Full-Gradient Successor Feature Representations

Ritish Shrirao
IIIT Bangalore

Aditya Priyadarshi
IIIT Bangalore

I. Introduction

A. Successor Feature Representations (SFR)

While standard Successor Features assume a linear relationship between features and rewards, Successor Feature Representations (SFR) [3] generalize this to handle arbitrary, potentially non-linear reward functions. SFR decouples the environment dynamics from the rewards by learning the cumulative discounted probability of observing specific features. This is encapsulated in the ξ-function.

For a policy π, the ξ-function represents the future cumulative discounted probability of successor features $\phi \in \Phi$:

$$\xi^{\pi}(s,a,\phi) = \sum_{k=0}^{\infty} \gamma^k P(\phi_{t+k} = \phi \mid s_t = s, a_t = a, \pi) \quad (1)$$

where $\gamma \in [0,1)$ is the discount factor. Crucially, if the reward function $R(\phi)$ depends only on the features, the action-value function $Q^{\pi}(s,a)$ can be exactly reconstructed by aggregating the ξ-values:

$$Q^{\pi}(s,a) = \sum_{\phi \in \Phi} \xi^{\pi}(s,a,\phi) R(\phi) \quad (2)$$

This decomposition allows an agent to compute the value of an existing policy π under a *new* reward function R' simply by replacing $R(\phi)$ in the equation above, without requiring new environmental interactions.

B. Generalized Policy Improvement (GPI)

The primary mechanism for transfer in this framework is Generalized Policy Improvement (GPI). Suppose an agent has learned a library of policies $\Pi = \{\pi_1, \ldots, \pi_n\}$ for various previous tasks. When facing a new task with reward R_{new}, the agent can estimate the value of *all* stored policies using their respective learned ξ-functions:

$$Q^{\pi_i}_{new}(s,a) = \sum_{\phi \in \Phi} \xi^{\pi_i}(s,a,\phi) R_{new}(\phi) \quad (3)$$

GPI defines a new policy π_{GPI} that selects the action maximizing the value across all known policies:

$$\pi_{GPI}(s) \in \arg\max_a \max_i Q^{\pi_i}_{new}(s,a) \quad (4)$$

This transfer mechanism allows the agent to instantly perform at the level of the best-suited previous policy and provides a strong initialization for further learning.

C. The Semi-Gradient Problem

Despite their empirical success, SFR and standard SF methods rely on Temporal Difference (TD) learning, specifically Q-learning variants to train the ξ-functions. From an optimization perspective, these are *semi-gradient* methods. In the standard Bellman update, the target value (which includes the parameters of the neural network) is treated as a fixed constant during the gradient computation.

Semi-gradient methods lack robust convergence guarantees when combined with non-linear function approximation. The learning process effectively becomes a moving target problem. This instability is particularly detrimental in the SF framework, where GPI relies on the accurate estimation of ξ-values. If the ξ-functions diverge or oscillate due to semi-gradient instability, the GPI upper-bound guarantee loosens, leading to poor transfer performance.

D. Proposed Solution: FG-SFRQL

To address these theoretical deficiencies, we propose **Full-Gradient Successor Feature Representations Q-Learning (FG-SFRQL)**. Drawing on the Full Gradient DQN (FG-DQN) framework [4], we formulate the learning of ξ-functions as the direct minimization of the Mean Squared Bellman Error (MSBE). Unlike standard methods, we compute the gradient with respect to the parameters in both the online estimate and the target value.

By applying full-gradient updates, we transform the training dynamics from a fixed-point iteration into a true stochastic gradient descent procedure. In this work, we provide a theoretical convergence proof for FG-SFRQL in the multi-task setting and demonstrate empirically that minimizing the full residual leads to faster adaptation and higher cumulative rewards compared to standard semi-gradient baselines.

Our code is available on github.

II. Algorithmic setup and notation

We assume a finite action set $\mathcal{A}$ and a finite feature set Φ. There are m tasks indexed by $j \in \{1, \ldots, m\}$. For each task j the agent maintains a parametric successor-feature approximator $\xi_j : \mathcal{S} \times \mathcal{A} \times \Phi \times \mathbb{R}^d \to \mathbb{R}$, given by $(s,a,\phi,\theta^{(j)}) \mapsto \xi_j(s,a,\phi;\theta^{(j)})$. We collect all task-parameters into the joint vector $\Theta = (\theta^{(1)}, \ldots, \theta^{(m)}) \in \mathbb{R}^{md}$.

At time k the environment (under the current task i_k) produces a transition (s_k, a_k, s'_k). The behaviour may depend on past iterates through Generalized Policy Improvement (GPI).

The algorithm selects a prior policy index c_k by GPI, choosing from the set

$$S(\Theta, s'_k, i_k) = \arg \max_{(k',a') \in \{1,\ldots,m\} \times \mathcal{A}} Q_{k'}(s'_k, a'; \theta^{(k')}), \quad (5)$$

where the action-value function $Q_{k'}$ is defined as

$$Q_{k'}(s', a'; \theta^{(k')}) = \sum_{\phi \in \Phi} \xi_{k'}(s', a', \phi; \theta^{(k')}) \, \mathcal{R}_{i_k}(\phi), \quad (6)$$

and $\mathcal{R}_{i_k}$ is the current task's reward expressed on features. If the maximizer is not unique the set $S(\Theta, s', i_k)$ contains multiple pairs. This is handled by considering the convex hull of the gradients corresponding to all maximizing pairs.

The algorithm updates the parameter blocks for the current task i_k and for the chosen prior c_k (if $c_k \neq i_k$) by taking full gradients of the squared Bellman residual evaluated at the single sampled transition. All other blocks are left unchanged. Appendix A contains the pseudocode of our proposed algorithm.

III. Averaged Loss and Full Gradient

To address the bias inherent in minimizing the Bellman error with stochastic transitions, we employ a modified experience replay strategy (similar to [4]). For a visited state-action pair (s, a) and current task i, let $\mathcal{K} = \{(r_p, s'_p, \phi_p)\}_{p=1}^{N}$ be a set of N transitions sampled from the replay buffer $\mathcal{D}$ that start with (s, a).

Let (k', a') be the specific action-value maximizing pair chosen via GPI, i.e., $(k', a') \in S(\Theta, \mathbb{E}[s'|s, a], i)$. We define the *averaged* squared Bellman residual for task j as:

$$\ell_j\big(\Theta, (s,a), \mathcal{K}, (k', a')\big) := \left(\frac{1}{N} \sum_{p=1}^{N} \Big[\mathbb{I}(\phi = \phi_p) + \gamma \xi_j(s'_p, a', \phi; \theta^{(j)}) \Big] - \xi_j(s, a, \phi; \theta^{(j)}) \right)^2 \quad (7)$$

This loss minimizes the distance between the current prediction and the *empirical mean* of the target, which approximates the true expectation $\mathbb{E}[\cdot|s, a]$ as $N \to \infty$.

The per-block full gradient for task j is computed by taking the gradient of (7) with respect to $\theta^{(j)}$. Crucially, following the Full-Gradient DQN scheme, we treat occurrences of θ in both the target and the prediction as variables to be differentiated:

$$\nabla_{\theta^{(j)}} \ell_j = \sum_{\phi \in \Phi} 2 \cdot \underbrace{\left[\left(\frac{1}{N} \sum_{p=1}^{N} (\mathbb{I}(\phi = \phi_p) + \gamma \xi_j(s'_p, a', \phi; \theta^{(j)})) \right) - \xi_j(s, a, \phi; \theta^{(j)}) \right]}_{\text{Averaged Bellman Error Component}} \cdot \underbrace{\left(\left(\frac{1}{N} \sum_{p=1}^{N} \gamma \nabla_{\theta^{(j)}} \xi_j(s'_p, a', \phi; \theta^{(j)}) \right) - \nabla_{\theta^{(j)}} \xi_j(s, a, \phi; \theta^{(j)}) \right)}_{\text{Averaged Feature Gradient Component}} . \quad (8)$$

The full joint update vector $G(\Theta)$ uses this gradient in the appropriate blocks. If the current task is i and the chosen prior is c, the j-th block of the stochastic update vector G is:

$$G_j(\Theta, s, a, s', i, c) = \begin{cases} \nabla_{\theta^{(i)}} \ell_i & \text{if } j = i \\ \nabla_{\theta^{(c)}} \ell_c & \text{if } j = c \text{ and } c \neq i \\ \mathbf{0} & \text{otherwise} \end{cases} \quad (9)$$

This formulation removes the stochastic bias associated with the single-sample squared error, ensuring the update direction aligns with the gradient of the true Mean Squared Bellman Error.

IV. Assumptions

Assumption 1 (Properties of the System)**.**

1) ***(Regularity and Boundedness)*** *For every policy $j \in \{1, \ldots, m\}$, the map $(s, a, \phi, \theta^{(j)}) \mapsto \xi_j(s, a, \phi; \theta^{(j)})$ is bounded and twice continuously differentiable in its parameter vector $\theta^{(j)}$, with bounded first and second derivatives.*
2) ***(Almost-Everywhere Uniqueness of GPI Maximizer)*** *For any given state s' and current task index i, let the value of taking action a' according to policy k be $Q_k(s', a'; \theta^{(k)}) = \sum_{\phi'} \xi_k(s', a', \phi'; \theta^{(k)}) \mathcal{R}_i(\phi')$. Define the set of optimizers for the GPI step as:*

$$S(\Theta, s', i) = \operatorname*{argmax}_{(k,a) \in \{1,\ldots,i\} \times \mathcal{A}} Q_k(s', a; \theta^{(k)}) \quad (10)$$

The set of joint parameters Θ for which this optimizer is not unique is assumed to have Lebesgue measure zero. That is, the set

$$\{\Theta \in \mathbb{R}^{md} \ : \ |S(\Theta, s', i)| > 1\} \quad (11)$$

has measure zero for any fixed (s', i).
3) ***(Finiteness of Critical Points)*** *Let the joint Bellman error be defined as the sum of individual policy errors: $E(\Theta) = \sum_{j=1}^{m} E_j(\theta^{(j)})$. The set of critical points of $E(\Theta)$, where the generalized gradient $\partial E(\Theta)$ contains the zero vector, is assumed to be finite.*

Assumption 2 (Stability of Iterates)**.** *The sequence of joint parameter iterates $\{\Theta_k\}_{k \geq 0}$ generated by the algorithm remains almost surely bounded. That is, $\sup_k \|\Theta_k\| < \infty$ a.s.*

Assumption 3 (Decomposition of Stochastic Error)**.** *Let $\tilde{G}_k = G(s_k, a_k, \mathcal{K}_k, \Theta_k)$ denote the stochastic update vector computed at step k using a batch of N_k transitions $\mathcal{K}_k$. Let $\bar{G}_k(\Theta_k) = -\nabla E(\Theta_k)$ be the true mean-field update, corresponding to the exact gradient of the aggregate Bellman error $E(\Theta)$ weighted by the stationary distribution.*

The update vector can be decomposed into the true mean field, a martingale difference noise component, and a residual bias component:

$$\tilde{G}_k = \bar{G}_k(\Theta_k) + M_{k+1} + \varepsilon_k$$

where the terms satisfy the following conditions:

1) ***(Martingale Difference Noise)*** *The term $M_{k+1} = \tilde{G}_k - \mathbb{E}[\tilde{G}_k \mid \mathcal{F}_k]$ is a martingale difference sequence with*

respect to the filtration $\mathcal{F}_k$, satisfying $\mathbb{E}[M_{k+1} \mid \mathcal{F}_k] = \mathbf{0}$ and having bounded conditional second moments: $\mathbb{E}[\|M_{k+1}\|^2 \mid \mathcal{F}_k] \leq C(1 + \|\Theta_k\|^2)$ for some constant $C > 0$.

2) ***(Residual Approximation Bias)** The term $\varepsilon_k = \mathbb{E}[\tilde{G}_k \mid \mathcal{F}_k] - \bar{G}_k(\Theta_k)$ represents the bias arising from using a finite batch size N_k to approximate the conditional expectations in the full gradient. This bias is assumed to diminish asymptotically as the batch size increases and be summable in a weighted sense:*
   - $\|\varepsilon_k\| \to 0$ *almost surely (as $N_k \to \infty$).*
   - $\sum_{k=0}^{\infty} \alpha_k \mathbb{E}[\|\varepsilon_k\|] < \infty$.

Assumption 4 (Step-sizes and visitation)**.** *The scalar step-sizes $\{\alpha_k\}$ satisfy $\alpha_k > 0$, $\sum_k \alpha_k = \infty$, and $\sum_k \alpha_k^2 < \infty$. Every coordinate (block) j is updated infinitely often almost surely.*

Assumption 5 (Finite actions and finite tie-set)**.** *The action set $\mathcal{A}$ is finite and m (number of tasks) is finite, so for each (Θ, s', i) the maximizer set $S(\Theta, s', i)$ is a finite subset of $\{1, \ldots, m\} \times \mathcal{A}$.*

V. Set-valued mean-field map

For a fixed transition sample (s, a, s') and task index i, we use the set of GPI maximizers $S(\Theta, s', i)$ defined in Assumption 1.2. The single-step, conditional mean-field update from Assumption 3 is a selection from the following set-valued map:

$$H(\Theta, (s, a, i)) \;:=\; -\mathbb{E}_{s' \sim p(\cdot|s,a)} \Big[\mathrm{co}\Big\{\nabla_\Theta \ell\big(\Theta, (s, a, s', i), (k', a')\big) \;:\; (k', a') \in S(\Theta, s', i)\Big\}\Big],$$

where $\nabla_\Theta \ell$ is the joint gradient from (**??**). By Assumption 1.1 (Regularity) and Assumption 5 (Finiteness), each gradient term is a continuous function of Θ and the set of maximizers is finite. Therefore, the set-valued map H is nonempty, compact, convex-valued, and upper semi-continuous in Θ.

The mean-field dynamics are governed by the averaged map

$$\overline{H}(\Theta) \;:=\; \mathbb{E}_{(s,a,i) \sim \mu}\big[H(\Theta, (s, a, i))\big],$$

where μ is the stationary sampling distribution over state-action-task triplets. This map inherits the properties of H.

VI. Main theorem

Theorem 1 (Joint-iterate convergence)**.** *Under Assumptions 1–5, the joint iterate sequence $\{\Theta_k\}$ produced by the FG-SFRQL algorithm (Algorithm 1) converges almost surely to a stationary point Θ^* of the aggregate Bellman error $E(\Theta) = \sum_{j=1}^{m} E_j(\theta^{(j)})$, i.e., a point satisfying $\nabla E(\Theta^*) = 0$.*

Proof. The proof follows the o.d.e. approach for stochastic approximation, adapted for set-valued maps and differential inclusions [1], [2].

a) Update rule: The FG-SFRQL update is given by $\Theta_{k+1} = \Theta_k - \alpha_k \tilde{G}_k$, where $\tilde{G}_k$ is the stochastic update vector computed at step k. This vector has at most two non-zero blocks, corresponding to the current task i_k and the GPI-selected prior policy c_k. We analyze the update by decomposing it into its conditional mean and a noise term. Conditioned on the history $\mathcal{F}_k = \sigma(\Theta_m, s_m, a_m, s'_m$ for $m < k; \Theta_k, s_k, a_k)$, let $\bar{G}_k(\Theta_k) = \mathbb{E}[\tilde{G}_k \mid \mathcal{F}_k]$. We define the martingale difference sequence $M_{k+1} := \tilde{G}_k - \bar{G}_k(\Theta_k)$. Using the decomposition from Assumption 3, the update can be written as:

$$\Theta_{k+1} = \Theta_k - \alpha_k \left(\bar{G}_k(\Theta_k) + M_{k+1} + \varepsilon_k\right). \tag{12}$$

b) The set-valued map H: The term $\bar{G}_k(\Theta_k)$ is the conditional mean of the stochastic update vector. The update for block i_k is the negative gradient of the Bellman error $E_{i_k}(\theta^{(i_k)})$, while the update for block c_k is the negative gradient of the Bellman error $E_{c_k}(\theta^{(c_k)})$. When the GPI maximizer for the next action is not unique, the update is a selection from a set. We define the set-valued map H for a fixed state-action-task triplet (s, a, i) as:

$$H(\Theta, (s, a, i)) \;:=\; \mathbb{E}_{s' \sim p(\cdot|s,a)} \left[\mathrm{co}\left\{G(\Theta, (s, a, s', i, c)) \;:\; c \in \underset{k'}{\mathrm{argmax}} Q_{k'}(s', a')\right\}\right], \tag{13}$$

where $\mathrm{co}\{\cdot\}$ denotes the convex hull and G is the joint update vector whose j-th block is $-\nabla E_j(\theta^{(j)})$ if $j \in \{i, c\}$ and zero otherwise. The term $\bar{G}_k(\Theta_k)$ is thus a selection from $H(\Theta_k, (s_k, a_k, i_k))$.

c) Stochastic recursive inclusion and verification of conditions: Using the map H, the update rule (12) is a stochastic recursive inclusion:

$$\Theta_{k+1} \in \Theta_k - \alpha_k \left(H(\Theta_k, (s_k, a_k, i_k)) + M_{k+1} + \varepsilon_k\right). \tag{14}$$

We verify the conditions for the convergence theorem of such inclusions (Theorem 7.1 of [2]). Assumptions 1-5 and 4 are constructed to meet these conditions: (A1) H is non-empty, compact, convex-valued, and upper semi-continuous; (A2) The driving process is Markov; (A3) Step-sizes are Robbins-Monro; (A4) Noise terms are summable; and (A5) Iterates are stable. With these conditions satisfied, the iterates $\{\Theta_k\}$ will asymptotically track the solutions of a limiting differential inclusion.

d) Asymptotic behavior and limiting differential inclusion: In a sequential task setting, the system is non-stationary. However, if we consider a system where tasks are sampled from a stationary distribution $\pi(i)$, the limiting behavior of the iterates is described by the differential inclusion:

$$\begin{aligned} &\dot{\Theta}(t) \in -\overline{H}(\Theta(t)), \\ \text{where} \quad &\overline{H}(\Theta) = \sum_i \pi(i) \mathbb{E}_{(s,a) \sim \mu_i}[H(\Theta, (s, a, i))]. \end{aligned} \tag{15}$$

By Assumption 1.2, the GPI maximizer is unique for almost every Θ. This reduces the differential inclusion to an ODE

$\dot{\Theta}(t) = V(\Theta(t))$, where $V(\Theta)$ is the mean-field update vector field. For a given active task i and GPI-selected policy c, the block components of this vector field are $V_j(\Theta) = -\nabla E_j(\theta^{(j)})$ for $j \in \{i, c\}$ and $V_j(\Theta) = 0$ otherwise.

e) Convergence Analysis: We analyze the aggregate Bellman error $E(\Theta) = \sum_{j=1}^{m} E_j(\theta^{(j)})$ function. Its time derivative along the trajectories of the limiting ODE is given by the inner product $\frac{d}{dt}E(\Theta) = \langle \nabla E(\Theta), \dot{\Theta} \rangle$. Substituting $\dot{\Theta} = V(\Theta)$:

$$\begin{aligned}
\frac{d}{dt}E(\Theta(t)) &= \langle \nabla E(\Theta), V(\Theta) \rangle \\
&= \sum_{j=1}^{m} \langle \nabla E_j(\theta^{(j)}), V_j(\Theta) \rangle \\
&= \langle \nabla E_i(\theta^{(i)}), -\nabla E_i(\theta^{(i)}) \rangle + \langle \nabla E_c(\theta^{(c)}), -\nabla E_c(\theta^{(c)}) \rangle \\
&= -\|\nabla E_i(\theta^{(i)})\|^2 - \|\nabla E_c(\theta^{(c)})\|^2. \qquad (16)
\end{aligned}$$

Since the squared norms are non-negative, we have $\frac{d}{dt}E(\Theta(t)) \leq 0$. This establishes that even though the update direction $V(\Theta)$ is not the gradient of $E(\Theta)$, it is a descent direction for it. The total error is guaranteed to be non-increasing along the system's trajectories.

The discrete-time counterpart follows from a Taylor expansion of $E(\Theta)$. The change in energy after one step is analyzed by starting with the first-order expansion of the function around the current iterate Θ_k:

$$E(\Theta_{k+1}) = E(\Theta_k) + \langle \nabla E(\Theta_k), \Theta_{k+1} - \Theta_k \rangle + O(\|\Theta_{k+1} - \Theta_k\|^2).$$

We substitute the update rule $\Theta_{k+1} - \Theta_k = -\alpha_k(\bar{G}_k(\Theta_k) + M_{k+1} + \varepsilon_k)$. Since the update step is proportional to α_k, the remainder term is $O(\alpha_k^2)$. Taking the conditional expectation with respect to the history $\mathcal{F}_k$ and noting that $\mathbb{E}[M_{k+1} \mid \mathcal{F}_k] = 0$, we get:

$$\mathbb{E}[E(\Theta_{k+1}) \mid \mathcal{F}_k] = E(\Theta_k) - \alpha_k \langle \nabla E(\Theta_k), \bar{G}_k(\Theta_k) + \varepsilon_k \rangle + O(\alpha_k^2).$$

The key term is the inner product $\langle \nabla E(\Theta_k), \bar{G}_k(\Theta_k) \rangle$. The vector $\nabla E(\Theta_k)$ has the gradient of each policy's Bellman error in its corresponding block. The mean-field update vector $\bar{G}_k(\Theta_k)$ has non-zero components only in the blocks for the current task i_k and the GPI-selected policy c_k. The inner product thus becomes:

$$\begin{aligned}
\langle \nabla E(\Theta_k), \bar{G}_k(\Theta_k) \rangle &= \sum_{j=1}^{m} \langle \nabla E_j(\theta_k^{(j)}), [\bar{G}_k(\Theta_k)]_j \rangle \\
&= \langle \nabla E_{i_k}(\theta_k^{(i_k)}), \nabla E_{i_k}(\theta_k^{(i_k)}) \rangle \\
&\quad + \langle \nabla E_{c_k}(\theta_k^{(c_k)}), \nabla E_{c_k}(\theta_k^{(c_k)}) \rangle \\
&= \|\nabla E_{i_k}(\theta_k^{(i_k)})\|^2 + \|\nabla E_{c_k}(\theta_k^{(c_k)})\|^2.
\end{aligned}$$

Substituting this back into the expansion yields the evolution of the expected energy:

$$\begin{aligned}
\mathbb{E}[E(\Theta_{k+1}) \mid \mathcal{F}_k] = E(\Theta_k) - \alpha_k \Big(&\|\nabla E_{i_k}(\theta_k^{(i_k)})\|^2 + \|\nabla E_{c_k}(\theta_k^{(c_k)})\|^2 \\
&+ \langle \nabla E(\Theta_k), \varepsilon_k \rangle \Big) + O(\alpha_k^2). \qquad (17)
\end{aligned}$$

This again has the form required for the almost supermartingale convergence theorem. Given the assumptions, it follows that $E(\Theta_k)$ converges a.s., and $\sum_k \alpha_k(\|\nabla E_{i_k}\|^2 + \|\nabla E_{c_k}\|^2)$ must be finite. As tasks are visited infinitely often, this implies that $\|\nabla E_j(\theta^{(j)})\| \to 0$ for all j. Combined with Assumption 1.3 (finiteness of critical points), the iterates $\{\Theta_k\}$ must converge almost surely to a single stationary point Θ^* where $\nabla E(\Theta^*) = 0$. □

VII. Experiments

A. Experimental Setup

We evaluate the algorithms on two distinct domains: a discrete grid-world navigation task (four rooms) and a continuous control robotic task (reacher). In both settings, the agents must solve a sequence of tasks where the dynamics remain fixed, but the reward functions (goals) change over time. Our implementation is adapted from this github repository.

1) Discrete Features - Object Collection Environment: **Environment:** The domain consists of a grid-world divided into four interconnected rooms. The agent must navigate from a starting position to a goal location while collecting objects. Objects possess two attributes: color (orange or blue) and shape (box or triangle). The state space uses Gaussian radial basis functions (RBFs) to encode the agent's position and memory of collected items. The feature vector $\phi \in \{0,1\}^5$ is binary, encoding whether a specific type of object has been collected or if the goal has been reached.

Tasks: Agents are trained on a sequence of tasks that differ in their reward schemes. For linear tasks, rewards are defined as $r = \phi^\top \mathbf{w}_i$. The weights corresponding to object properties are sampled uniformly from $\mathcal{U}(-1,1)$, while reaching the goal yields a fixed positive reward. This setup tests the agent's ability to adapt to changing preferences for object types while maintaining navigation capabilities.

2) Continuous Features - Reacher Environment: **Environment:** We utilize a Reacher environment involving a two-joint robotic arm aiming to reach a target coordinates. The state space is continuous, consisting of joint angles, angular velocities, and the vector from the end-effector to the target. The action space is discretized to control joint torques.

Tasks: The agent solves a sequence of tasks where the target location is randomized. Features ϕ concatenate the state vector and a proximity metric, allowing the reward function to be approximated linearly with respect to the distance to the goal.

B. Baselines and Hyperparameters

We compare the following algorithms:

1) **DQN:** The standard Deep Q-Network with experience replay and target networks. It learns a single value function from scratch or fine-tunes for each new task, lacking a specific transfer mechanism.
2) **SFDQN:** The standard Successor Feature approach. It decouples the value function into successor features (ψ) and reward weights (w). It uses Generalized Policy Improvement (GPI) to transfer knowledge but relies on

the standard semi-gradient update (treating the target network parameters as fixed constants).

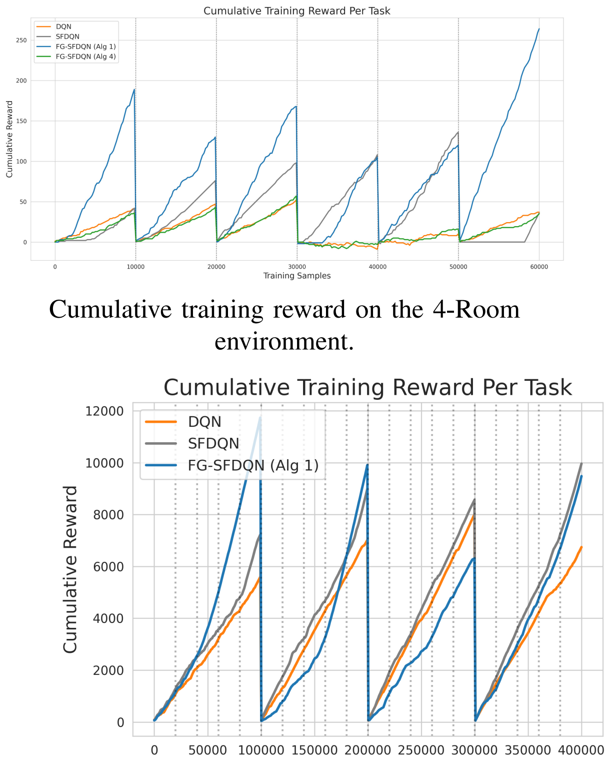
3) **FG-SFRQL (Ours):** The proposed algorithm which also uses SF and GPI but updates the parameters by minimizing the full Bellman residual, differentiating with respect to both the online and target parameters.

All agents utilize Multi-Layer Perceptrons (MLP). Table I details the specific hyperparameters used during training.

TABLE I
EXPERIMENTAL HYPERPARAMETERS

Parameter	**Value**
Training Steps (per task)	$10,000$
Number of Tasks	3
Batch Size	64
Replay Buffer Size	$200,000$
Discount Factor (γ)	0.95
Exploration Rate (ϵ)	0.60
Max Episode Horizon (T)	200
SFQL Learning Rate (α)	0.001
Reward Weight Learning Rate (α_w)	0.5

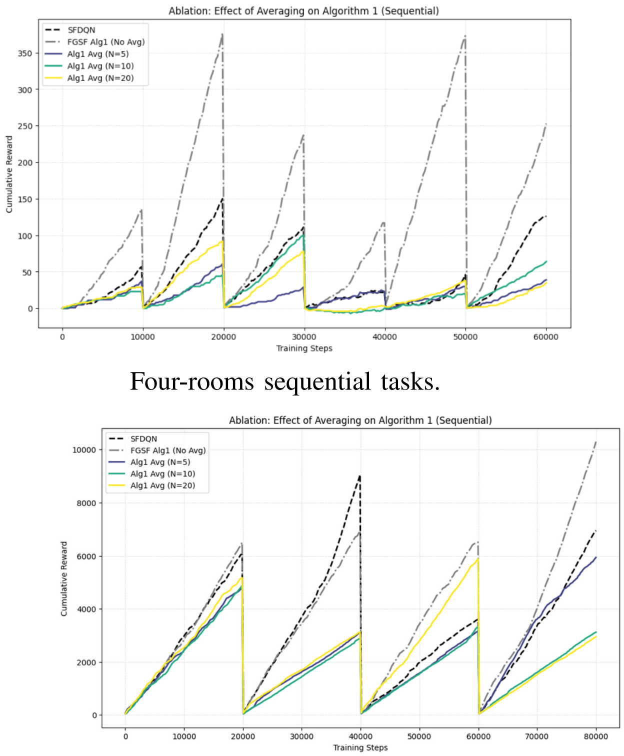
C. Results and Analysis



Cumulative training reward on the 4-Room environment.

Cumulative training reward on the Reacher environment.

Fig. 1. Empirical comparison between DQN, SFDQN and FG-SFDQN (Algo-1 1) on the two benchmark domains described in (VII). Each vertical band corresponds to training on a single task. Dotted vertical lines indicate task switches. All curves plot cumulative reward collected by the agent during training on the active task (higher = better).



Four-rooms sequential tasks.

Reacher sequential tasks.

Fig. 2. Ablation of averaging for FG-SFDQN (Alg. 1). Curves plot cumulative reward collected on the active task over training steps. Dashed black: SFDQN (semi-gradient SF baseline). Grey dash-dot: FG-SFDQN without averaging ("No Avg"). Blue/green/yellow: FG-SFDQN with averaging using $N = 5, 10, 20$ next-state encodings respectively. FG-SFDQN without averaging outperforms SFDQN achieving nearly *twice* the cumulative reward.

1) ***Comparison: DQN vs. SFDQN vs. FG-SFDQN:*** Figures VII-C1 and VII-C1 present the cumulative training reward across tasks for the Four-Room environment and the Reacher domain, respectively. In both settings, FG-SFDQN substantially outperforms all baselines, achieving markedly higher cumulative reward throughout training.

FG-SFDQN exhibits significantly faster reward accumulation, indicating more efficient credit assignment and improved utilization of experience. Across task switches, FG-SFDQN rapidly adapts and maintains a steeper growth in cumulative reward compared to DQN and semi-gradient SFDQN. This consistent advantage suggests that full-gradient optimization leads to more effective updates of the successor feature representation, enabling improved generalization and transfer across tasks.

2) ***Effect of Averaging on Algorithm 1:*** Figures VII-C2 and VII-C2 report the ablation study on the effect of averaging. In this experiment, the averaging variant implements the theoretically motivated update rule, where we sample a batch of N transitions conditional on a pivot (s, a), then compute the empirical mean of the *successor feature vectors* output by the network, $\bar{\xi}(s', a') = \frac{1}{N} \sum \xi(s'_i, a'; \theta)$, and use this averaged vector to select the GPI prior and compute the Bellman target.

Despite the theoretical stability provided by approximating the expected feature vector $\mathbb{E}[\xi(s', a')]$, the results indicate that averaging is detrimental to performance in this setting. The FG-SFDQN without averaging (grey dash-dot) learns rapidly, achieving nearly twice the cumulative reward of the standard SFDQN baseline in the four-room environment. Conversely, introducing averaging ($N = 5, 10, 20$) significantly suppresses this early learning phase. All averaged variants produce substantially lower cumulative reward than the single-sample FG-SFDQN. While larger N reduces variance, it does not translate to faster convergence or higher returns and struggles to match the aggressive learning curve of the stochastic, unaveraged update.

We hypothesize the following reasons for this counter-intuitive result.

- **Beneficial stochasticity.** The randomness in single-sample gradient updates can be helpful for learning. This noise allows the parameters to explore different directions in optimization space and can help avoid getting stuck in suboptimal regions. Averaging approximates a mean-field update, reducing gradient stochasticity and leading to more conservative parameter updates and slower policy improvement.
- **Dampening of strong learning signals.** During learning, some transitions might produce large updates and change the behavior early in training. When multiple transitions are averaged together, these strong updates are mixed with many weaker ones, which may reduce their immediate impact. As a result, the network may adapt more slowly during the early stages of learning.

*3) **Final evaluation**:* To rigorously quantify the final performance, we conducted a held-out evaluation after the conclusion of the training phase. The evaluation protocol tested each agent on every task for $n_{\text{episodes}} = 10$ episodes, with each episode capped at 100 steps.

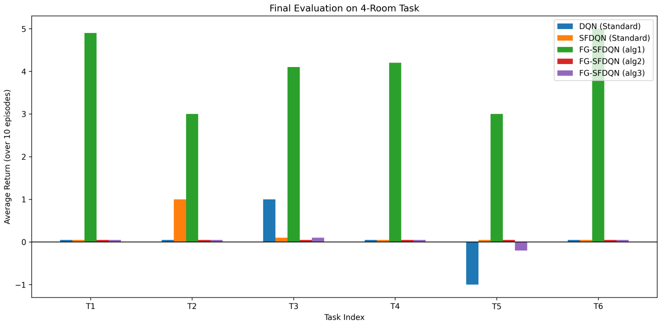


Fig. 3. Final evaluation on 4-room tasks (average return over 10 episodes).

FG-SFDQN (Alg. 1) achieves the strongest final performance across both evaluated domains. In the four-rooms environment, Alg. 1 attains mean returns in the range of approximately 3.0–5.0, while all baselines obtain near-zero returns, with DQN exhibiting negative performance on some tasks.

In the continuous Reacher suite with four target locations, Alg. 1 similarly achieves the highest average returns across the majority of targets. In contrast, the alternative full-gradient variants (Alg. 2 and Alg. 3) display mixed behavior: while they achieve moderate performance in the Reacher domain, their performance degrades to near-zero in the four-rooms evaluation. This highlights the robustness of Alg. 1 across both discrete and continuous settings.

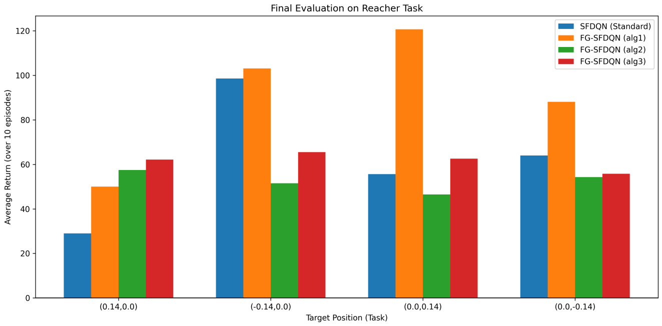


Fig. 4. Final evaluation on Reacher tasks (average return over 10 episodes).

TABLE II
FINAL EVALUATION: 4-ROOM TASKS (AVERAGE RETURN OVER 10 EPISODES).

Algorithm	T1	T2	T3	T4	T5	T6
DQN (Standard)	0.0	0.0	1.0	0.0	-1.0	0.0
SFDQN (Standard)	0.0	1.0	0.1	0.0	0.0	0.0
FG-SFDQN (alg1)	4.9	3.0	4.1	4.2	3.0	5.0
FG-SFDQN (alg2)	0.0	0.0	0.0	0.0	0.0	0.0
FG-SFDQN (alg3)	0.0	0.0	0.1	0.0	-0.2	0.0

TABLE III
FINAL EVALUATION: REACHER TASKS (AVERAGE RETURN OVER 10 EPISODES).

Algorithm	$(0.14, 0.0)$	$(-0.14, 0.0)$	$(0.0, 0.14)$	$(0.0, -0.14)$
SFDQN (Standard)	29.0	98.6	55.6	63.98
FG-SFDQN (alg1)	50.0	103.1	120.7	88.09
FG-SFDQN (alg2)	57.5	51.5	46.5	54.31
FG-SFDQN (alg3)	62.2	65.5	62.6	55.80

We have presented three algorithms are presented in the Appendix:

- **Algorithm 1** presents the sequential implementation used in the experiments, where tasks are learned one after another. This is a direct modification of the standard SFRQL algorithm that incorporates full gradient updates.
- **Algorithm 2** introduces randomized task sampling. By drawing tasks from a distribution $\pi(i)$, it ensures that state-action-task triplets are visited according to a stationary distribution, satisfying the stationarity requirement of the mean-field analysis.
- **Algorithm 3** extends Algorithm 2 by incorporating *averaging* over next-state transitions (using a batch size N).

REFERENCES

[1] Borkar, V. S.: Stochastic Approximation: A Dynamical Systems Viewpoint. Hindustan Publishing Agency, New Delhi, and Cambridge University Press, Cambridge, UK (2008)

[2] Yaji, V.G., Bhatnagar, S.: Stochastic recursive inclusions with non-additive iterate- dependent Markov noise. Stochastics, 90, 330-363 (2018)

[3] Reinke, C., Alameda-Pineda, Xavier: Successor Feature Representations. arXiv preprint arXiv:2110.15701 (2021)

[4] Avrachenkov, K.E., Borkar, V.S., Dolhare, H.P., Patil, K.: Full gradient DQN reinforcement learning: A provably convergent scheme. In: Modern Trends in Controlled Stochastic Processes: Theory and Applications, V. III, pp. 192–220. Springer (2021)

Appendix A: Algorithm

Algorithm 1 Joint Full Gradient Model-free SFRQL (FG-SFRQL)

Input: exploration rate ϵ; learning rate for ξ-functions α; learning rate for reward models $\mathcal{R}$: α_R
Input: features ϕ or Φ; optional: reward functions for tasks: $\{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_{\text{num\_tasks}}\}$

```
for $i = 1$ to num_tasks do
   if $\mathcal{R}_i$ not provided then
      initialize $\tilde{\mathcal{R}}_i$: $\theta_i^R \leftarrow$ small random initial values
   if $i = 1$ then
      initialize $\xi_1$: $\theta_1^\xi \leftarrow$ small random initial values
   else
      $\theta_i^\xi \leftarrow \theta_{i-1}^\xi$
   new_episode $\leftarrow$ true
   for $t = 1$ to num_steps do
      if new_episode then
         new_episode $\leftarrow$ false
         $s_t \leftarrow$ initial state
      $c \leftarrow \arg\max_{k \in \{1,\ldots,i\}} \max_a \sum_{\phi \in \Phi} \xi_k(s_t, a, \phi; \theta_k^\xi) \tilde{\mathcal{R}}_i(\phi)$ {GPI optimal policy}
      With probability $\epsilon$ select a random action $a_t$, otherwise $a_t \leftarrow \arg\max_a \sum_{\phi \in \Phi} \xi_c(s_t, a, \phi; \theta_c^\xi) \tilde{\mathcal{R}}_i(\phi)$
      Take action $a_t$, observe reward $r_t$ and next state $s_{t+1}$. Let $\phi_t = \phi(s_t, a_t, s_{t+1})$.
      if $\tilde{\mathcal{R}}_i$ not provided then
         Update $\theta_i^R$ using SGD($\alpha_R$) with $\mathcal{L}_R = (r_t - \tilde{\mathcal{R}}_i(\phi_t))^2$
      if $s_{t+1}$ is a terminal state then
         $\gamma_t \leftarrow 0$; new_episode $\leftarrow$ true
      else
         $\gamma_t \leftarrow \gamma$

      {GPI optimal next action for task $i$}
      $\hat{a}_{t+1} \leftarrow \arg\max_{a'} \arg\max_{k \in \{1,\ldots,i\}} \sum_{\phi' \in \Phi} \xi_k(s_{t+1}, a', \phi'; \theta_k^\xi) \tilde{\mathcal{R}}_i(\phi')$
      {Full Gradient update for policy $i$}
      $\nabla_{\theta_i^\xi} \mathcal{L}_t^{(i)} \leftarrow \sum_{\phi \in \Phi} \Big[ \Big( \mathbb{I}(\phi = \phi_t) + \gamma_t \xi_i(s_{t+1}, \hat{a}_{t+1}, \phi; \theta_i^\xi) \Big) - \xi_i(s_t, a_t, \phi; \theta_i^\xi) \Big] \times$
                  $\Big( \gamma_t \nabla_{\theta_i^\xi} \xi_i(s_{t+1}, \hat{a}_{t+1}, \phi; \theta_i^\xi) - \nabla_{\theta_i^\xi} \xi_i(s_t, a_t, \phi; \theta_i^\xi) \Big)$
      $\theta_i^\xi \leftarrow \theta_i^\xi - \alpha_t \nabla_{\theta_i^\xi} \mathcal{L}_t^{(i)}$
      if $c \neq i$ then
         {Optimal next action for executed policy $c$}
         $\hat{a}_{t+1} \leftarrow \arg\max_{a'} \sum_{\phi' \in \Phi} \xi_c(s_{t+1}, a', \phi'; \theta_c^\xi) \tilde{\mathcal{R}}_c(\phi')$
         {Full Gradient update for policy $c$}
         $\nabla_{\theta_c^\xi} \mathcal{L}_t^{(c)} \leftarrow \sum_{\phi \in \Phi} \big[ \big( \mathbb{I}(\phi = \phi_t) + \gamma_t \xi_c(s_{t+1}, \hat{a}_{t+1}, \phi; \theta_c^\xi) \big) - \xi_c(s_t, a_t, \phi; \theta_c^\xi) \big] \times$
                  $\Big( \gamma_t \nabla_{\theta_c^\xi} \xi_c(s_{t+1}, \hat{a}_{t+1}, \phi; \theta_c^\xi) - \nabla_{\theta_c^\xi} \xi_c(s_t, a_t, \phi; \theta_c^\xi) \Big)$
         $\theta_c^\xi \leftarrow \theta_c^\xi - \alpha_t \nabla_{\theta_c^\xi} \mathcal{L}_t^{(c)}$
      For all other $j \notin \{i, c\}$, $\theta_j^\xi \leftarrow \theta_j^\xi$.
      $s_t \leftarrow s_{t+1}$
=0
```

Algorithm 2 Full-Gradient SFRQL with Randomized Task Sampling

```
Input: exploration rate $\epsilon$; learning rates $\alpha_\xi$, $\alpha_R$
Input: features $\Phi$; set of tasks $\{1, \dots, m\}$; optional task rewards $\{\mathcal{R}_i\}$
Initialize $\{\theta_i^\xi\}_{i=1}^m$ (successor feature parameters) and $\{\theta_i^R\}_{i=1}^m$ (reward models)
Initialize replay buffer $\mathcal{D} = \emptyset$

for each training iteration $t = 1, 2, \dots$ do
    Sample a task index $i \sim \pi(i)$ {Sample current task}
    Sample a state-action pair $(s, a)$ from buffer $\mathcal{D}$ or environment
    Take action $a$ in task $i$, observe reward $r$, next state $s'$, and feature $\phi_t = \phi(s, a, s')$

    if $\mathcal{R}_i$ not provided then
        Update reward model $\tilde{\mathcal{R}}_i$ using $\mathrm{SGD}(\alpha_R)$ on loss $(r - \tilde{\mathcal{R}}_i(\phi_t))^2$

    {Generalized Policy Improvement (GPI)}
    $c \leftarrow \arg\max_{k \in \{1,\dots,m\}} \max_{a'} \sum_{\phi' \in \Phi} \xi_k(s', a', \phi'; \theta_k^\xi)\, \tilde{\mathcal{R}}_i(\phi')$
    With prob. $\epsilon$, choose a random $a_t$; else $a_t \leftarrow \arg\max_a \sum_{\phi \in \Phi} \xi_c(s, a, \phi; \theta_c^\xi) \tilde{\mathcal{R}}_i(\phi)$

    if $s'$ is terminal then
        $\gamma_t \leftarrow 0$
    else
        $\gamma_t \leftarrow \gamma$

    {Update for current task $i$}
    $\hat{a}_{t+1} \leftarrow \arg\max_{a'} \sum_{\phi' \in \Phi} \xi_c(s', a', \phi'; \theta_c^\xi) \tilde{\mathcal{R}}_i(\phi')$
    $\nabla_{\theta_i^\xi} \mathcal{L}_t^{(i)} \leftarrow \sum_{\phi \in \Phi} \Big[\mathbb{I}(\phi = \phi_t) + \gamma_t \xi_i(s', \hat{a}_{t+1}, \phi; \theta_i^\xi) - \xi_i(s, a_t, \phi; \theta_i^\xi)\Big] \times \Big(\gamma_t \nabla_{\theta_i^\xi} \xi_i(s', \hat{a}_{t+1}, \phi; \theta_i^\xi) - \nabla_{\theta_i^\xi} \xi_i(s, a_t, \phi; \theta_i^\xi)\Big)$
    $\theta_i^\xi \leftarrow \theta_i^\xi - \alpha_\xi \nabla_{\theta_i^\xi} \mathcal{L}_t^{(i)}$

    if $c \neq i$ then
        {Update for GPI-selected policy $c$}
        $\hat{a}_{t+1} \leftarrow \arg\max_{a'} \sum_{\phi' \in \Phi} \xi_c(s', a', \phi'; \theta_c^\xi) \tilde{\mathcal{R}}_c(\phi')$
        $\nabla_{\theta_c^\xi} \mathcal{L}_t^{(c)} \leftarrow \sum_{\phi \in \Phi} \Big[\mathbb{I}(\phi = \phi_t) + \gamma_t \xi_c(s', \hat{a}_{t+1}, \phi; \theta_c^\xi) - \xi_c(s, a_t, \phi; \theta_c^\xi)\Big] \times \Big(\gamma_t \nabla_{\theta_c^\xi} \xi_c(s', \hat{a}_{t+1}, \phi; \theta_c^\xi) - \nabla_{\theta_c^\xi} \xi_c(s, a_t, \phi; \theta_c^\xi)\Big)$
        $\theta_c^\xi \leftarrow \theta_c^\xi - \alpha_\xi \nabla_{\theta_c^\xi} \mathcal{L}_t^{(c)}$
    Store $(s, a, r, s', i)$ in $\mathcal{D}$
=0
```

Algorithm 3 Full-Gradient SFRQL with Randomized Task Sampling and Averaging

```
Input: exploration rate $\epsilon$; learning rates $\alpha_\xi$, $\alpha_R$; batch size $N$
Input: features $\Phi$; set of tasks $\{1, \ldots, m\}$
Initialize parameters $\{\theta_i^\xi\}_{i=1}^m$ and replay buffer $\mathcal{D} = \emptyset$

for each training iteration $t = 1, 2, \ldots$ do
    Environment Step:
    Sample task $i \sim \pi(i)$. Take step using $\epsilon$-greedy GPI.
    Store transition $(s, a, r, s', \phi, i)$ in $\mathcal{D}$.

    Averaged Full-Gradient Update:
    Sample a pivot state-action pair $(s_k, a_k)$ from $\mathcal{D}$.
    Retrieve a batch of $N$ transitions $\mathcal{K} = \{(s'_p, \phi_p)\}_{p=1}^N$ from $\mathcal{D}$ that correspond to $(s_k, a_k)$.
    if $|\mathcal{K}| < N$ then continue

    {Compute Averaged Targets}
    Select GPI prior: $c \leftarrow \arg\max_k \max_{a'} \sum_{\phi'} \xi_k(\bar{s}', a', \phi'; \theta_k^\xi)\tilde{\mathcal{R}}_i(\phi')$ using mean next-state $\bar{s}'$.
    Select next action: $\hat{a} \leftarrow \arg\max_{a'} \sum_{\phi'} \xi_c(\bar{s}', a', \phi'; \theta_c^\xi)\tilde{\mathcal{R}}_i(\phi')$.

    {Calculate Averaged Vector Components (Eq. 8)}
    $\bar{\Phi}_{target} \leftarrow \frac{1}{N} \sum_{p=1}^N \mathbb{I}(\phi_p)$
    $\bar{\xi}_{next}^{(j)} \leftarrow \frac{1}{N} \sum_{p=1}^N \xi_j(s'_p, \hat{a}, \cdot; \theta^{(j)})$ for $j \in \{i, c\}$
    $\bar{\nabla}\xi_{next}^{(j)} \leftarrow \frac{1}{N} \sum_{p=1}^N \nabla\xi_j(s'_p, \hat{a}, \cdot; \theta^{(j)})$ for $j \in \{i, c\}$

    {Update Current Task $i$}
    $\delta_i \leftarrow (\bar{\Phi}_{target} + \gamma\bar{\xi}_{next}^{(i)}) - \xi_i(s_k, a_k, \cdot; \theta^{(i)})$
    $\Delta g_i \leftarrow \gamma\bar{\nabla}\xi_{next}^{(i)} - \nabla\xi_i(s_k, a_k, \cdot; \theta^{(i)})$
    $\theta_i^\xi \leftarrow \theta_i^\xi - \alpha_\xi \sum_\phi [2 \cdot \delta_i(\phi) \cdot \Delta g_i(\phi)]$

    if $c \neq i$ then
        {Update Prior Task $c$}
        $\delta_c \leftarrow (\bar{\Phi}_{target} + \gamma\bar{\xi}_{next}^{(c)}) - \xi_c(s_k, a_k, \cdot; \theta^{(c)})$
        $\Delta g_c \leftarrow \gamma\bar{\nabla}\xi_{next}^{(c)} - \nabla\xi_c(s_k, a_k, \cdot; \theta^{(c)})$
        $\theta_c^\xi \leftarrow \theta_c^\xi - \alpha_\xi \sum_\phi [2 \cdot \delta_c(\phi) \cdot \Delta g_c(\phi)]$
=0
```