



Robust Optimal Well Control using an Adaptive Multigrid Reinforcement Learning Framework

Atish Dixit¹ · Ahmed H. Elsheikh¹

Received: 30 March 2022 / Accepted: 14 October 2022 / Published online: 4 November 2022
© The Author(s) 2022

Abstract Reinforcement learning (RL) is a promising tool for solving robust optimal well control problems where the model parameters are highly uncertain and the system is partially observable in practice. However, the RL of robust control policies often relies on performing a large number of simulations. This could easily become computationally intractable for cases with computationally intensive simulations. To address this bottleneck, an adaptive multigrid RL framework is introduced which is inspired by principles of geometric multigrid methods used in iterative numerical algorithms. RL control policies are initially learned using computationally efficient low-fidelity simulations with coarse grid discretization of the underlying partial differential equations (PDEs). Subsequently, the simulation fidelity is increased in an adaptive manner towards the highest fidelity simulation that corresponds to the finest discretization of the model domain. The proposed framework is demonstrated using a state-of-the-art, model-free policy-based RL algorithm, namely the proximal policy optimization algorithm. Results are shown for two case studies of robust optimal well control problems, which are inspired from SPE-10 model 2 benchmark case studies. Prominent gains in computational efficiency are observed using the proposed framework, saving around 60-70% of the computational cost of its single fine-grid counterpart.

Keywords Reinforcement learning · Adaptive · Multigrid framework · Transfer learning · Robust optimal control

✉ Atish Dixit
ad181@hw.ac.uk

Ahmed H. Elsheikh
a.elsheikh@hw.ac.uk

¹ Heriot-Watt University, Edinburgh, UK

1 Introduction

Optimal control problem involves finding controls for a dynamical system such that a certain objective function is optimized over a predefined simulation time. Recently, reinforcement learning (RL) has been demonstrated as an effective method to solve stochastic optimal control problems in fields like manufacturing (Dornheim et al. 2020), energy (Anderlini et al. 2016) and fluid dynamics (Rabault et al. 2019). RL, being virtually a stochastic optimization method, involves a large number of exploration and exploitation attempts to learn the optimal control policy. As a result, the learning process for the optimal policy comprises a large number of simulations of the controlled dynamical system, which is often computationally expensive.

Various research studies have shown the effectiveness of using multigrid methods to improve the convergence rate of reinforcement learning. Anderson and Crawford-Hines (1994) extend Q-Learning by casting it as a multigrid method and has shown a reduction in the number of updates required to reach a given error level in the Q-function. Ziv and Shimkin (2005) and Pareigis (1996) formulate the value function learning process with a Hamilton-Jacobi-Bellman equation (HJB), which is solved using algebraic multigrid methods. However, despite the effectiveness of this strategy, the HJB formulation is only feasible when the model dynamics is well defined. As a result, these methods cannot be applied to problems where the model dynamics is an approximate representation of reality. Li and Xia (2015) used multigrid approach to compute tabular Q values for energy conservation and comfort of HVAC in buildings, which is applicable to certain simple RL problems with finite and discrete state-action space. In this paper, the aim is to present a generalized multigrid RL approach that can be applied to both discrete and continuous state and action space where HJB formulation may not be possible. For instance, when the transition in model dynamics is not necessarily differentiable and/or when the model is stochastic.

In the context of the reinforcement learning literature, the proposed multigrid learning process can be categorized as a framework for transfer learning. In transfer learning, the agent is first trained on one or more source task(s), and the acquired knowledge is then transferred to aid in solving the desired target task (Taylor and Stone 2009). In the presented study, the highest fidelity simulation corresponds to the target task, which is assumed to have the fine-grid discretization. The fine-grid discretization is presumed to guarantee a good approximation of the output quantities of interest with the accuracy required by the problem at hand. On the other hand, low-grid-fidelity simulations that compromise the accuracy of these quantities correspond to source tasks. These low-grid-fidelity simulations are generated using a degree-of-freedom parameter called the grid fidelity factor (much like the study by Narvekar et al. (2016)). Transfer learning is a much broader subdomain of RL that covers knowledge transfer in the form of data samples (Lazaric et al. 2008), policies (Fernández et al. 2010), models (Fachantidis et al. 2013), or value functions (Taylor and Stone 2005). In this study, knowledge transfer is done in the form of a policy for a model-free, on-policy algorithm called proximal policy optimization (PPO). Since the policy is designed for the state and actions corresponding to the highest-fidelity simulation, a predefined mapping function is used, which maps states and actions from low-fidelity simulations to high-fidelity simulations, and vice versa. This is done by defining restriction (map-

ping from high- to low-fidelity simulation) and prolongation (mapping from low- to high-fidelity simulation) operators, which are normally found in classical geometric multigrid methods.

The effectiveness of this multigrid RL framework is demonstrated for the robust optimal well control problem, which is a subject of intensive research activities in subsurface reservoir management (van Essen et al. 2009; Roseta-Palma and Xepapadeas 2004; Brouwer et al. 2001). Recently, several researchers have proposed the use of reinforcement learning to solve the optimal well control problem (Miftakhov et al. 2020; Nasir et al. 2021; Dixit and ElSheikh 2022). For this study, the dynamical system under consideration is non-linear and, in practice, is partially observable since the data is only available at a sparse set of points (i.e., well locations). Furthermore, the subsurface model parameters are highly uncertain due to the sparsity of available field data. Optimal well control problem consists of optimizing the control variables like valve openings of wells in order to maximize sweep efficiency of injector fluid throughout the reservoir life. The reservoir permeability field is considered as an uncertain model parameter for which the uncertainty distribution is known. Two test cases – both representing a distinct model parameter uncertainty and control dynamics – are used to demonstrate the computational gains of using the multigrid idea.

In summary, a multigrid reinforcement learning framework is proposed to solve the optimal well control problem for subsurface flow with uncertain parameters. This framework is essentially inspired by the principles of geometric multigrid methods used in iterative numerical algorithms. The optimal policy learning process is initiated using a low-fidelity simulation that corresponds to a coarse grid discretization of the underlying partial differential equations (PDEs). This learned policy is then reused to initialize training against a high-fidelity simulation environment in an adaptive and incremental manner. That is, the shifting from a low fidelity to higher fidelity environments is done adaptively after the convergence of the learned policy with the low fidelity environment. Due to this adaptive learning strategy, most of the initial policy learning takes place against lower-fidelity environments, yielding a minimal computational cost in the initial stages (significant part) of the reinforcement learning process. Robustness of the policy learned using this framework is finally evaluated against uncertainties in the model dynamics.

The outline of the remainder of this paper is as follows. Section 2 provides the description of the problem and the proposed framework to solve the robust optimal well control problem. Section 3 details the model parameters for the two case studies designed for demonstration. Results of the proposed framework on these two case studies are demonstrated in Sect. 4. Finally, Sect. 5 concludes with a summary of the research study and an outlook on future research directions.

2 Methodology

Fluid flow control in subsurface reservoirs has many engineering applications, ranging from the financial aspects of efficient hydrocarbon production to the environmental problems of contaminant removal from polluted aquifers (Whitaker 1999). In this paper, a canonical single-phase subsurface flow control problem (also referred to as

robust optimal well control problem) is studied where water is injected in porous media to displace a contaminant. This process is commonly modeled using an advection equation for tracer flow through porous media (also called Darcy flow through porous media) over the temporal domain $\mathcal{T} = [t_0, t_M] \subset \mathbb{R}$ and spatial domain $\mathcal{X} \subset \mathbb{R}^2$. In the context of fluid displacement (e.g., groundwater decontamination), the tracer corresponds to clean water injected in the reservoir from the injector wells and the non-traced fluid corresponds to the displaced contaminated water from the reservoir through producer wells. The source and sink locations within the model domain correspond to the injector and producer wells, respectively. Tracer flow models water flooding with the fractional variable $s(x, t) \in [0, 1]$ (also known as saturation). Saturation $s(x, t)$, represents the fraction which is calculated as the ratio of injected clean water mass to the displaced contaminated water mass at location $x \in \mathcal{X}$ and time $t \in \mathcal{T}$. The flow of fluid in and out of the domain is represented by $a(x, t)$, which is treated as the source / sink terms of the governing equation. The set of well locations is denoted as $x' \in \mathcal{X}'$ (where $\mathcal{X}' \subset \mathcal{X}$). In other words, $a(x, t)$ is assigned to zero everywhere in the domain \mathcal{X} except the set of locations x' . The controls $a^+(x, t)$ (formulated as $\max(0, a(x, t))$) and $a^-(x, t)$ (formulated as $\min(0, a(x, t))$) represent the injector and producer flow controls, respectively (note that $a = a^+ + a^-$). The task of the problem under consideration is to find optimal controls $a^*(x', t)$, which is the solution of the closed-loop optimization problem defined as

$$\max_{s(\cdot), a(\cdot)} \int_{t_0}^{t_M} \left(\sum_{x'} a^-(x', t)(1 - s(x', t)) \right) dt, \quad x' \in \mathcal{X}', t \in \mathcal{T} \quad (1a)$$

$$\frac{ds}{dt} = \frac{1}{\phi} (a^+ + sa^- - \nabla \cdot sv), \quad x \in \mathcal{X}, t \in \mathcal{T} \quad (1b)$$

$$s(\cdot, t_0) = s_0, \quad v \cdot \mathbf{n} = 0, \quad (1c)$$

$$\sum_{x'} a^+(x', t) = - \sum_{x'} a^-(x', t) = c. \quad x' \in \mathcal{X}', t \in \mathcal{T} \quad (1d)$$

The objective function defined in Eq. 1a represents the total flow of fluid displaced from the reservoir (for example, contaminated water production) and is maximized over a finite time interval \mathcal{T} . The integrand in this function is referred to as Lagrangian term in control theory and is often denoted by $L(s, a)$. The water flow trajectory $s(x, t)$, is governed by advection Eq. 1b which is solved given the velocity field v , which is obtained from the Darcy law: $v = -(k/\mu)\nabla p$. The pressure $p(x, t) \in \mathbb{R}$, is obtained from the pressure equation $-\nabla \cdot (k/\mu)\nabla p = a$. Porosity $\phi(x, \cdot)$, permeability $k(x, \cdot)$, and viscosity $\mu(x, \cdot)$ are the model parameters. Permeability k , represents the model uncertainty and is treated as a random variable that follows a known probability density function \mathcal{K} with K as its domain. The initial and no-flow boundary conditions are defined in Eq. 1c, where \mathbf{n} denotes outward normal vector from the boundary of \mathcal{X} . The constraint defined in Eq. 1d represent the fluid incompressibility assumption along with the fixed total source/sink term c , which represents total water injection rate in the reservoir. In a nutshell, the optimization problem provided in Eqs. 1 is solved to

find the optimal controls $a^*(x', t)$ such that they are robustly optimal over the entire uncertainty domain of permeability, K .

2.1 RL Framework

According to RL convention, the optimal control problem defined in Eq. 1 is modelled as a Markov decision process, which is formulated as a quadruple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$. Here, $\mathcal{S} \subset \mathbb{R}^{n_s}$ is a set of all possible states with the dimension n_s , $\mathcal{A} \subset \mathbb{R}^{n_a}$ is a set of all possible actions with the dimension n_a . The state S , is represented with the saturation $s(x, \cdot)$ and pressure $p(x, \cdot)$ values over the entire domain \mathcal{X} . The action A , is represented by an array of well control values $a(x', \cdot)$. More details of this array, like the representation of action, are presented in Sect. 3.3. The optimal control problem defined in Eq. 1 is discretized into M control steps and as a result, its solution is a set of optimal control values $a^*(x', t_1), a^*(x', t_2), \dots, a^*(x', t_M)$ where $t_0 < t_1 < t_2 < \dots < t_M$. The transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, is assumed to follow the Markov property. That is, transition to the state $S(t_{m+1})$ is obtained by executing the actions $A(t_m)$ when in the state $S(t_m)$. Such transition function is obtained by discretizing Eq. 1b. For a transition from state $S(t_m)$ to state $S(t_{m+1})$, the real value reward $R(t_{m+1})$ is calculated as $R(t_{m+1}) = \mathcal{R}(S(t_m), A(t_m), S(t_{m+1}))$, where $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the reward function. The reward function is obtained by discretizing the objective function (Eq. 1a) into control steps such that

$$R(t_{m+1}) = \int_{t_m}^{t_{m+1}} L(s, a) dt. \quad (2)$$

Optimal controls are obtained by learning a control policy function, which is defined as $\pi : \mathcal{S} \rightarrow \mathcal{A}$. This function is denoted as $\pi(A|S)$ and is generally represented by a neural network. Essentially, the control policy $\pi(A|S)$, maps a given state $S(t_m)$, to an action $A(t_m)$. For an optimal control problem, with M control steps, the goal of reinforcement learning is to find an optimal policy $\pi^*(A|S)$ such that the expected reward $G = \sum_{m=1}^M \gamma^{m-1} R(t_m)$, is maximized. Note that immediate rewards R , are exponentially decayed by the discount rate $\gamma \in [0, 1]$. The discount rate represents how myopic the learned policy is; for example, a learned policy is considered completely myopic when $\gamma = 0$. The controller, which is also referred to as an agent, follows the policy and explores various control trajectories by interacting with the environment, which consists of a transition function \mathcal{P} and a reward function \mathcal{R} . The data gathered by these control trajectories are used to update the policy towards optimality. Each such update of the policy is called a policy iteration. In RL literature, a single complete control trajectory is referred to as an episode. Essentially, RL algorithms attempt to learn the optimal policy $\pi^*(A|S)$ from a randomly initialized policy $\pi(A|S)$, by exploring the state-action space by executing a high number of episodes.

In order to represent the variability in permeability, a finite number of well spread samples is chosen from the predefined uncertainty distribution. This is achieved with a cluster analysis (see Appendix 1 for the formulation of cluster analysis) of the distribution domain K . The sample vector $\mathbf{k} = \{k_1, k_2, \dots, k_l\}$, is constructed with

samples of the distribution \mathcal{K} , which are located nearest to the cluster centres. The policy $\pi^*(A|S)$, is learned by randomly selecting the parameter k from the training vector \mathbf{k} at the beginning of each episode. The policy return $R^{\pi(A|S)}$, is computed by averaging the returns of policy $\pi(A|S; k_i)$ (policy applied to the simulation where the permeability is set to k_i) in l simulations, which is formulated as

$$R^{\pi(A|S)} = \frac{1}{l} \sum_{i=1}^l \sum_{m=0}^{M-1} \int_{t_m}^{t_{m+1}} L(s, \pi(A|S; k_i)) dt. \quad (3)$$

In optimal well control problems, the system is partially observable; that is, reservoir information is only available at well locations throughout the reservoir life cycle. To accommodate this fact, the agent is provided with the available observation as its state. For this study, observation is represented with a set of saturation and pressure values at the well locations x' . This is also apparent in the definition of Lagrangian term $L(s, a)$, where values s and a are taken at well locations x' , as defined in Eq. 1a. Note that with such a representation of states, the underlying assumption of the Markov property of the transition function is approximated. Such system is referred to as partially observable Markov decision process (POMDP). By the definition of POMDP, the policy requires the observations and actions of all previous control steps to return the action for a certain control step. However, for the presented case studies, observation from only the previous control step is observed to be sufficient for policy representation.

2.2 Learning Convergence Criteria

The optimal policy convergence is detected by monitoring the policy return $R^{\pi(A|S)}$, after every policy iteration. Conventionally, when this value converges to a maximum value, the optimal policy is assumed to be learned. The convergence criteria for i th policy iteration is defined as

$$\delta_i = \left| \frac{R_i^{\pi(A|S)} - R_{i-1}^{\pi(A|S)}}{\max(R_{i-1}^{\pi(A|S)}, \epsilon)} \right| < \delta, \quad (4)$$

where δ_i is the return tolerance at i th policy iteration, δ is the stopping tolerance and ϵ is a small non-zero number used to avoid division by zero. The convergence of policy learning process is often flat near the optimal result. For this reason, the convergence criteria defined in Eq. 4 is checked for the last n consecutive policy iterations. For example, if \mathbf{r} is the array of monitored values of $R^{\pi(A|S)}$ at all policy iterations, the policy $\pi(A|S)$ is considered converged when the convergence criteria are met (Eq. 4) for last n policy iterations is met. Algorithm 1 delineates the pseudocode for this convergence criteria.

Figure 1 illustrates the effect of n and δ on the convergence criteria for an example of a reinforcement learning process.

Algorithm 1 Learning convergence criteria

```
1: procedure ISCONVERGED( $\mathbf{r}, n, \delta$ )
2:   if  $length(\mathbf{r}) < n$  then return False
3:   end if
4:   compute  $\delta_i$  (Eq. 4) for last  $n$  values of  $\mathbf{r}$  and get its maximum  $\delta_{max}$ 
5:   if  $\delta_{max} < \delta$  then
6:     return True
7:   else
8:     return False
9:   end if
10: end procedure
```

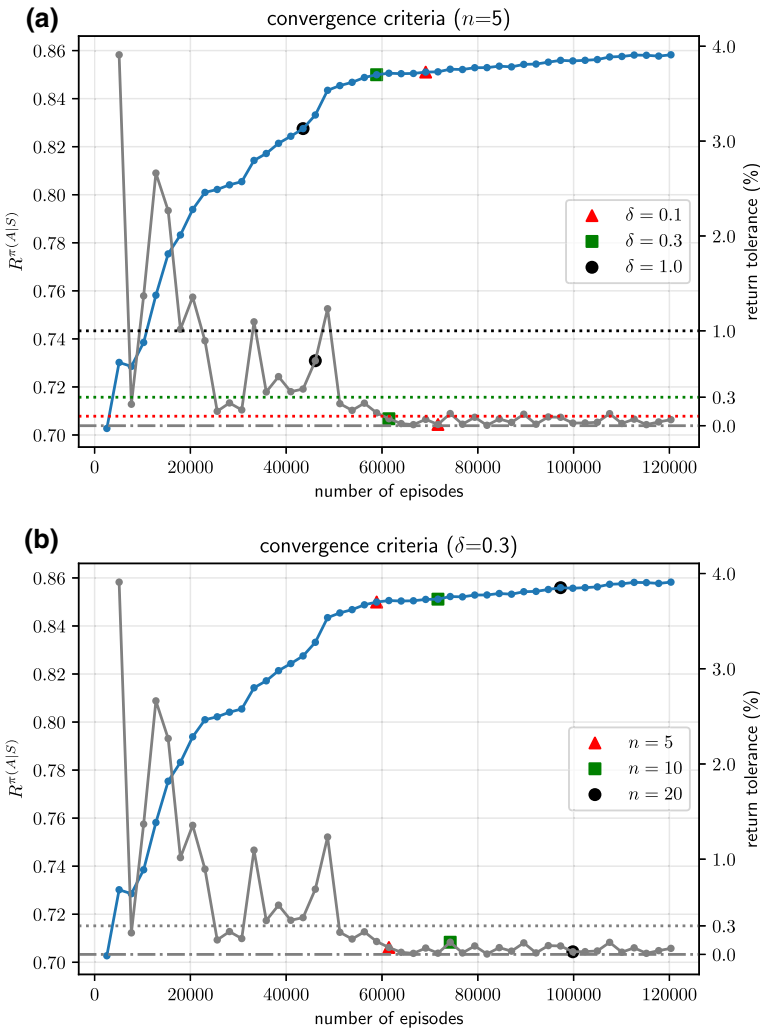


Fig. 1 Plot of policy returns versus number of training episodes: **a** illustrates effect of δ on convergence criteria and **b** illustrates effect of n on convergence criteria

The policy return plot is shown in blue, where each value at the policy iteration is shown with a dot. The corresponding return tolerance is plotted in gray color, which is represented in percentage format ($\delta_i \times 100$, where δ_i is calculated from Eq. 4). It can be seen that the convergence criteria (denoted with markers on these plots) takes longer to satisfy when the stopping tolerance δ , is smaller and consecutive policy iteration steps n , are higher.

2.3 Adaptive Multigrid RL Framework

An adaptive multigrid RL framework is proposed where, essentially, the policies learned using lower grid fidelity environments are transferred and trained with higher-fidelity environments. The fidelity of the grid for an environment is described by the factor $\beta \in (0, 1]$. The environment with $\beta = 1$ is assumed to have the fine-grid discretization, which guarantees good approximation of fluid flow production out of the domain as defined in Eq. 1a. For any environment where $\beta < 1$, the size of the environment grid is coarsened by the factor of β . For example, if a high-fidelity environment where $\beta = 1$ corresponds to the simulation with grid size 64×64 , the simulation grid size is reduced to 32×32 when β is set to 0.5. Restriction operator $\Phi_\beta(\cdot)$, is used to coarsen the high fidelity simulation parameters with the factor of β . This is done by partitioning a finer grid of size $m \times n$ (corresponding to $\beta = 1$) into coarser dimensions $\lfloor \beta m \rfloor \times \lfloor \beta n \rfloor$ (corresponding to $\beta < 1$ where $\lfloor \cdot \rfloor$ is the floor operator) and computing these values of the coarse grid cell as a function \mathbf{f} , of the values in the corresponding partition. Figure 2a illustrate this restriction operator for a variable $x \in \mathbb{R}^{n \times m}$.

The function \mathbf{f} , for different parameters of the reservoir simulation, are listed in Table 1.

On the other hand, the prolongation operator $\Phi_\beta^{-1}(\cdot)$, maps the coarse grid environment parameters to the fine grid as shown in Fig. 2b.

A typical agent-environment interaction using this framework is illustrated in Fig. 3. Note that the transition function \mathcal{P} and the reward function \mathcal{R} are subscripted with β to indicate the grid fidelity of the environment. State $S(t_m)$, action $A(t_m)$ and reward $R(t_m)$ are denoted with shorthand notations, S_m , A_m and R_m , respectively. Throughout the learning process, the policy is represented with states and actions corresponding to the high-fidelity grid environment. As a result, actions, and states to and from the environment, undergo the restriction Φ_β and prolongation Φ_β^{-1} operations at each time-step as shown in the environment box in the Fig. 3.

The proposed framework is demonstrated for PPO algorithm. PPO (Schulman et al. 2017) is a policy gradient algorithm that models stochastic policy $\pi_\theta(A|S)$, with a neural network (also known as the actor network). Essentially, the network parameters θ , are obtained by optimizing the objective function defined as

$$J_{\text{ppo}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) A \hat{v}(S_t, A_t), \right. \right. \\ \left. \left. \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) A \hat{v}(S_t, A_t) \right) \right], \quad (5)$$

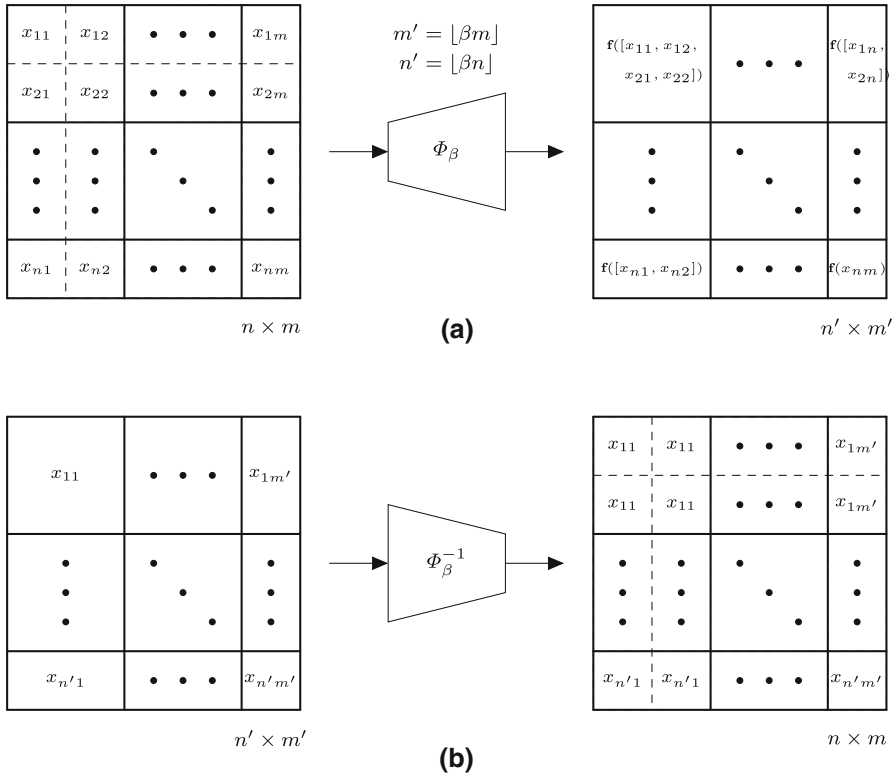


Fig. 2 Illustration for the restriction operator Φ_β (a) and prolongation operator Φ_β^{-1} (b) for a parameter x

Table 1 Restriction operator function for simulation parameters

Simulation parameter	Function, f
Saturation, s	Mean
Porosity, ϕ	Mean
Pressure, p	Mean
Permeability, k	Harmonic mean
Flow control, a	Sum

where $r_t(\theta) = \pi_\theta(A_t|S_t)/\pi_{\theta_{old}}(A_t|S_t)$ and θ_{old} correspond to the policy parameters before the policy update. The advantage function estimator $A\hat{d}v_t$, is calculated using the generalized advantage estimator (Schulman et al. 2015) derived from the value function V_t . The value function estimator \hat{V}_t is learned through a separate neural network, termed as the critic network. The definitions of advantage and value functions are provided in Appendix 2. In practice, a single neural network is used to represent both the actor and critic networks. The objective function for this integrated actor-critic network is the sum of the actor loss term (Eq. 5), value loss term and entropy loss term. For the purpose of maintaining brevity in our description, these latter loss

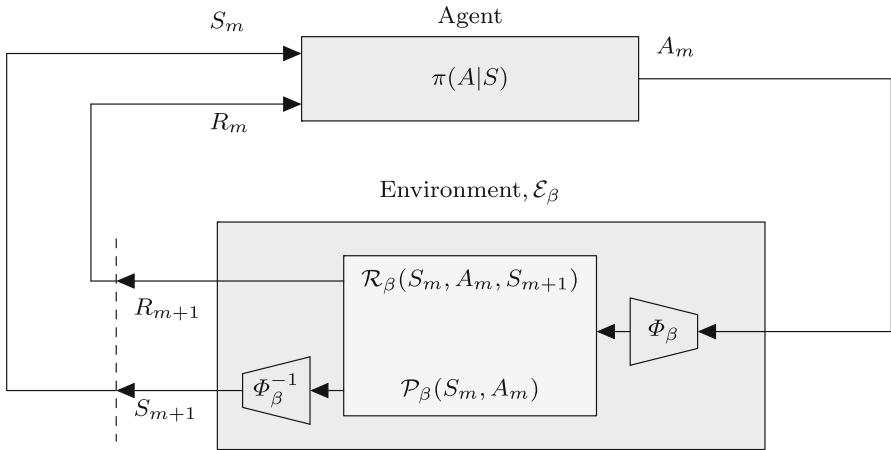


Fig. 3 A typical agent-environment interaction in the proposed multigrad RL framework

terms are omitted and the policy network’s objective function is treated as $J_{ppo}(\theta)$ in further discussion. However, please note that they are considered while executing the framework. The readers are referred to Schulman et al. (2017) for a detailed definition of the policy network loss term. The Algorithm 2 presents the pseudocode for the proposed multigrad RL framework.

Algorithm 2 PPO with adaptive multigrad framework

```

1: Define  $\delta, n$  and an empty array  $\mathbf{r}$  for convergence criteria
2: Define a grid fidelity factor array  $\beta = [\beta_1, \beta_2, \dots, \beta_m]$ , where  $\beta_m = 1$  and  $\beta_1 < \beta_2 < \dots < \beta_m$ .
3: Define an episode limit array  $\mathbf{E} = [E_1, E_2, \dots, E_m]$ , where  $E_1 < E_2 < \dots < E_m$ .
4: Define total episode count,  $e = 0$ 
5: for  $i = 1, 2, \dots, m$  do
6:   Generate the environment  $\mathcal{E}_{\beta_i}$ , with the grid fidelity factor  $\beta_i$ 
7:   for  $iteration = 1, 2, \dots$  do
8:     for  $actor = 1, 2, \dots, N$  do
9:       Run policy  $\pi_{\theta_{old}}$  in environment  $\mathcal{E}_{\beta_i}$ , for  $T$  time steps (in total,  $E$  episodes)
10:      Compute value function estimates  $\hat{V}_1, \dots, \hat{V}_T$  using critic network
11:      Compute advantage function estimates  $\hat{A}_1, \dots, \hat{A}_T$ 
12:    end for
13:    Optimize  $J_{ppo}(\theta)$  with  $K$  epochs and minibatch size  $M \leq NT$ 
14:     $\theta_{old} \leftarrow \theta$ 
15:    Compute the policy return  $R^{\pi_{\theta}(A|S)}$  and append it in  $\mathbf{r}$ 
16:     $e := e + E$ 
17:    if  $\text{IsConverged}(\mathbf{r}, n, \delta)$  or  $e \geq E_i$  then
18:      break
19:    end if
20:  end for
21: end for

```

The framework consists of, in total, m values of grid fidelity factor which are represented with an array $\beta = [\beta_1, \beta_2, \dots, \beta_m]$, where $\beta_m = 1$ and $\beta_1 < \beta_2 < \dots <$

Table 2 Reservoir model parameters

	Case 1	Case 2	Units
Spatial domain \mathcal{X}	(1200×1200)	(620×1820)	ft ²
Temporal domain \mathcal{T}	[0,125]	[0,25]	days
Initial saturation s_0	0.0	0.0	–
Viscosity μ	0.3	0.3	cP
Porosity ϕ	0.2	0.2	–
Number of producers n_p	31	14	–
Number of injectors n_i	31	7	–
Total injector flow $\sum a^+$	2304	9072	ft ² /day

β_m . The environment is denoted as \mathcal{E}_{β_i} , which represents the environment with the grid fidelity factor β_i . Policy $\pi_{\theta}(A|S)$ is initially learned with the environment \mathcal{E}_{β_1} , until the convergence criteria are met. The convergence criteria are checked using the Algorithm 1 with predefined parameters δ and n . Upon convergence, further policy iterations are learned using the environment \mathcal{E}_{β_2} , and so on until the convergence criteria are met for the highest grid fidelity environment \mathcal{E}_{β_m} . A limit for the number of episodes to be executed at each grid level is also set. This is done by defining an episode limit array $\mathbf{E} = [E_1, E_2, \dots, E_m]$, where E_m is the total number of episodes to be executed and $E_1 < E_2 < \dots < E_m$. That is, for every environment with grid fidelity factor, β_j the maximum number of episodes to be trained is limited to E_j .

3 Case Studies

Two test cases are designed, representing two distinct uncertainty distributions of permeability and control dynamics. For both cases, the values for model parameters emulate those in the benchmark reservoir simulation cases, SPE-10 model 2 (Christie et al. 2001). Table 2 delineates these values for test cases 1 and 2. As per the convention in geostatistics, the distribution of $\log(k)$ is assumed to be known and is denoted by \mathcal{G} . As a result, $g = \log(k)$ is treated as a random variable in the problem description defined in Eq. 1. The uncertainty distributions for test cases 1 and 2 are indicated with \mathcal{G}_1 and \mathcal{G}_2 , respectively.

3.1 Uncertainty Distribution for Test Case 1

The log-permeability uncertainty distribution for test case 1 is inspired by the case study of Brouwer et al. (2001). Figure 4a shows schematics of the spatial domain for this case. In total, 31 injector wells (illustrated with blue circles) and 31 producer wells (illustrated with red circles) are placed on the left and right edges of the domain, respectively. As illustrated in Fig. 4a, a linear high-permeability channel (shown in gray) passes from the left to right side of the domain.

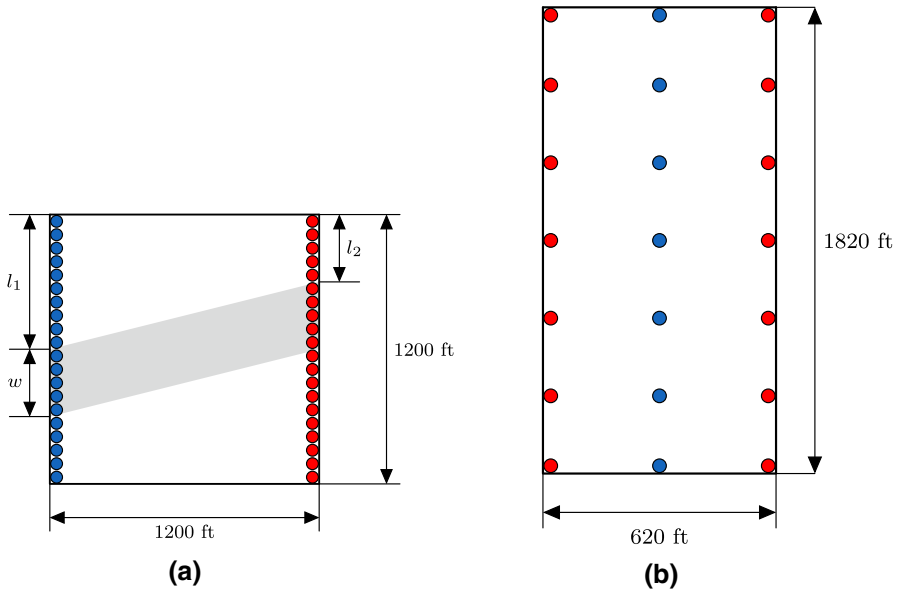


Fig. 4 Schematic of the spatial domain for test case 1 (a) and 2 (b)

l_1 and l_2 represent the distance from the upper edge of the domain on the left and right sides, while the width of the channel is indicated by w . These parameters follow uniform distributions defined as $w \sim U(120, 360)$, $l_1 \sim U(0, L - w)$ and $l_2 \sim U(0, L - w)$, where L is the domain length. In other words, the random variable g follows the probability distribution \mathcal{G}_1 that is parameterized with w, l_1 and l_2 which is described as

$$g \sim \mathcal{G}_1(w, l_1, l_2).$$

To be specific, the log-permeability g at a location (x, y) is formulated as

$$g(x, y) = \begin{cases} \log(245) & \text{if } \frac{l_2-l_1}{L}x + l_1 \leq y \leq \frac{l_2-l_1}{L}x + l_1 + w, \\ \log(0.14) & \text{otherwise,} \end{cases}$$

where x and y are horizontal and vertical distances from the upper left corner of the domain illustrated in Fig. 4a. The values for permeability at the channel (245 mD) and the rest of the domain (0.14 mD) are inspired from Upperness log-permeability distribution peak values specified in SPE-10 model 2 case.

3.2 Uncertainty Distribution for Test Case 2

Test case 2 represents the uncertainty distribution of a smoother permeability field. Figure 4b illustrates reservoir domain for this case. It comprises 14 producers (illus-

trated with red circles) located symmetrically on the left and right edges (7 on each edge) of the domain and 7 injectors (illustrated with blue circles) located at the central vertical axis of the domain. A prior distribution F is assumed on all locations $x \in \mathcal{X}$ as

$$\begin{aligned} F(x) &= \mu + Z(x), \text{ where,} \\ \mathbb{E}(Z(x)) &= 0, \\ \text{Cov}(Z(x), Z(\tilde{x})) &= \sigma^2 k(x, \tilde{x}), \end{aligned} \tag{6}$$

where the process variance, σ , is assigned as 5 and the exponential covariance function (kernel), $k(x, \tilde{x})$, is defined as

$$k(x, \tilde{x}) = \exp \left[- \left(\frac{(x_1 - \tilde{x}_1)^2}{l_1^2} + \frac{(x_2 - \tilde{x}_2)^2}{l_2^2} \right)^{1/2} \right],$$

where the parameters l_1 and l_2 are assigned to be 620ft (width of the domain) and 62ft (10% of domain width), respectively. The posterior distribution given the observed log-permeability vector, $\mathbf{g}(x') = [g(x'_1), g(x'_2), \dots, g(x'_n)]$, where each observation corresponds to a log-permeability value of 2.41 at a well location (that is, $n = 21$ since there are, in total, 21 wells in this case). From the principle of ordinary kriging, the posterior distribution, \mathcal{G}_2 , for log-permeability at a location $x \in \mathcal{X}$ is a normal distribution which is defined as

$$\begin{aligned} g(x) &\sim \mathcal{G}_2(\hat{g}(x), \hat{s}^2(x)), \text{ where,} \\ \hat{g}(x) &= \hat{\mu} + \mathbf{k}(x', x)^\top \mathbf{k}(x', x')^{-1} (\mathbf{g}(x') - \mathbf{1}\hat{\mu}), \\ \hat{s}^2(x) &= \sigma^2 \left[1 - \mathbf{k}(x', x)^\top \mathbf{k}(x', x')^{-1} \mathbf{k}(x', x) \right. \\ &\quad \left. + \frac{(1 - \mathbf{1}^\top \mathbf{k}(x', x')^{-1} \mathbf{k}(x', x))^2}{\mathbf{1}^\top \mathbf{k}(x', x')^{-1} \mathbf{1}} \right], \end{aligned}$$

where $\mathbf{k}(x', x)$ is the n dimensional vector whose i th value is $k(x'_i, x)$, $\mathbf{k}(x', x')$ is the $n \times n$ dimensional matrix whose value at (i, j) is $k(x'_i, x'_j)$, $\mathbf{1}$ is a n dimensional vector with all elements of one ($\mathbf{1} = [1, 1, \dots, 1]^\top$) and $\hat{\mu}$ is an estimate of the global mean μ , which is obtained from the kriging model based on the maximum likelihood estimate of the distribution $F(x)$ (from Eq. 6) for the observations $\mathbf{g}(x')$, and is formulated as

$$\hat{\mu} = \frac{\mathbf{1}^\top \mathbf{k}(x', x')^{-1} \mathbf{g}(x')}{\mathbf{1}^\top \mathbf{k}(x', x')^{-1} \mathbf{1}}.$$

The log-permeability distribution \mathcal{G}_2 , is created with an ordinary kriging model using the geostatistics library `gstools` (Müller and Schüler 2019). In the simulation, samples of the permeability fields are obtained with a clockwise rotation angle of $\pi/8$.

3.3 State, Action and Reward Formulation

PPO algorithm attempts to learn the parameters θ of the policy neural network $\pi_\theta(A|S)$. The episodes (i.e., the entire simulation in the temporal domain \mathcal{T}) are divided into five control steps. Each episode timestep corresponding to a control step is denoted with t_m , where $m \in \{1, 2, \dots, 5\}$. The state S , is represented by an observation vector that consists of saturation and pressure values at well locations x' . Since the saturation values in the injector wells are always one, regardless of time t_m , they are omitted from the observation vector. Consequently, the observation vector is of the size $2n_p + n_i$ (i.e., $n_s = 93$ for test case 1 and $n_s = 35$ for test case 2). Note that this observation vector forms the input to the policy network $\pi_\theta(A|S)$. A vector of flow control values of all the injector and producer wells, denoted by A , is represented as the action. The action vector A consists of in total $n_p + n_i$ values (that is, $n_a = 62$ for test case 1 and $n_a = 21$ for test case 2). To maintain constraints defined in Eq. 1d, the action vector is represented by a weight vector $w \in \mathbb{R}^{n_a}$, such that $0.001 \leq w_j \leq 1$. Each weight value w_j , corresponds to the proportion of flow through the j th well. As a result, the values in the action vector are written as $(w_1, \dots, w_{n_i}, w_{n_i+1}, \dots, w_{n_i+n_p})$. Flow through j th injector A_j , is computed such that the constraint defined in Eq. 1d is satisfied

$$A_j = -\frac{w_j}{\sum_{i=j}^{n_i} w_j} c.$$

Similarly, the flow through the j th producer, A_{j+n_i} , is calculated as

$$A_{j+n_i} = \frac{w_{j+n_i}}{\sum_{j=1}^{n_p} w_{j+n_i}} c.$$

The reward function, as defined in Eq. 2, is divided by total pore volume ($\phi \times l_x \times l_y$) as a form of normalization to obtain a reward function in the range $[0, 1]$. The normalized reward represents the recovery factor or the sweep efficiency of the contaminated fluid. Recovery factor represents the total amount of contaminants swept out of the domain. For example, the recovery factor of 0.65 means that 65% of contaminants are swept out of the domain using water flooding. To put it in the context of a groundwater decontamination problem, the optimal controls correspond to the well controls that maximize the percentage of contaminants swept out of the reservoir.

3.4 Multigrid Framework Formulations

The proposed framework is demonstrated using three levels of simulation grid fidelity. Note that $\beta = 1$ case corresponds to the finest level of simulation, which provides an accurate estimate of recovery factor. In this study, the aim is to exploit the simulations with $\beta < 1$, which correspond to computationally cheaper simulation run times by definition. Furthermore, $\beta < 1$ simulations are deliberately chosen such that they correspond to a recovery factor estimate that deviates from accurate $\beta = 1$ simulation

Table 3 Grid fidelity factor and corresponding grid size

	Test case 1	Test case 2
$\beta = 1$	61×61	31×91
$\beta = 0.5$	30×30	15×45
$\beta = 0.25$	15×15	7×22

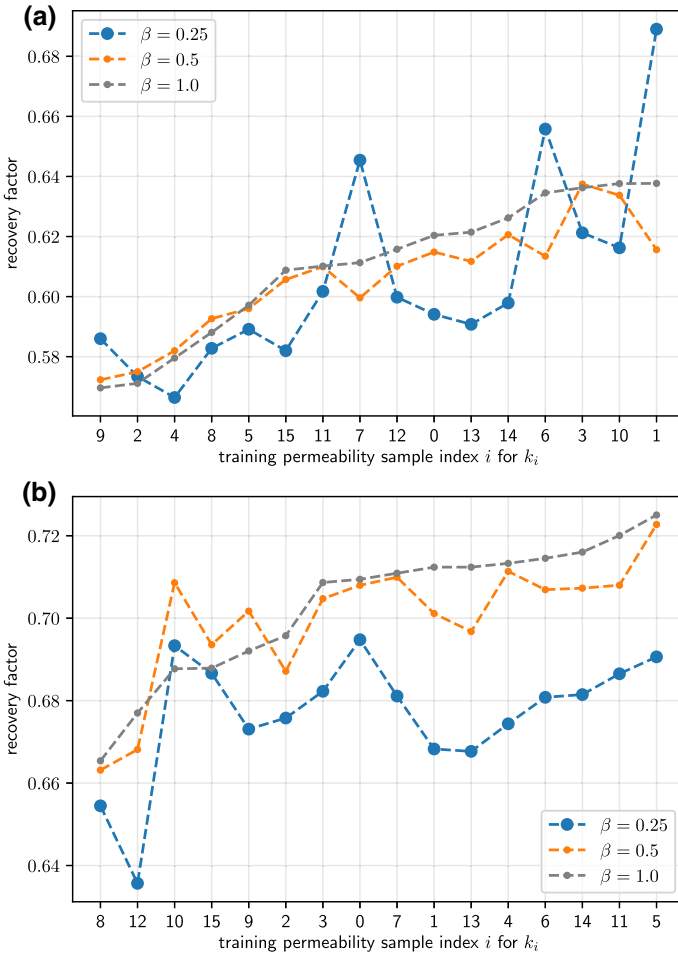


Fig. 5 Comparison of recovery factor estimates with $\beta = 1$, $\beta = 0.5$ and $\beta = 0.25$ for test case 1 (a) and test case 2 (b)

estimates. Table 3 lists the discretization grid size corresponding to these grid fidelity factors for both test cases.

Figures 5a and b plot the recovery factor estimates (with all wells open equally) for these grid fidelity factors for each permeability sample k_i , in test cases 1 and 2, respectively.

Table 4 Multigrid framework experiments

	Test case 1	Test case 2
Single grid ($\beta = 0.25$)	$\beta = [0.25]$	$\beta = [0.25]$
	$\mathbf{E} = [75, 000]$	$\mathbf{E} = [150, 000]$
	$n = \infty; \delta = 0$	$n = \infty; \delta = 0$
Single grid ($\beta = 0.5$)	$\beta = [0.5]$	$\beta = [0.5]$
	$\mathbf{E} = [75, 000]$	$\mathbf{E} = [150, 000]$
	$n = \infty; \delta = 0$	$n = \infty; \delta = 0$
Single grid ($\beta = 1.0$)	$\beta = [1.0]$	$\beta = [1.0]$
	$\mathbf{E} = [75, 000]$	$\mathbf{E} = [150, 000]$
	$n = \infty; \delta = 0$	$n = \infty; \delta = 0$
Fixed multigrid	$\beta = [0.25, 0.5, 1.0]$	$\beta = [0.25, 0.5, 1.0]$
	$\mathbf{E} = [25, 000, 50, 000, 75, 000]$	$\mathbf{E} = [50, 000, 100, 000, 150, 000]$
	$n = \infty; \delta = 0$	$n = \infty; \delta = 0$
Adaptive multigrid	$\beta = [0.25, 0.5, 1.0]$	$\beta = [0.25, 0.5, 1.0]$
	$\mathbf{E} = [25, 000, 50, 000, 75, 000]$	$\mathbf{E} = [50, 000, 100, 000, 150, 000]$
	$n = 25; \delta = 0.2$	$n = 25; \delta = 0.2$

The deviation from the accurate recovery factor (that is, for $\beta = 1$) for $\beta = 0.5$ and $\beta = 0.25$ can be seen for both cases. As expected, the recovery factor estimates with $\beta = 0.25$ show a higher deviation from the estimates of $\beta = 1$ compared to those of $\beta = 0.5$. To show the effectiveness of the proposed framework, the obtained results are compared with single-grid and multigrid frameworks. The results for a single-grid framework are the same as if they were obtained using the classical PPO algorithm, where the environment has a fixed fidelity factor throughout the policy-learning process. This is done by setting the grid fidelity factor β , and episode limit array \mathbf{E} , with a single value in Algorithm 2. The factor n in convergence criteria procedure (delineated in Algorithm 1) is set to infinity. In other words, convergence criteria is unchecked and the policy learning takes place for a predefined number of episodes. In total, three such single-grid experiments are performed corresponding to $\beta = 0.25$, $\beta = 0.5$, and $\beta = 1.0$. In addition, two multigrid experiments are performed to demonstrate the effectiveness of the proposed framework. The first multigrid experiment is referred to as fixed, where convergence criteria are kept unchecked just like single-grid frameworks. Multiple levels of grids are defined by setting the grid fidelity factor array β , and episode limit array \mathbf{E} , as an array of multiple values corresponding to each fidelity factor value and its corresponding episode count. In the fixed multigrid framework, policy learning takes place by updating the fidelity factor of the environment according to β without checking the convergence criteria (i.e., by setting $n = \infty$). Second, the parameters of the adaptive multigrid framework are set similarly to those used in the fixed multigrid framework, except for the convergence criteria parameters n and δ . Table 4 delineates the number of experiments and their corresponding parameters for test cases 1 and 2.

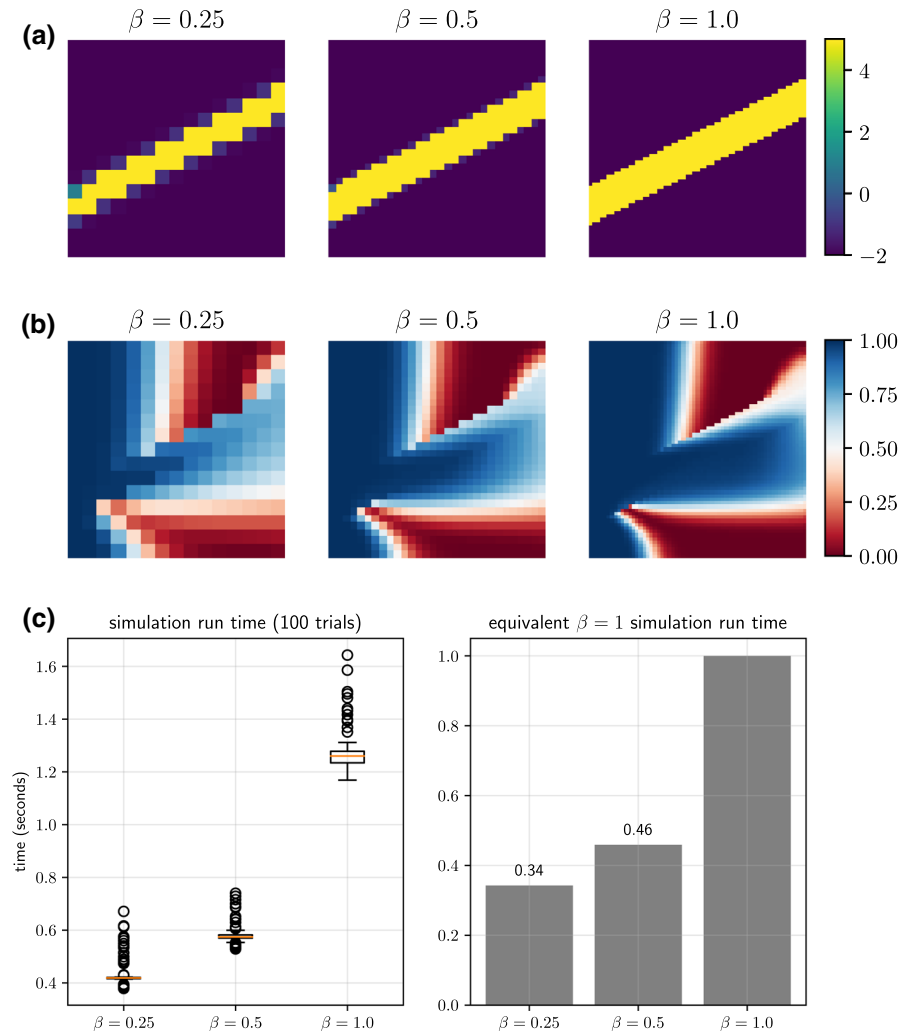


Fig. 6 Effect of grid fidelity factor β on the environment for test case 1: **a** on a sample of log-permeability, **b** on corresponding saturation and **c** on simulation run time

Figure 6 provides visualization for the effect of fidelity factor β , on the simulation in test case 1.

Figure 6a and b show log-permeability and saturation plots corresponding to $\beta = 0.25$, $\beta = 0.5$ and $\beta = 1.0$. Furthermore, Fig. 6c illustrates the effect of grid fidelity on simulation run time for a single episode (shown on left with a box plot of 100 simulations) and the equivalent $\beta = 1$ simulation run time for each grid fidelity factor (shown on right). Equivalent $\beta = 1$ simulation run time is defined as the ratio of average simulation run time for a grid fidelity factor β , to that corresponding to $\beta = 1$. This quantity is used as a scaling factor to convert the number of simulations

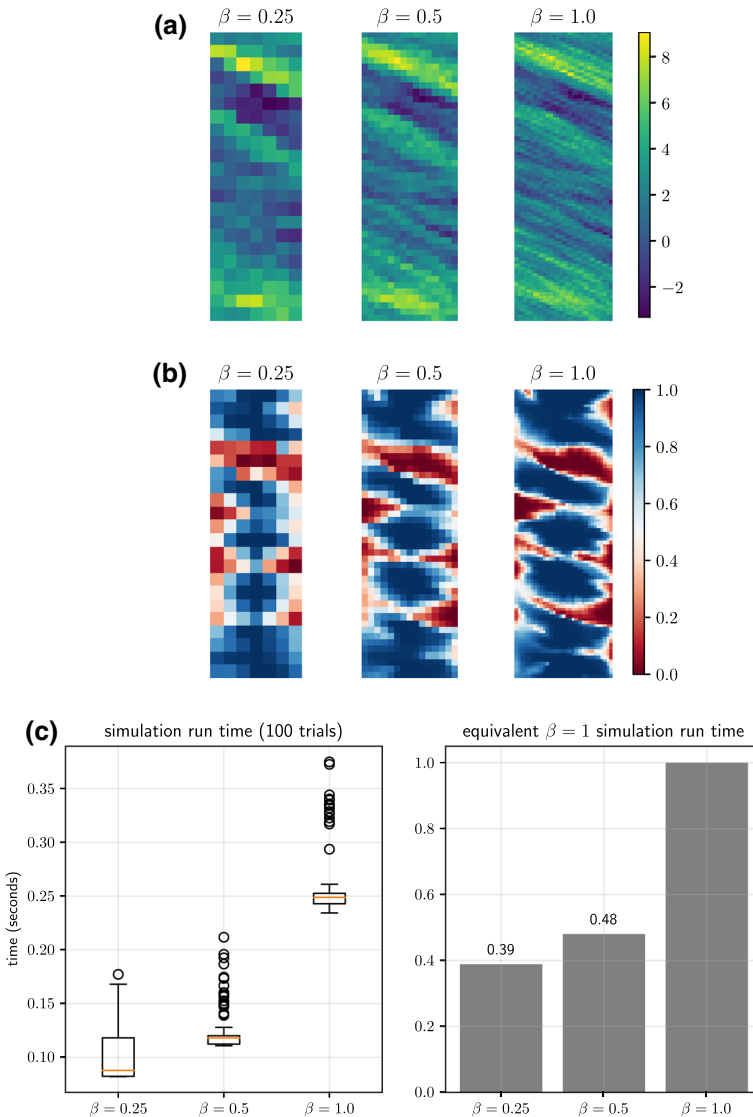


Fig. 7 Effect of grid fidelity factor β on the environment for test case 2: **a** on a sample of log-permeability, **b** on corresponding saturation and **c** on simulation run time

for any value of β to its equivalent number of simulations as if they were performed with $\beta = 1$. Similar plots for test case 2 are shown in Fig. 7.

The results obtained using the proposed framework are evaluated against the benchmark optimization results. These benchmark optimal results are obtained using the differential evolution (DE) algorithm (Storn and Price 1997). For both optimization methods (PPO and DE), multi-processing is used to reduce total computational time. However, the parallelism behavior is quite varied between the PPO and DE algorithms.

For instance, neural networks are back propagated synchronously at the end of each policy iteration of PPO, which causes extra computational time in waiting and data distribution. For this reason, a computational cost comparison among various experiments is performed by comparing the number of simulation runs in each experiment. The PPO algorithm for the proposed framework is executed using the stable baselines library (Raffin et al. 2019), while python's SciPy (Virtanen et al. 2020) library is used as provided in Appendix 3 delineates all the algorithm parameters used in this study.

4 Results

The control policy in which the injector and producer wells are equally open throughout the entire episode is called the base policy. Under such policy, the water flooding prominently takes place in the high permeability region, leaving the low permeability region swept inefficiently. The optimal policy for these test cases would be to control the producer and injector flow to mitigate this imbalance in water flooding. The optimal policy, learned using reinforcement learning for test case 1, shows on average around 12% improvement with respect to the recovery factor achieved using the base policy. While for test case 2, the average improvement is in the order of 25%.

Figure 8 illustrates the plots for the policy return $R^{\pi(A|S)}$, corresponding to all the frameworks listed in Table 4 for test case 1. At the beginning of the learning process, the policy return values for single-grid framework keeps improving and eventually converge to a maximum value when the policy converges to an optimal policy. Note that for lower value of grid fidelity factor β , the optimal policy return is also low. In other words, the coarsening of simulation grid discretization also reflects in overall reduction in recovery factor. This is due to the low accuracy of the state and action representation for environments with $\beta < 1$. However, the overall computational gain is observed as a result of coarser grid sizes. The simulation run time corresponding to $\beta = 0.25$ and $\beta = 0.5$ shows a reduction of around 66% and 54% compared to $\beta = 1$. The results of multigrid frameworks are compared with the single grid framework corresponding to $\beta = 1$ which refers to the classical PPO algorithm that uses the environment with a fixed high-fidelity grid factor. As shown in the plots at the center and right of Fig. 8, both multigrid frameworks show convergence to the optimal policy, which is achieved using the high-fidelity single-grid framework. In the fixed multigrid framework, the fidelity factor is incremented at a fixed interval of 25,000 number of episodes. The adaptive framework is also provided with the same interval but with additional convergence check within each interval. For multigrid learning plots shown in Fig. 8 (center and right plots), the equivalent number of episodes corresponding to the environment with $\beta = 1$ is illustrated as a secondary horizontal axis.

In this way, the computational effect of multigrid frameworks is directly compared to a single-grid framework (with $\beta = 1$). The equivalent number of $\beta = 1$ episodes corresponding to episodes with a certain value β is computed by multiplying it with the equivalent $\beta = 1$ simulation run time. For example, the number of episodes with $\beta = 0.25$ is multiplied by 0.37. For a fixed multigrid framework, it takes 45,496 equivalent $\beta = 1$ episodes to achieve an equally optimal policy that is obtained with a 75,000 number of episodes using single grid ($\beta = 1$) framework. Similarly,

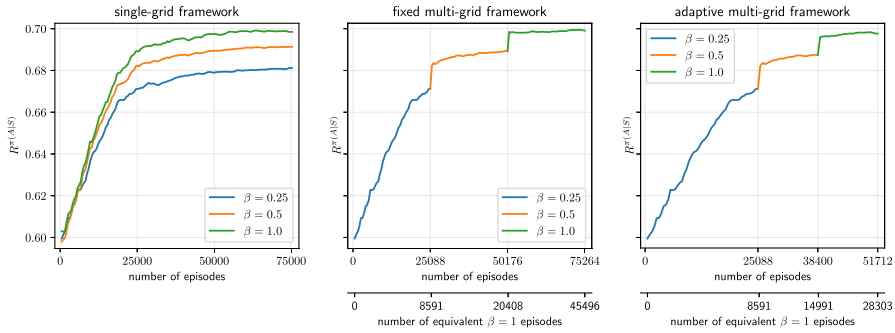


Fig. 8 Plots of policy return versus number of episodes for test case 1

the same is achieved with just 28,303 equivalent $\beta = 1$ episodes using the adaptive multigrid framework. In other words, around 38% and 61% reductions are observed in the simulation run time using fixed and adaptive multigrid frameworks, respectively. Further, the robustness of the policy learned using these frameworks is compared by applying it on the highest fidelity environment with random permeability samples from the distribution \mathcal{G}_1 , which were never seen during the policy learning process. Figure 9a shows the plots of these unseen permeability fields, while the corresponding results obtained using these frameworks are plotted in Fig. 9b.

Optimal results obtained using differential evolutionary (DE) algorithms are provided as benchmark (marked as DE in Fig. 9b). Note that DE algorithm is not a suitable method to solve the robust optimal control problem since it can provide optimal controls only for certain permeability samples as opposed to PPO algorithm where the learned policy is applicable to all samples of permeability distribution. However, DE results are used as the reference optimal results, which are achieved by direct optimization on sample-by-sample basis. Equivalence in the optimality of learned policies obtained using these three experiments can be observed from the closeness in their corresponding optimal recovery factors. Figure 10 demonstrates the visualization of the policy for an example of the permeability sample in case 1.

In this figure, the results are shown for permeability sample index 4 from the Fig. 9a where a high permeability channel passes through the lower region of the domain. The optimal policy in this case would be to restrict flow through the injector and producer wells that are in the vicinity of the channel. The superpositioned comparison of optimal results for base case, differential evolution, single-grid framework (where $\beta = 1$), fixed multigrid framework and adaptive multigrid framework shows that the optimal policy is learned successfully using the proposed framework.

For test case 2, similar results are observed as shown in Fig. 11.

The single-grid algorithms converge to an optimal policy in total 150,000 number of episodes. The fixed multigrid algorithm is trained with 50,000 episode interval for each grid fidelity factor as shown in the central plot in Fig. 11. The optimal policy is learned in 94,141 equivalent $\beta = 1$ episodes thus saving around 38% of simulation run time. The adaptive multigrid framework further reduces computational cost by achieving the optimal policy in 39,582 equivalent $\beta = 1$ episodes (simulation time

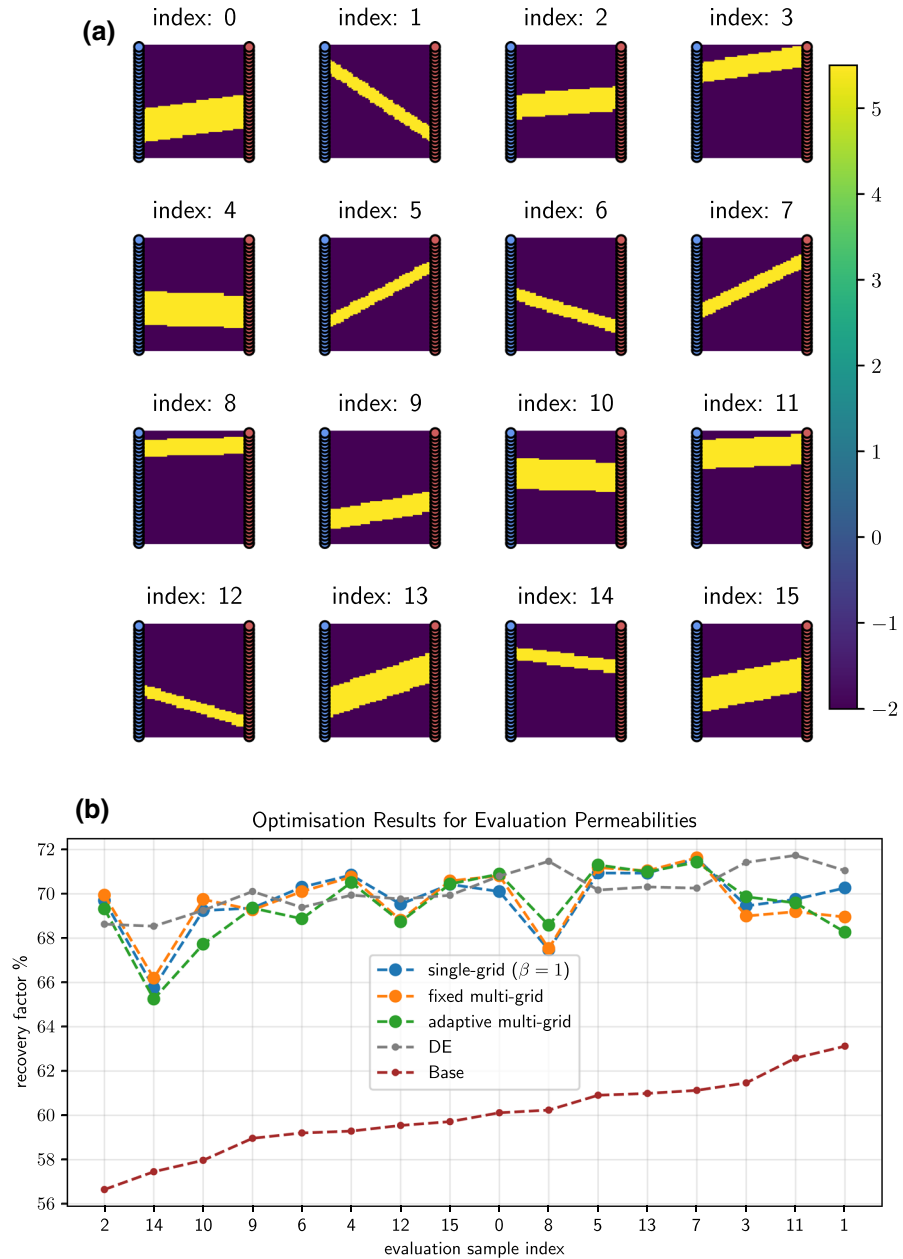


Fig. 9 Evaluation of learned policies for test case 1: **a** evaluation samples of log-permeability distribution G_1 , **b** recovery factor (in % format) versus evaluation sample index (from **a**) plot

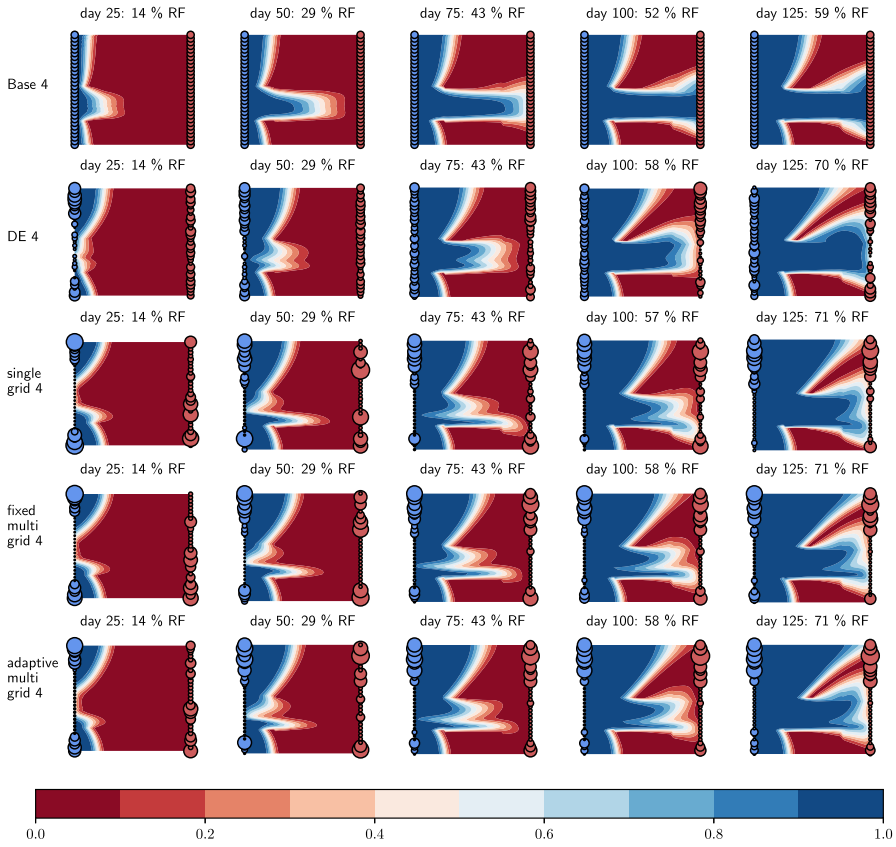


Fig. 10 Illustration of learned optimal control policies for test case 1

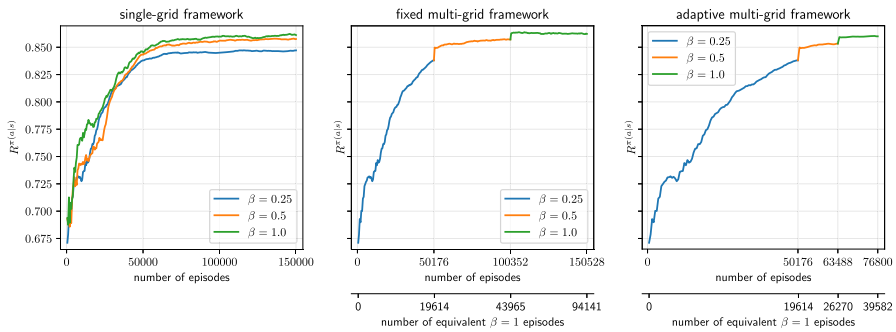


Fig. 11 Plots of policy return versus number of episodes for test case 2

reduction of about 76% with respect to the $\beta = 1$ single-grid framework). Figure 12 illustrates the results of policy evaluation on unseen permeability samples from the distribution \mathcal{G}_2 .

The permeability samples are shown in Fig. 12a and the optimal recovery factor corresponding to the learned policies is plotted in Fig. 12b. Figure 13 shows the optimal controls for an example of the permeability sample index 5 from Fig. 12a.

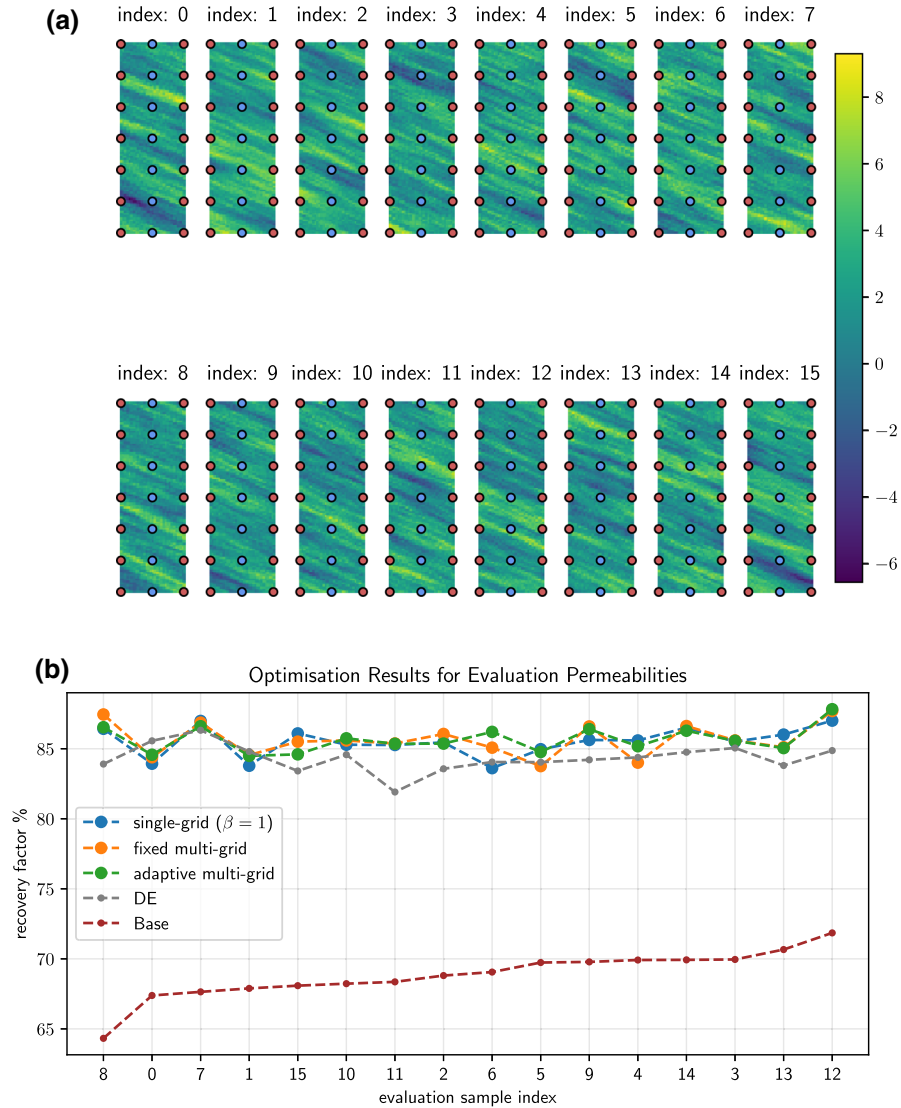


Fig. 12 Evaluation of learned policies for test case 2: **a** evaluation samples of log-permeability distribution G_2 , **b** recovery factor (in % format) versus evaluation sample index (from **a**) plot

The optimal policy learned using differential evolution algorithm refers to increasing the flow through injector wells which are in the low permeability region while restricting the flow through producer wells for which the water cut-off is reached. Policies learned using the RL framework take advantage of the default location and orientation of high-permeability regions. In this case, the optimal policy is achieved by controlling the well flow control such that the flow traverses through the perme-

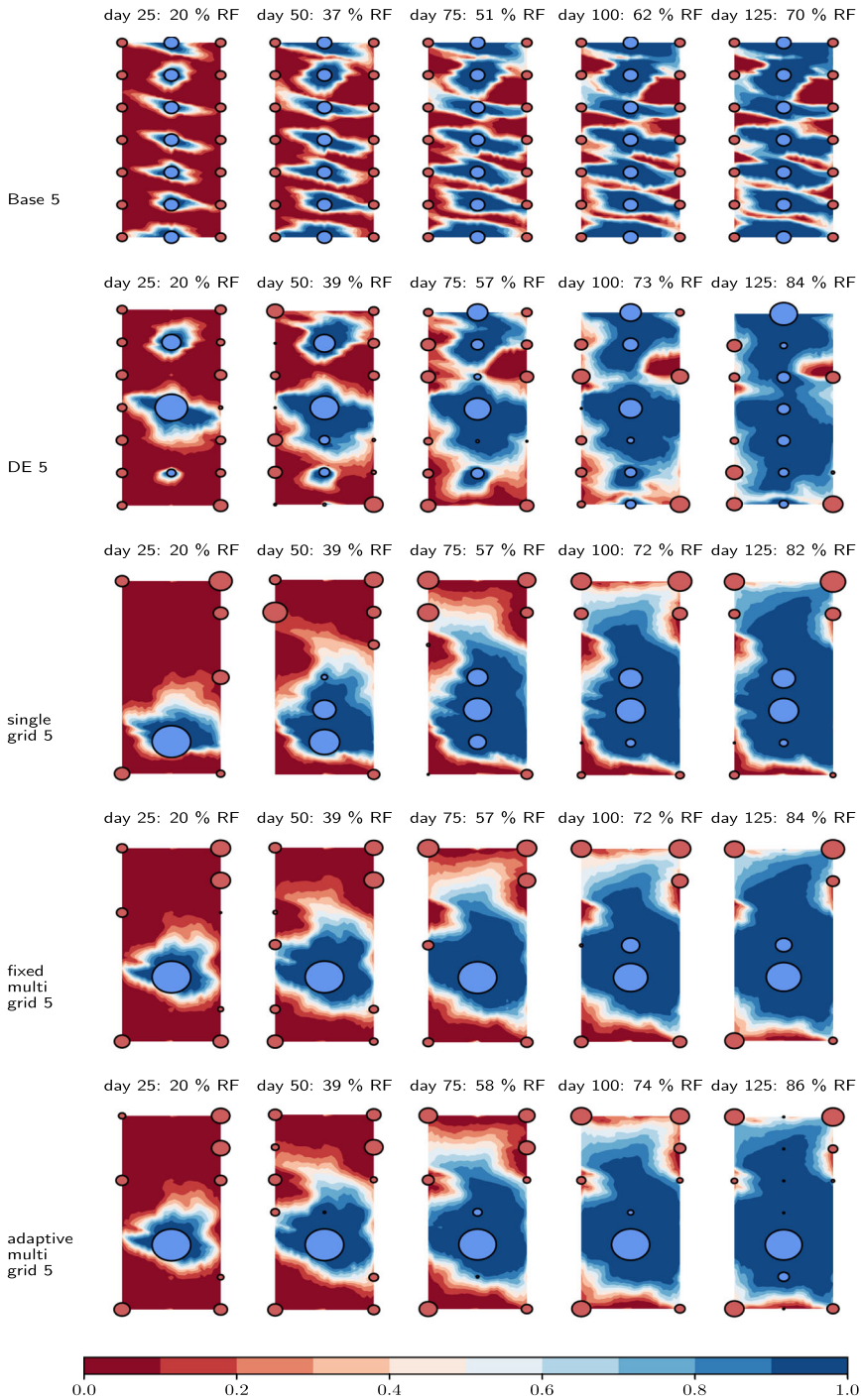


Fig. 13 Illustration of learned optimal control policies for test case 2

ability channels (that is, the flow is more or less perpendicular to the permeability orientation).

5 Conclusion

An adaptive multigrid RL framework is introduced to solve a robust optimal well control problem. The proposed framework is designed to be general enough to be applicable to similar optimal control problems governed by a set of time dependant nonlinear PDEs. Numerically, a significant reduction in the computational costs of policy learning is observed compared to the results of the classical PPO algorithm. In the presented case studies, 61% computational savings in simulation runtime for test case 1 and 76% for test case 2 is observed. However, note that these results are highly dependent on the right choice of the algorithm hyperparameters (e.g., δ , n , β and \mathbf{E}) which were tuned heuristically. As a future direction for this research study, the aim is to find the optimal values for β that maximize the overall computational savings. Furthermore, the policy transfer was performed sequentially in the current framework, which seemed to have worked optimally. However, to improve the generality of the proposed framework, it would be important to study the effect of the sequence of policy transfers on overall performance.

Acknowledgements The first author would like to acknowledge the Ali Danesh scholarship to fund his Ph.D. studies at Heriot-Watt University. The authors also acknowledge EPSRC funding through the EP/V048899/1 grant.

Declaration

Conflict of interest The authors declare that they have no known conflicting financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Appendix 1: Cluster Analysis of Permeability Uncertainty Distribution

Training vector \mathbf{k} is chosen to represent the variability in the permeability distribution \mathcal{K} . For the optimal control problem, our main interest is in the uncertainty in the dynamical response of permeability rather than the uncertainty in permeability itself. As a result, the connectivity distance (Park 2011) is used as a measure of the distance between the permeability field samples. The connectivity distance matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$

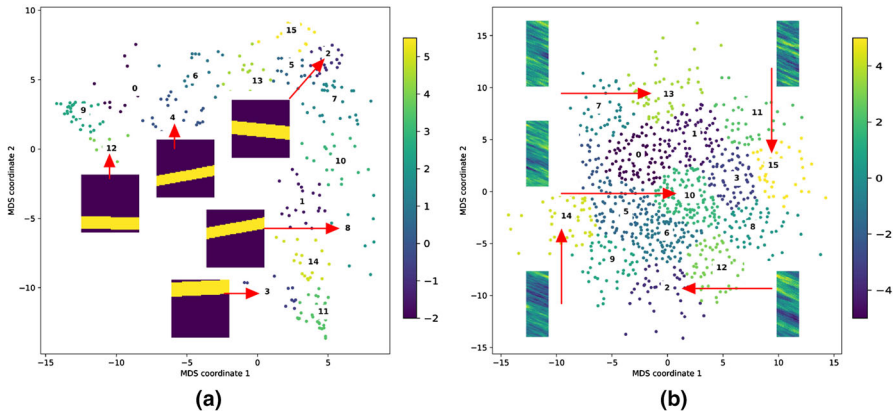


Fig. 14 Log-permeability plots for training data of test case 1 and 2: **a** and **b** illustrate clustering for \mathcal{G}_1 and \mathcal{G}_2 distribution samples

among the N samples of \mathcal{K} is formulated as

$$\mathbf{D}(k_i, k_j) = \sum_{x''} \int_{t_0}^T [s(x'', t; k_i) - s(x'', t; k_j)]^2 dt,$$

where N corresponds to a large number of samples of uncertainty distribution, $s(x'', t; k_i)$ is saturation at location x'' , and time t , when the permeability is set to k_i and all wells are open equally. Multidimensional scaling of the distance matrix \mathbf{D} is used to produce N two-dimensional coordinates d_1, d_2, \dots, d_N , each representing a permeability sample. The coordinates d_1, d_2, \dots, d_N are obtained such that the distance between d_i and d_j is equivalent to $\mathbf{D}(k_i, k_j)$. In the k-means clustering process, these coordinates are divided into l sets S_1, S_2, \dots, S_l , obtained by solving the optimization problem, defined as

$$\arg \min_S \sum_i^l \sum_{d_j \in S_i} \|d_j - \mu_{S_i}\|,$$

where μ_{S_i} is average of all coordinates in the set S_i . The training vector \mathbf{k} is a set of l samples of \mathcal{K} where each of its value k_i correspond to the one nearest to μ_{S_i} .

The total number of samples N and clusters l are chosen to be 1000 and 16 for both uncertainty distributions, \mathcal{G}_1 and \mathcal{G}_2 . A training vector \mathbf{k} is obtained with samples k_1, \dots, k_{16} each corresponding to a cluster center. Figure 14a and b show cluster plots for samples of permeability distribution \mathcal{G}_1 and \mathcal{G}_2 , respectively. Furthermore, 16 permeability samples, each randomly chosen from a cluster, are chosen to evaluate the learned policies. Figures 9a and 12a illustrate these samples for test case 1 and 2, respectively.

Appendix 2: Definitions of Value and Advantage Function

In RL, the policy $\pi(A|S)$ is said to be optimal if it maps the state S_t with an action A_t that correspond to maximum expected return value. These return values are learned through the data obtained from agent-environment interactions. The following are some definitions of return values typically used in RL:

The value function is the expected future return for a particular state S_t and is defined as

$$V(S) = \mathbb{E}_{\pi} \left[\sum_m \gamma^m R_{m+t+1} | S_t = S \right],$$

where $\mathbb{E}_{\pi}[\dots]$ denotes the expected value given that the agent follows the policy π . As a shorthand notation, $V(S)$ in the state S_t is denoted as V_t .

Q function is similar to value function except that it represents the expected return when the agent takes action a_t in the state S_t . It is defined as

$$Q(S, A) = \mathbb{E}_{\pi} \left[\sum_m \gamma^m R_{m+t+1} | S_t = S, A_t = A \right].$$

Advantage function is defined as the difference between Q function and value function and is denoted by $Adv(S, A)$ at state S and action A .

Appendix 3: Algorithm Parameters

Parameters used for PPO are tabulated in Table 5 which were tuned using trial and error. For the PPO algorithm, the parameters were tuned to find the least variability in the learning plots. Figures 15 and 16 show learning plots corresponding to three different seeds to show the stochasticity of the obtained results. The parameters of the

Table 5 PPO algorithm parameters

	Case 1	Case 2
Number of CPUs, N	64	64
Number of steps, T	40	40
Mini-batch size, M	16	16
Epochs, K	20	20
Discount rate, γ	0.99	0.99
Clip range, ϵ	0.1	0.15
Policy network MLP layers	[93,150,100,80,62]	[35,70,70,50,21]
Policy network activation functions	tanh	tanh
Policy network optimizers	Adam	Adam
Learning rate	3e-6	1e-4

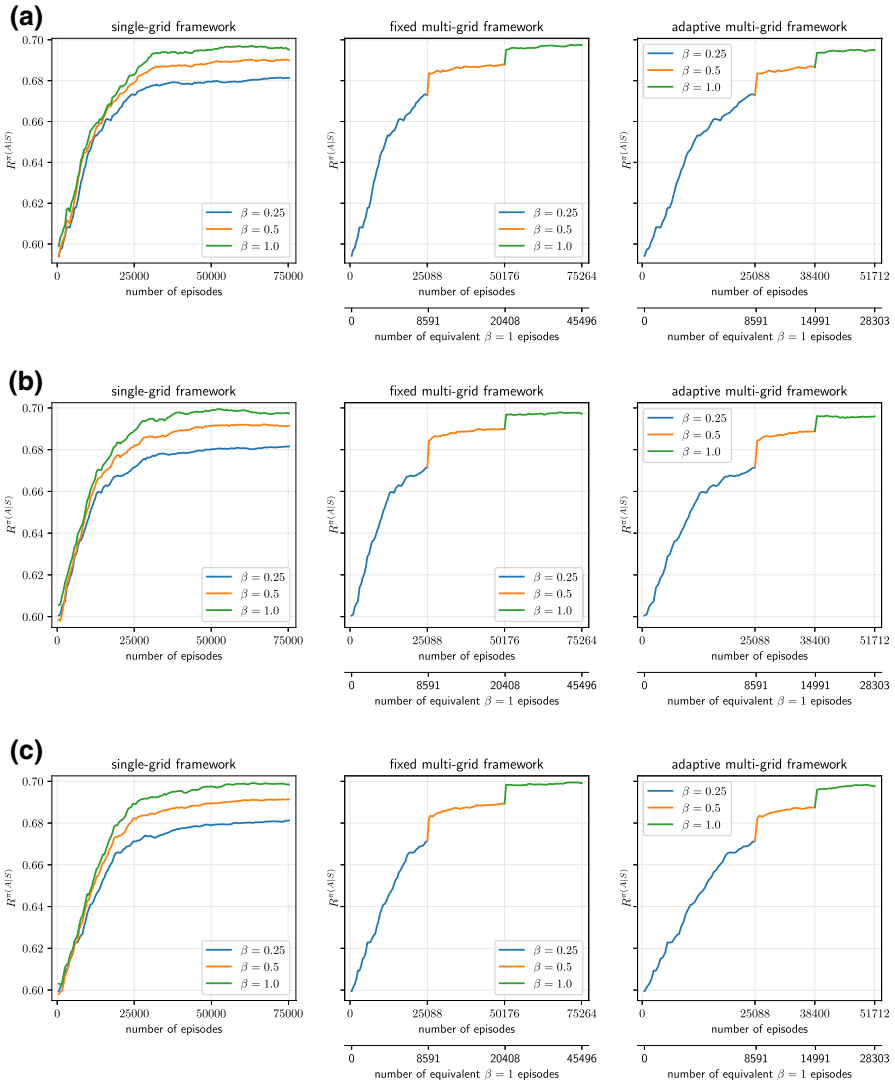


Fig. 15 Learning plots with seed 1 (a), 2 (b) and 3 (c) for test case 1

DE algorithm are delineated in Table 6. The code repository for both the test cases presented in this paper can be found on the link: https://github.com/atishdixit16/ada_multigrid_ppo.

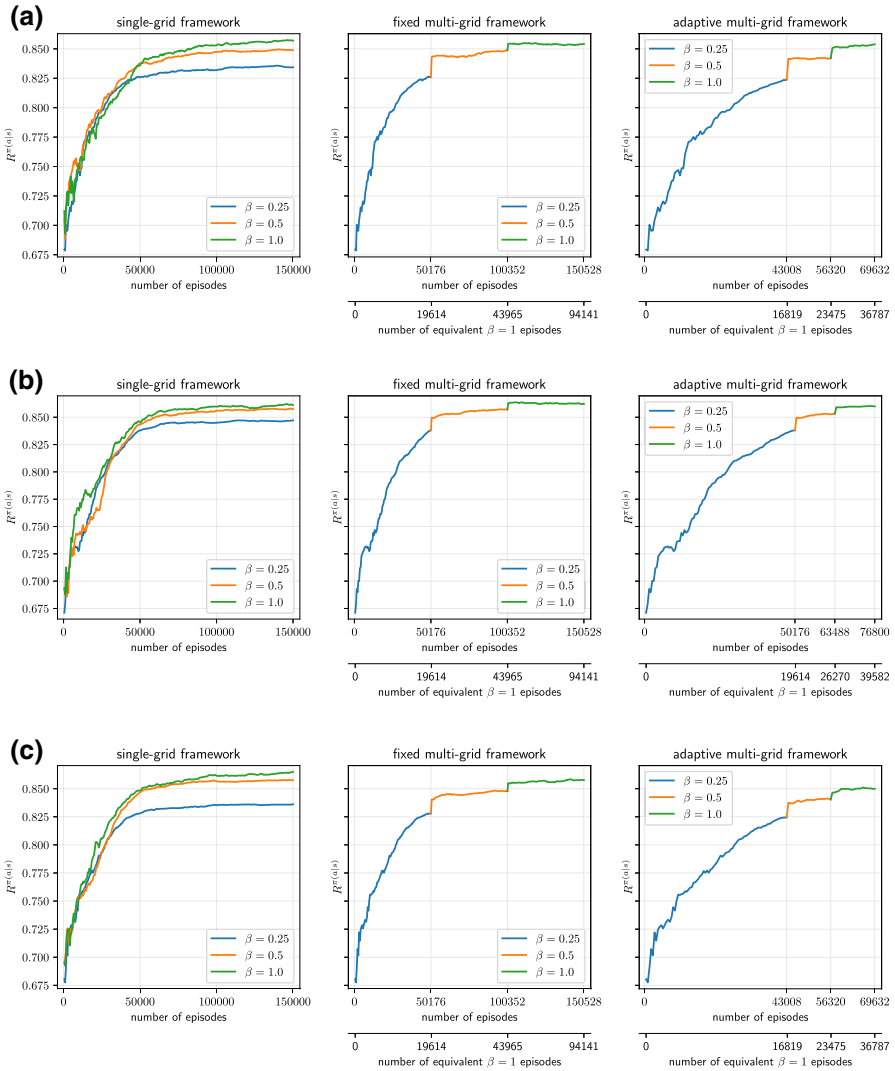


Fig. 16 Learning plots with seed 1 (a), 2 (b) and 3 (c) for test case 2

Table 6 DE algorithm parameters

	Case 1	Case 2
Number of CPUs	64	64
Number of iterations	1024	1024
Population size	310	105
Recombination factor	0.9	0.9
Mutation factor	(0.5,1)	(0.5,1)

References

- Anderlini E, Forehand DI, Stansell P, Xiao Q, Abusara M (2016) Control of a point absorber using reinforcement learning. *IEEE Trans Sustain Energy* 7(4):1681–1690
- Anderson C, Crawford-Hines S (1994) Multigrid q-learning. In Technical Report CS-94-121, Citeseer
- Brouwer D, Jansen J, Van der Starre S, Van Kruijsdijk C, Berentsen C, et al. (2001) Recovery increase through water flooding with smart well technology. In: SPE European formation damage conference, society of petroleum engineers
- Christie MA, Blunt M et al (2001) Tenth SPE comparative solution project: A comparison of upscaling techniques. Society of Petroleum Engineers. In: SPE reservoir simulation symposium
- Dixit A, ElSheikh AH (2022) Stochastic optimal well control in subsurface reservoirs using reinforcement learning. *Eng Appl Artif Intell* 114:105106
- Dornheim J, Link N, Gumbsch P (2020) Model-free adaptive optimal control of episodic fixed-horizon manufacturing processes using reinforcement learning. *Int J Control Autom Syst* 18(6):1593–1604
- Fachantidis A, Partalas I, Tsoumakas G, Vlahavas I (2013) Transferring task models in reinforcement learning agents. *Neurocomputing* 107:23–32
- Fernández F, García J, Veloso M (2010) Probabilistic policy reuse for inter-task transfer learning. *Robot Auton Syst* 58(7):866–871
- Lazaric A, Restelli M, Bonarini A (2008) Transfer of samples in batch reinforcement learning. In Proceedings of the 25th international conference on Machine learning, 544–551
- Li B, Xia L (2015) A multi-grid reinforcement learning method for energy conservation and comfort of HVAC in buildings. In 2015 IEEE international conference on automation science and engineering (CASE), IEEE, 444–449
- Miftakhov R, Al-Qasim A, Efremov I (2020) Deep reinforcement learning: reservoir optimization from pixels. In: International petroleum technology conference, OnePetro
- Müller S, Schüler L (2019) Geostat-framework/gstools: Bouncy blue
- Narvekar S, Sinapov J, Leonetti M, Stone P (2016) Source task creation for curriculum learning. In: Proceedings of the 2016 international conference on autonomous agents & multiagent systems, 566–574
- Nasir Y, He J, Hu C, Tanaka S, Wang K, Wen X (2021) Deep reinforcement learning for constrained field development optimization in subsurface two-phase flow. *arXiv preprint arXiv:2104.00527*
- Paregis S (1996) Multi-grid methods for reinforcement learning in controlled diffusion processes. In: NIPS, Citeseer, pp 1033–1039
- Park K (2011) Modeling uncertainty in metric space. Stanford University
- Rabault J, Kuchta M, Jensen A, Réglade U, Cerardi N (2019) Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *J Fluid Mech* 865:281–302
- Raffin A, Hill A, Ernestus M, Gleave A, Kanervisto A, Dormann N (2019) Stable baselines3. <https://github.com/DLR-RM/stable-baselines3>
- Roseta-Palma C, Xepapadeas A (2004) Robust control in water management. *J Risk Uncertain* 29(1):21–34
- Schulman J, Moritz P, Levine S, Jordan M, Abbeel P (2015) High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*
- Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*
- Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J Glob Optim* 11(4):341–359
- Taylor ME, Stone P (2005) Behavior transfer for value-function-based reinforcement learning. In: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, 53–59
- Taylor ME, Stone P (2009) Transfer learning for reinforcement learning domains: a survey. *J Mach Learn Res* 10(7)
- van Essen G, Zandvliet M, Van den Hof P, Bosgra O, Jansen JD et al (2009) Robust waterflooding optimization of multiple geological scenarios. *SPE J* 14(01):202–210
- Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat İ, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P, SciPy 10 Contributors, (2020) SciPy 1.0: fundamental algorithms for scientific computing in python. *Nat Methods* 17:261–272

- Whitaker S (1999) Single-phase flow in homogeneous porous media: Darcy's law. In: The method of volume averaging, Springer, Berlin, 161–180
- Ziv O, Shimkin N (2005) Multigrid methods for policy evaluation and reinforcement learning. In: Proceedings of the 2005 IEEE international symposium on, mediterranean conference on control and automation intelligent control, 2005., IEEE, 1391–1396