{ee}} \in \mathbb{R}^3$ is the end-effector velocity, and $\dot{\boldsymbol{q}}_{\mathrm{finger}} \in \mathbb{R}^2$ contains the finger velocities. The target switches from West to Southwest at radius 0.075. The success threshold is 0.05, and the episode horizon is 50 steps. PandaReach tests the method under more complex robot physics, kinematic constraints, and controller-induced nonlinearities.

Together, these environments test task adaptation across direct Cartesian motion, nonlinear planar articulation, and higher-complexity robotic control. Using multiple environment families helps distinguish a general switching mechanism from behavior that is specific to one set of dynamics.

G.2 Hyperparameter Choices

The hyperparameter choice has been done by grid search. After running three seeds for each parameter configuration, the best parameter has chosen by the one maximizing the training average and then has been extend to 15 seeds.

Table 5: DQN and DDPG hyperparameters. Bold entries indicate the configuration selected separately for each algorithm by maximizing mean performance across environments. For DQN, performance is averaged over evaluation checkpoints in the final 10% of each task phase in the second cycle, then over selection seeds, tasks, and environments with equal weight. For DDPG, training-goal evaluation scores are averaged over the final 10% of training and over selection seeds, then min–max normalized across candidate configurations within each environment and averaged equally across environments. The selected configuration is shared across the corresponding environments.

(a) DQN

Parameter	Search grid	Value
Tuned parameters		
Feature dimension	$\{32, 48\}$	$\mathbf{48}$
Learning rate	$\{10^{-4}, 10^{-3}\}$	$\mathbf{10^{-4}}$
Target update frequency	$\{1, 10, 100\}$	$\mathbf{10}$
Fixed parameters		
Replay buffer size	—	10^4
Parallel environments	—	64
Update-to-data ratio	—	0.5
Batch size	—	64
Optimizer	—	Adam
Adam ϵ	—	1.5×10^{-4}

(b) DDPG

Parameter	Search grid	Value
Tuned parameters		
Feature dimension	$\{48, 128\}$	$\mathbf{128}$
Critic learning rate	$\{10^{-4}, 10^{-3}\}$	$\mathbf{10^{-3}}$
Critic target soft-update coefficient	$\{10^{-2}, 10^{-1}\}$	$\mathbf{10^{-1}}$
Fixed parameters		
Actor learning rate	—	5×10^{-4}
Actor target soft-update coefficient	—	10^{-1}
Replay buffer size	—	10^5
Parallel environments	—	64
Update-to-data ratio	—	0.5
Batch size	—	64
Optimizer	—	Adam
Adam ϵ	—	10^{-8}
Exploration noise	—	0.1

Table 6: SF-simple hyperparameters. Bold entries indicate jointly selected values from the search grid for each η across FourRoomS9, DoorKey8×8, and KeyCorridorS3R1.

Parameter	Search grid	Value		
		$\eta = 0.0$	$\eta = 0.99$	$\eta = 1.0$
Tuned parameters				
Feature dimension	$\{32, 48\}$	$\mathbf{32}$	$\mathbf{32}$	$\mathbf{32}$
ψ target update frequency	$\{1, 10, 100\}$	$\mathbf{10}$	$\mathbf{10}$	$\mathbf{10}$
$\boldsymbol{w}$ target update frequency	$\{1, 10, 100, 1000\}$	$\mathbf{100}$	$\mathbf{100}$	$\mathbf{100}$
$\boldsymbol{w}$ update interval	$\{1, 10\}$	$\mathbf{10}$	$\mathbf{10}$	$\mathbf{10}$
ψ learning rate	$\{10^{-4}, 10^{-3}\}$	$\mathbf{10^{-3}}$	$\mathbf{10^{-3}}$	$\mathbf{10^{-3}}$
$\boldsymbol{w}$ learning rate	$\{10^{-4}, 10^{-3}\}$	$\mathbf{10^{-4}}$	$\mathbf{10^{-4}}$	$\mathbf{10^{-4}}$
Fixed parameters				
Replay buffer size	—	10^4		
Parallel environments	—	64		
Update-to-data ratio	—	0.5		
Batch size	—	64		
Optimizer	—	Adam		
Adam ϵ	—	1.5×10^{-4}		

Table 7: SF-simple hyperparameters. Bold entries indicate jointly selected values from the search grid for each η across PointMaze, Reacher, and PandaReach in the non-visible switching setting.

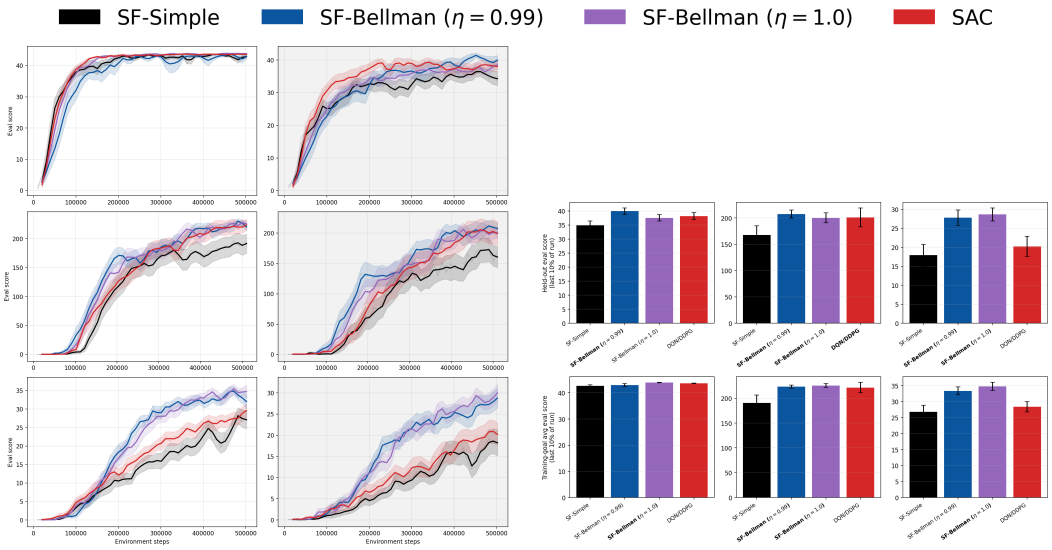
Parameter	Search grid	Value		
		$\eta = 0.0$	$\eta = 0.99$	$\eta = 1.0$
Tuned parameters				
Feature dimension	$\{48, 128\}$	$\mathbf{48}$	$\mathbf{48}$	$\mathbf{128}$
ψ learning rate	$\{10^{-4}, 10^{-3}\}$	$\mathbf{10^{-3}}$	$\mathbf{10^{-3}}$	$\mathbf{10^{-3}}$
$\boldsymbol{w}$ learning rate	$\{10^{-4}, 10^{-3}\}$	$\mathbf{10^{-4}}$	$\mathbf{10^{-4}}$	$\mathbf{10^{-4}}$
ψ target soft-update coefficient	$\{10^{-2}, 10^{-1}\}$	$\mathbf{10^{-1}}$	$\mathbf{10^{-2}}$	$\mathbf{10^{-1}}$
$\boldsymbol{w}$ target soft-update coefficient	$\{10^{-1}, 1\}$	$\mathbf{10^{-1}}$	$\mathbf{10^{-1}}$	$\mathbf{10^{-1}}$
$\boldsymbol{w}$ update interval	$\{1, 10\}$	$\mathbf{10}$	$\mathbf{10}$	$\mathbf{10}$
Fixed parameters				
Actor learning rate	—	5×10^{-4}		
Actor target soft-update coefficient	—	10^{-1}		
Replay buffer size	—	10^5		
Parallel environments	—	64		
Update-to-data ratio	—	0.5		
Batch size	—	64		
Optimizer	—	Adam		
Adam ϵ	—	10^{-8}		
Exploration noise	—	0.1		

Additional Implementation Details. In switching environments, SF-Simple and SF-Bellman use task vector conditioned value functions based on universal value function approximation (Schaul et al., 2015), as described in Algorithm 3. We store the task vector with each transition, allowing replay data to remain associated with their task vectors, and keep the replay buffers across task switches. In contrast, DQN and DDPG do not receive task vectors, and their observations do not identify the current task: goals are unobserved in continuous-control tasks and only partially observable in MiniGrid. We therefore reset their replay buffers at each switch to avoid potentially conflicting learning targets from different tasks.

For discrete tasks, we use an ϵ-greedy exploration schedule, decaying ϵ from 1.0 during the first task and from 0.3 after each subsequent task switch. This lower initial exploration rate allows the agent to reuse learned behavior while continuing to explore.

H GOAL CONDITIONED EXPERIMENTS

H.1 LEARNING CURVES AND METRICS



(a) Learning curve

(b) Held-out score and train score

Figure 9: Pick-and-Place results.

H.2 ENVIRONMENT DETAILS

Goal sampling. The goal space is divided into eight equal angular wedges of width $45°$. Let $\mathcal{A}_{\text{train}}$ denote the set of angles that the goals can be sampled during training:

$$\mathcal{A}_{\text{train}} = [-22.5°, 22.5°] \cup [67.5°, 112.5°] \cup [157.5°, 202.5°] \cup [247.5°, 292.5°].$$

The held-out evaluation goals are located in the diagonal angular regions (Northeast, Northwest, Southwest, and Southeast) which are excluded from training.

The wedges in $\mathcal{A}_{\text{trian}}$ are centered on East, North, West, and South, respectively. Because they have equal angular width, each wedge was selected with probability $1/4$, followed by uniform sampling within that wedge. A new goal was sampled independently at every episode reset for each parallel environment. The goal position was

$$g(\alpha) = c + r \begin{bmatrix} \cos\alpha \\ \sin\alpha \end{bmatrix},$$

where c is the environment-specific reference position and r is a fixed radius. For Reacher, c was the arm origin and $r = 0.15$; for PointMaze, c was the origin and $r = 0.60$; and for PandaReach, c was the initial gripper position and $r = 0.15$. The evaluation score was measured as follows:

$$\texttt{eval_test_avg} = \frac{1}{4}\left(G_{\text{NE}} + G_{\text{NW}} + G_{\text{SW}} + G_{\text{SE}}\right)$$

where G_i denotes the return given goal i. Each fixed goal was evaluated over 10 episodes.

H.3 HYPERPARAMETER CHOICES

Table 8: DDPG hyperparameters. Bold entries indicate jointly selected values from the search grid.

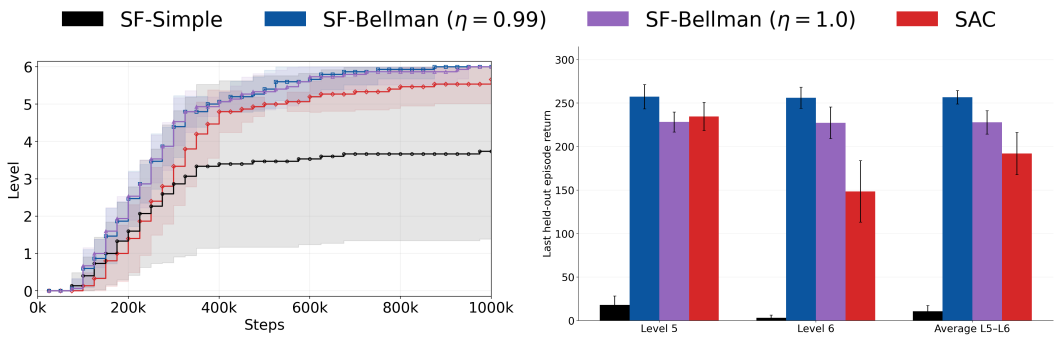
Parameter	Search grid	Value
Tuned parameters		
Critic network width	$\{48, 128\}$	$\mathbf{128}$
Critic learning rate	$\{10^{-4}, 10^{-3}\}$	$\mathbf{10^{-3}}$
Critic target coefficient	$\{10^{-2}, 10^{-1}\}$	$\mathbf{10^{-2}}$
Network architecture		
Actor hidden layers	—	2
Actor hidden blocks	—	Linear–ReLU
Actor output	—	Linear–Tanh, rescaled to action bounds
Policy	—	Deterministic
Fixed parameters		
Replay buffer size	—	10^5
Parallel environments	—	64
Update-to-data ratio	—	0.5
Batch size	—	64

Table 9: Hyperparameters for SF-Bellman with a goal-conditioned $\boldsymbol{w}$ network. Bold entries indicate jointly selected values from the search grid for each η.

Parameter	Search grid	Value		
		$\eta = 0.0$	$\eta = 0.99$	$\eta = 1.0$
Tuned parameters				
Feature dimension	$\{48, 128\}$	$\mathbf{128}$	$\mathbf{128}$	$\mathbf{128}$
ψ learning rate	$\{10^{-4}, 10^{-3}\}$	$\mathbf{10^{-3}}$	$\mathbf{10^{-3}}$	$\mathbf{10^{-3}}$
$\boldsymbol{w}$ learning rate	$\{10^{-4}, 10^{-3}\}$	$\mathbf{10^{-3}}$	$\mathbf{10^{-3}}$	$\mathbf{10^{-4}}$
ψ target soft-update coefficient	$\{10^{-2}, 10^{-1}\}$	$\mathbf{10^{-2}}$	$\mathbf{10^{-1}}$	$\mathbf{10^{-2}}$
$\boldsymbol{w}$ target soft-update coefficient	$\{10^{-1}, 1\}$	$\mathbf{1}$	$\mathbf{10^{-1}}$	$\mathbf{10^{-1}}$
$\boldsymbol{w}$ update interval	$\{1, 10\}$	$\mathbf{10}$	$\mathbf{1}$	$\mathbf{1}$
Network architecture				
Goal-conditioned $\boldsymbol{w}$ network	—	Linear–LayerNorm–ReLU–Linear		
Actor hidden layers	—	2		
Actor hidden blocks	—	Linear–ReLU		
Actor output	—	Linear–Tanh, rescaled to action bounds		
Policy	—	Deterministic		
Fixed parameters				
Actor learning rate	—	5×10^{-4}		
Actor target soft-update coefficient	—	10^{-1}		
Replay buffer size	—	10^5		
Parallel environments	—	64		
Update-to-data ratio	—	0.5		
Batch size	—	64		
Optimizer	—	Adam		
Adam ϵ	—	10^{-8}		
Exploration noise	—	0.1		

I CURRICULUM LEARNING

I.1 LEARNING CURVES



(a) Curriculum level

(b) Success rate

Figure 10: Pick-and-Place results.

I.2 ENVIRONMENT DETAILS

Pick and Place. We used the Meta-World MT1 `pick-place-v3` environment (Yu et al., 2020). Each task was specified by a cube $\boldsymbol{o} \in \mathbb{R}^3$ and a goal position $\boldsymbol{g} \in \mathbb{R}^3$. Candidate positions were sampled uniformly from

$$\begin{aligned} \mathcal{O} &= [-0.1, 0.1] \times [0.6, 0.7] \times \{0.02\}, \\ \mathcal{G} &= [-0.1, 0.1] \times [0.8, 0.9] \times [0.05, 0.30], \end{aligned} \tag{26}$$

where the coordinate units are in meters. For each seed, we generated a fixed set of 50 training goals. At every episode reset, a task was selected uniformly with replacement from this 50 training goals. The selected task fixed the cube and goal position for that episode. The goal coordinates were included in the agent's observation. The promotion to next level occurs when the agent scores more than 80% on its current level training evaluation.

Reset curriculum. We used a curriculum that progressively expand the distribution of starting states away from a goal as the agent improves (Florensa et al., 2017). The levels considered here were $\ell \in \{0, \dots, 6\}$.

At levels 0–3, the object was initially grasped and lifted. Let $\boldsymbol{o}$ denote the original cube position and $\boldsymbol{g}$ the goal. We defined the lifted reference position as follows:

$$\tilde{\boldsymbol{o}} = [[\boldsymbol{o}]_0 \quad [\boldsymbol{o}]_1 \quad \max([\boldsymbol{g}]_3, 0.10)]^\top, \qquad D = \|\tilde{\boldsymbol{o}} - \boldsymbol{g}\|_2.$$

For each reset, we sampled $\delta \sim \mathcal{U}(0.08, 0.10)$ where $\mathcal{U}$ is the uniform distribution on the corresponding interval, and set the desired cube–goal distance to

$$d_\ell = \left(1 - \frac{\ell}{4}\right) \delta + \frac{\ell}{4} D. \tag{27}$$

The initialized position of the cube is then defined as

$$\boldsymbol{p}_\ell = \boldsymbol{g} + \frac{\tilde{\boldsymbol{o}} - \boldsymbol{g}}{D}\, d_\ell.$$

The individual levels were:

- **Level 0:** The object started grasped and lifted near the goal, positioned 8–10 cm from the goal.
- **Level 1:** Separation by (27).

- **Level 2:** Separation by (27).
- **Level 3:** Separation by (27).
- **Level 4:** The gripper was closed around the cube at $\boldsymbol{o}$.
- **Level 5:** The gripper started open around the cube. The policy therefore had to establish the grasp before lifting and transporting the cube.
- **Level 6:** The gripper started open, with its initial position moved 25% from the grasp position toward the original hand position. The policy had to approach, grasp, lift, and transport the cube.

Specifically, the grasp reference was $\boldsymbol{h}_{\mathrm{grasp}} = \boldsymbol{o} + [-0.005, 0, 0.03]^\top$ and the original hand position was $\boldsymbol{h}_{\mathrm{orig}} = [0, 0.6, 0.2]^\top$, in meters. Level 5 used $\boldsymbol{h}_{\mathrm{grasp}}$, whereas level 6 used

$$\boldsymbol{h}_6 = 0.75\boldsymbol{h}_{\mathrm{grasp}} + 0.25\boldsymbol{h}_{\mathrm{orig}}.$$

Reward and success criterion. We replaced the dense reward in Meta-World with a sparse binary reward:

$$r_k = \mathbf{1}\left[\left\|\boldsymbol{p}_{k+1}^{\mathrm{cube}} - \boldsymbol{g}\right\|_2 \le 0.07\right],$$

where $\boldsymbol{p}_k^{\mathrm{cube}}$ is the position of the cube. Episodes retained the original 500-step horizon. An episode was counted as successful if the success condition was satisfied at least once:

$$S = \mathbf{1}\left[\exists\, k \le 500 : \left\|\boldsymbol{p}_k^{\mathrm{obj}} - \boldsymbol{g}\right\|_2 \le 0.07\right].$$

Consequently, episode success rate and cumulative return are different metrics.

Held-out evaluation. For each training seed n, we generated a separate, fixed set of 50 evaluation goals using seed $n + 10{,}000$, which we call the held-out goals. These goals were sampled from the same cube–goal distribution in (26). Held-out performance therefore measures generalization to unseen goals.

Evaluation used one episode for each of the 50 held-out goals:

$$\texttt{eval_test_avg} = \frac{1}{50}\sum_{i=1}^{50} G_i$$

where G_i denotes the return with goal i for $i \in [50]$.

I.3 HYPERPARAMETER CHOICES

Table 11: SAC hyperparameters on pick and place. The bold entry indicates the selected value based on minimum mean steps to first reach level 5. The critic depth was chosen to approximately match the trainable parameter count of the SF-based method.

Parameter	Search grid	Value
Tuned parameters		
Critic learning rate	$\{3 \times 10^{-4}, 10^{-3}\}$	$\mathbf{3 \times 10^{-4}}$
Network architecture		
Network width	—	256
Critic hidden layers	—	6
Critic hidden blocks	—	Linear–LayerNorm–ReLU
Critic output	—	Linear to scalar Q
Actor hidden layers	—	2
Actor hidden blocks	—	Linear–ReLU
Actor output heads	—	Mean and log standard deviation
Policy	—	Tanh-squashed Gaussian
Fixed parameters		
Actor learning rate	—	3×10^{-4}
Temperature learning rate	—	3×10^{-4}
Initial temperature	—	1
Critic target coefficient	—	0.005
Replay buffer size	—	10^6
Update-to-data ratio	—	1
Batch size	—	256
Initial random steps	—	10^4

Table 12: Hyperparameters for *SF-Bellman*. Bold entries indicate the selected values for each η, based on minimum mean steps to first reach level 5, with failures assigned 10^6 steps.

Parameter	Search grid	Value		
		$\eta = 0.0$	$\eta = 0.99$	$\eta = 1.0$
Tuned parameters				
ψ learning rate	$\{3 \times 10^{-4}, 10^{-3}\}$	$\mathbf{3 \times 10^{-4}}$	$\mathbf{3 \times 10^{-4}}$	$\mathbf{3 \times 10^{-4}}$
$\boldsymbol{w}$ learning rate	$\{3 \times 10^{-4}, 10^{-3}\}$	$\mathbf{3 \times 10^{-4}}$	$\mathbf{3 \times 10^{-4}}$	$\mathbf{3 \times 10^{-4}}$
ψ target soft-update coefficient	$\{0.005, 0.02\}$	$\mathbf{0.005}$	$\mathbf{0.005}$	$\mathbf{0.005}$
$\boldsymbol{w}$ update interval	$\{1, 10\}$	$\mathbf{10}$	$\mathbf{1}$	$\mathbf{1}$
Network architecture				
Network width	—	256		
Feature dimension	—	256		
Shared encoder	—	2×(Linear–LayerNorm–ReLU)		
ψ head	—	Linear–LayerNorm–ReLU–Linear		
ϕ head	—	Linear, followed by ℓ_2 normalization		
Goal-conditioned $\boldsymbol{w}$ network	—	2×(Linear–LayerNorm–ReLU), then Linear		
Actor hidden layers	—	2		
Actor hidden blocks	—	Linear–ReLU		
Actor output heads	—	Mean and log standard deviation		
Policy	—	Tanh-squashed Gaussian		
Fixed parameters				
$\boldsymbol{w}$ target soft-update coefficient	—	0.005		
Actor learning rate	—	3×10^{-4}		
Temperature learning rate	—	3×10^{-4}		
Initial temperature	—	1		
Replay buffer size	—	10^6		
Update-to-data ratio	—	1		
Batch size	—	256		
Initial random steps	—	10^4		