Computational Mechanics (2023) 72:95–124
https://doi.org/10.1007/s00466-023-02335-6

ORIGINAL PAPER

Design of experiments for the calibration of history-dependent models via deep reinforcement learning and an enhanced Kalman filter

Ruben Villarreal[1] · Nikolaos N. Vlassis[2] · Nhon N. Phan[2] · Tommie A. Catanach[1] · Reese E. Jones[1] · Nathaniel A. Trask[3] · Sharlotte L. B. Kramer[3] · WaiChing Sun[2]

Received: 30 September 2022 / Accepted: 30 March 2023 / Published online: 12 May 2023

Abstract
Experimental data are often costly to obtain, which makes it difficult to calibrate complex models. For many models an experimental design that produces the best calibration given a limited experimental budget is not obvious. This paper introduces a deep reinforcement learning (RL) algorithm for design of experiments that maximizes the information gain measured by Kullback–Leibler divergence obtained via the Kalman filter (KF). This combination enables experimental design for rapid online experiments where manual trial-and-error is not feasible in the high-dimensional parametric design space. We formulate possible configurations of experiments as a decision tree and a Markov decision process, where a finite choice of actions is available at each incremental step. Once an action is taken, a variety of measurements are used to update the state of the experiment. This new data leads to a Bayesian update of the parameters by the KF, which is used to enhance the state representation. In contrast to the Nash–Sutcliffe efficiency index, which requires additional sampling to test hypotheses for forward predictions, the KF can lower the cost of experiments by directly estimating the values of new data acquired through additional actions. In this work our applications focus on mechanical testing of materials. Numerical experiments with complex, history-dependent models are used to verify the implementation and benchmark the performance of the RL-designed experiments.

Keywords Experimental design · Deep reinforcement learning · Enhanced Kalman filter · Elastoplasticity

1 Introduction

Finding an accurate representation for the physical response of a material with complex nonlinear behavior is a difficult endeavor that depends on the parametric complexity of the selected model, the experimental data available for calibration, and other factors such as indirect, noisy, or incomplete observations. This has motivated experimental design as a longstanding research area [15] with a multitude of approaches In particular, the field of Bayesian optimal experimental design [5, 55] provides a paradigm to incorporate both prior information and uncertainties. It is guided by a user-selected utility function which can be recast as a dynamic programming problem based on the well-known Hamilton–Jacobi–Bellman equation [24, 25]. Reinforcement learning solutions [2, 10, 36, 38, 51, 52], such as the one proposed in this work, have the same basis. Also related to the present endeavour is real-time data assimilation, such as the Dynamic Data Driven Application Systems framework [7].

The physical characteristics of the material of interest in large part guide optimal experimental design. Material symmetry plays a role in model and response complexity, and hence in the complexity of the experiments needed to characterize it. For instance, the irreducible number of parameters for isotropic linear elasticity is just two, so two linearly independent observations of stress are sufficient to characterize the model; however, the number of parameters increases for lower symmetry materials such as transversely isotropic, orthotropic, monoclinic and fully anisotropic elasticity (which has 21 independent parameters). This complexity requires more independent observations and poses a more challenging optimal experimental design problem. Nonlinearity and path/history dependence of the

✉ Nikolaos N. Vlassis
nnv2102@columbia.edu

[1] Sandia National Laboratories, Livermore, CA, USA

[2] Department of Civil Engineering and Engineering Mechanics, Columbia University, New York, USA

[3] Sandia National Laboratories, Albuquerque, NM, USA

material response also play a role in model complexity and the data needed to obtain a sufficiently accurate calibration. Given large enough loads, most materials exhibit both nonlinearity in the stress–strain response and dissipation which leads to path dependence. For instance, the material model in Ames et al. [1] requires 37 material parameters and complex forms to capture the thermo-mechanical response of amorphous polymers, and Ma and Sun [45] requires 25 material parameters to capture the crystal plasticity and phase transition of salt under high pressure and high temperature. While the use of machine learning to generate constitutive laws [30, 64] may enable one to bypass the need to identify a particular model form, the data required to obtain a sufficiently accurate neural network or Gaussian process model may be considerable and complex/ambiguous, and hence costly to obtain [16, 22, 66, 67].

The complexity of both traditional and emerging models, combined with the epistemic uncertainty that commonly occurs with experimental calibration, makes it difficult to use intuition alone to foresee the optimal experimental design for a fixed amount of resources (such as the duration of the experiment). While Bayesian design of experiments (DOE) [5, 53] may efficiently estimate and optimize the information gain of an experiment with a limited number of design variables, calibrating a material model often requires hundreds or even thousands of loading steps, each with a set of available options/control actions. This decision-tree, beginning with the material in its reference state, advances to subsequent states through a series of policy predictions that facilitate long-term planning, thereby rendering traditional myopic Bayesian Design of Experiments (DOE) inapplicable. The sequential process of decision–action-feedback of an experiment can be represented as a Markov decision process (MDP) in discrete time, where the optimal design of the experiment can be recast as a policy that optimizes a pre-selected set of rewards. The MDP is the foundation of reinforcement learning and has been successfully used for real-time decision making in robotics, control, and other fields [20, 23, 37, 72]. The interaction loop in a reinforcement learning algorithm is shown schematically in Fig. 1 and centers around an *actor* utilizing a *policy* to maximize *rewards*.

In this paper, we use a model-based deep reinforcement learning approach where an agent is constantly building a model based on the interaction with the environment. In contrast, a model-free algorithm does not involve the modeling and parameterization of the state and policy. Instead, the policy of the action is carried out through the statistics of rewards for a given action in the decision tree via Q table through policy gradient (see Feinberg et al. [14] and Sutton and Barto [63]). Due to the depth of the decision tree required for the design-of-experiment problem, we can only generate walks that visit a small fraction of states. Hence, the model-based method is preferred, as the model can estimate the states and

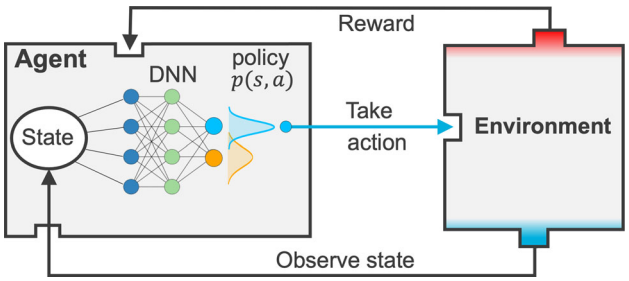

Fig. 1 Deep reinforcement learning. The *environment* consists of a physical experiment, which reacts to *actions*, such as a prescribed strain, to produce a new observed *state*. The reaction of the experiment, by way of the calibration method, produces a *reward* that drives the (external) *agent* to generate a *policy* that takes actions that maximizes rewards. In deep RL the action-value policy is represented with a deep neural network (DNN) that takes states and current rewards as inputs and produces favorable actions and values (which are accumulated rewards) as outputs

policies and hence enables planing with fewer interactions. The same approach has been used, for instance, in Chess and Go, where the total number of visited states is significantly less than the possible states (cf. Silver et al. [59]). Please refer to Sutton and Barto [63] for a comprehensive review of various types of reinforcement learning.

In this work, we introduce a reinforcement learning (RL) [33, 43, 63] approach to optimal experimental design that utilizes a deep neural network and an enhanced Kalman filter (KF) [41, 69] for policy estimation to maximize the information gain for an experiment. We develop a framework which exploits our prior knowledge of the underlying physics by utilizing the structure of common history dependent models, such as plasticity, and augments the RL state with the uncertainty associated with model parameters. This allows the agent to select actions which are guided by parameter sensitivities to reduce the estimated variance of the parameters. The enhanced Kalman filter we present will provide a computationally efficient means of estimating parameter uncertainty for nonlinear systems. This approach offers several salient features and improvements over previous efforts (cf. Wang and Sun [65]) and is focused on providing optimal experimental designs for complex models. First of all, the use of a KF to provide a parameter calibration and parametric uncertainty-based reward via the Kullback–Leibler (KL) divergence measuring information gain enables the method to bypass the costly bottleneck induced by the Nash–Sutcliffe efficiency (NSE) index [46] used to estimate the values of new experimental data. This bypass leads to significant cost savings for experimental data that are expensive to obtain. Second, the introduction of a deep neural network trained by Monte Carlo tree search (MCTS) balances the needs for exploration and exploitation. Third, the RL and KF are enhanced with additional state information and strategies to handle the history dependence of the process and model. In

essence, we seek to replace a traditional experimentalist who pre-conceives experiments with an RL actor who is guided by a pre-trained policy and reacts to real-time rewards to minimize uncertainty in the model calibration.

The remarkable adaptability of Kalman filters to a vast array of linear and non-linear problems stems from their ability to efficiently estimate uncertainty through recursive updates in state and measurement parameters. For instance, Kalman filters have been effectively employed for real-time tracking of structural damage in civil infrastructure. Their diverse applications span from detecting building damage resulting from extreme events like earthquakes or severe weather [17, 70, 73], monitor degradation using incomplete knowledge of dynamics [27, 42], detection of anomalous behaviour in structural sub-elements [49], non-collocated heterogeneous sensing in non-linear systems [6], and accounting for the effect of environmental changes on the response of the system [12, 26, 29].

In Sect. 2 we develop a Kalman filter that can handle the non-linearity, non-smoothness, and history-dependence of common material models. It also exploits known behavior of the model, for example specific parameter sensitivities dominate in certain regimes. Since the Kalman filter is essentially an incremental Bayesian calibration (and state estimation) method, it provides parameter covariances as well as estimates of the mean parameters. The deep reinforcement learning method, described Sect. 3, utilizes this information in both the state and the reward that defines and guides the policy. The policy for controlling an experiment with the chosen reward creates data along a path that maximizes information gain in the selected model's parameters. We employ a policy-value scheme that represents the rewards for control actions over the decision tree with a neural network. Since the possible paths of even simple experiments with a few allowable actions/decisions at every state have an enormous number of possible paths, we need to use Monte Carlo sampling on this tree to train the policy-value network. In Sect. 4, the proposed algorithm is demonstrated with widely-used plasticity models [44]. Two numerical examples are specifically designed to validate the deep RL where a benchmark of optimal experimental design is available. A third example demonstrates that the algorithm is effective in obtaining an optimal experimental design given a pre-selected model where the best design is not known *a priori*. Table 1 summarizes the notation used in the following sections. More details of the particular models and the KF applied to history dependent models are given in the Appendices.

Table 1 Notation for generic model, exemplar, Kalman filter and reinforcement learning method

Symbol	Description
d	Observable output (data)
$\mathbf{y}$	Model output
$\mathbf{x}$	Controllable input
$\mathbf{z}$	Hidden model state
$\mathbf{m}$	Observation/response model
$\mathbf{f}$	Hidden state dynamics
$\boldsymbol{\theta}$	Parameters
$\boldsymbol{\sigma}$	Stress (model output)
ϵ	Strain (observable input)
ϵ^p	Plastic strain (hidden model state)
$\boldsymbol{\mu}$	Mean estimate of parameters
$\boldsymbol{\Sigma}$	Covariance estimate of parameters
A	Parameter sensitivity of $\mathbf{m}$
F	Hidden state transition
R	Observation noise covariance
K	Kalman gain matrix
$\mathcal{S}$	Experiment states $\boldsymbol{s}$
$\mathcal{A}$	Experiment actions $\boldsymbol{a}$
$\mathcal{R}$	Reward
v	Value
$\boldsymbol{p}$	Policy

2 Model calibration

The calibration task is, given a model $\mathbf{m}$:

$$\mathbf{y} = \mathbf{m}(\mathbf{x}; \boldsymbol{\theta}) \tag{1}$$

with parameters $\boldsymbol{\theta}$, find a path $\{\mathbf{x}_k\}$ that leads to the highest accuracy and lowest uncertainty in the calibrated parameters. Here $\mathbf{x}_k = \mathbf{x}(t_k)$ is the input at discrete time t_k and the sequence $\{\mathbf{x}_k, k = 1, n\}$ represents an experimental protocol. The total number of steps n indicates the cost of the experiment. The controls $\mathbf{x}$ evoke an observable response $\mathbf{y}$ from the physical system that $\mathbf{m}$ models. At each step, a finite number of actions are available to the machine controlling the experiment; choosing these actions is the subject of Sect. 3.

Our focus is on models where the current response $\mathbf{y}_k$ depends on the current and previous inputs $\mathbf{x}_i,\ i \leq k$. With a history-dependent model

$$\mathbf{y} = \mathbf{m}(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}), \tag{2}$$

latent variables $\mathbf{z}$ are introduced that have their own evolution

$$\dot{\mathbf{z}} = \mathbf{f}(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}) \tag{3}$$

and are typically hidden from observation. In the realm of material physics the latent variables typically describe internal states that are linked to dissipation, such as plastic strain. This irreversible behavior greatly complicates experimental design, since trial loadings can lead to permanent changes in the material. Details of the elasto-plastic model we use as an exemplar are given in "Appendix A".

2.1 Kalman filter for calibration

Systems that rely on not only their present state but also on past states, inputs, and controls are referred to as history-dependent systems. Capturing this history dependence requires the history of the system to be sufficiently represented by the state. While directly concatenating the state history may seem feasible, doing so may lead to a very high-dimensional state space. This high dimensionality can be problematic for the training of the policy neural network, which must then take the high-dimensional state-action pair as input in order to output the policy for each action. Therefore, our primary objective is to optimize exploration by offering intermediate feedback to the agent in the form of rewards derived from Kalman Filter (KF) estimates. These intermediate rewards play a vital role in action selection, as shown in (25), and supply the necessary information to evaluate environments that deviate from pretraining conditions. To circumvent the need for batch processing the complete history-dependent evolution of data generated during training, we employ a sequential Bayesian filtering scheme. Moreover, providing sequential updates of the reward enables immediate feedback on the environment and affords flexibility in devising an online policy capable of distinguishing environments that diverge from pretraining circumstances.

For simplicity we will assume we have an experiment where we can control all components of strain and observe the stress of the material sample we would like to model; hence the calibration data consists of a sequence of strain–stress input–output pairs. There are many applicable methods to obtain parameter estimates given calibration data [62], such as nonlinear least squares regression. Here we use an online method, the Kalman filter (KF), that provides concurrent uncertainty quantification. The KF estimates a parameter covariance as well as a parameter mean since it is essentially an incremental Bayesian update of the parameters. It also provides a probabilistic estimate of the response. The current context presents a few complications that require an enhanced KF: (a) the model is nonlinear with respect to the inputs and parameters, (b) it is not smooth with respect to the parameters and (c) it is history-dependent.

Remark In our design of experiments, the KF is primarily tasked with guiding the experiments; after the data is collected, the model could be re-calibrated using other methods e.g. standard Bayesian calibration.

2.2 Extended Kalman filter

The complication of calibrating a nonlinear model with the Kalman filter, which was developed [34] for models that are linear in their parameters, can be handled with linearization. In the extended Kalman filter (EKF) [28], the state transition and observation models are linearized to maintain the usual Kalman update formula. Since we assume the stress is the only observable variable, the observation model is provided by the material model $\mathbf{m}$, and the appropriate parameter sensitivities are

$$\mathsf{A}_k = \partial_{\boldsymbol{\theta}} \mathbf{m}\big|_{\boldsymbol{\theta}_{k-1},\mathbf{x},\mathbf{z}}, \tag{4}$$

where k is the step, so that the linearization

$$\mathbf{m}(\mathbf{x},\mathbf{z};\boldsymbol{\theta}_k) \approx \mathbf{m}(\mathbf{x},\mathbf{z};\boldsymbol{\theta}_{k-1}) + \mathsf{A}_k[\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}] \tag{5}$$

is sufficiently accurate. This linearization provides a mapping from the parameter covariance $\boldsymbol{\Sigma}$ to the observable output covariance $\mathsf{A}\boldsymbol{\Sigma}\mathsf{A}^T$.

Remark Note that for a linear model $\mathbf{y} = \boldsymbol{\theta}\mathbf{x}$, like the elastic response described in "Appendix A", the sensitivities A increase with $\mathbf{x}$. This relationship has ramifications on the KL divergence reward discussed in Sect. 3.

We assume the (observable) data d corresponds to the model plus uncorrelated (measurement) noise

$$\mathsf{d} = \mathbf{m}(\mathbf{x},\boldsymbol{\theta}) + \boldsymbol{\varepsilon}, \tag{6}$$

where parameters $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ and noise $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0},\mathsf{R})$ follow normal distributions. The (continuous time, noiseless) hidden state transition model is

$$\dot{\boldsymbol{\theta}} = \mathbf{0} \tag{7}$$

$$\dot{\mathbf{z}} = \mathbf{f}(\mathbf{x},\mathbf{z};\boldsymbol{\theta}), \tag{8}$$

where states comprised of model parameters $\boldsymbol{\theta}$ and hidden material state $\mathbf{z} = \{\epsilon^p, \lambda\}$. Since the parameters $\boldsymbol{\theta}$ are fixed, their state transition is the identity. In discrete time, the state transition model become difference equations. For simplicity we assume the observations are complete, in the sense that all components of $\mathbf{y} = \mathbf{m}$ can be compared to data d. Due to the yield surface constraint (33), the plasticity model exemplar is actually a system of differential algebraic equations (DAEs). Handling DAEs in a KF requires additional care; "Appendix B" describes a rigorous treatment based on the work of Catanach [4]. For the results given in Sect. 4, we merely perform the chain rule

$$\mathsf{A} = \partial_\theta \mathbf{m}(\mathbf{x},\mathbf{z};\boldsymbol{\theta}) = \partial_{\boldsymbol{\theta}}\mathbf{m} + \partial_{\mathbf{z}}\mathbf{m}\,\partial_{\boldsymbol{\theta}}\mathbf{z}, \tag{9}$$

which accounts for the change in $\mathbf{z}$ over the step, but not the entire history.

The residual, residual covariance, and so-called Kalman gain are

$$\mathsf{r}_k = \mathsf{d}_k - \mathbf{m}(\mathbf{x}_k,\mathbf{z}_k,\boldsymbol{\mu}_{k-1}) \tag{10}$$

$$\mathsf{S}_k = \mathsf{A}_k \boldsymbol{\Sigma}_{k-1} \mathsf{A}_k^T + \mathsf{R} \tag{11}$$
$$\mathsf{K}_k = \boldsymbol{\Sigma}_{k-1} \mathsf{A}_k^T \mathsf{S}_k^{-1}, \tag{12}$$

respectively. They are used to update the parameter ($\boldsymbol{\theta}$) mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$:

$$\boldsymbol{\mu}_k = \boldsymbol{\mu}_{k-1} + \mathsf{K}_k \mathsf{r}_k \tag{13}$$
$$\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}_{k-1} - \mathsf{K}_k \mathsf{S}_k \mathsf{K}_k^T \tag{14}$$

given the data d_k provided at step k.

The KF is an iterative method that requires initialization. Initial values of the mean $\boldsymbol{\mu}_0$ and covariance $\boldsymbol{\Sigma}_0$ represent the prior information of the parameters $\boldsymbol{\theta}$. We normalized the parameters so that they were close to $\mathcal{O}(10^0)$ and this made a diagonal $\mathcal{O}(10^{-1})$ $\boldsymbol{\Sigma}_0$ matrix a reasonable prior. In the present context of low measurement noise experiments, the noise variance R is fixed and a scaling of the identity matrix with the scale chosen *a priori* to be small $\mathcal{O}(10^{-6})$; however, it can be calibrated as well. Furthermore, in this case, the diagonal of R primarily regularizes the inversion of S. Predictions can be made using the current values of the mean parameters $\boldsymbol{\mu}$ and their covariance $\boldsymbol{\Sigma}$:

$$\mathsf{y}^* \sim \mathcal{N}(\mathbf{m}(\mathbf{x}^*, \boldsymbol{\mu}), \mathsf{A}^* \boldsymbol{\Sigma} \mathsf{A}^{*T}). \tag{15}$$

It is important to note that for highly nonlinear models, traditional approaches like the Extended Kalman Filter (EKF) may not provide the accuracy needed. Instead, more advanced methods like the Unscented Kalman Filter (UKF) [32] or Ensemble Kalman Filter (EnKF) [13] can be implemented, which can achieve higher accuracy and/or computational efficiency. In addition to the UKF, there are other nonlinear Kalman filtering methods [8] such as particle filters or dual filter estimation frameworks, which can offer advantages over the EKF. However, we have chosen to use the EKF in our work due to its simple implementation, flexibility, and compatibility with DAE solvers (see "Appendix B") and policy updates in reinforcement learning, i.e. both EKF and machine learning networks use derivatives to update a function with current knowledge. It is important to note that changing the filter to any of the aforementioned methods would not pose any technical difficulties since the RL algorithm implemented is agnostic to the estimator.

2.3 Switching Kalman filter

Plasticity models consist of two response modes, (a) an elastic one where only some of the parameters are influential, and (b) a plastic one where all parameters affect the response. A customization of the EKF is, therefore, necessary to adapt to these physical regimes and avoid errors in the accumulation of latent variables in the plastic mode. With the exemplar in "Appendix A", no information on the plastic response is available from the material in its reference/starting state. If the exact state at which the material changes to a plastic response was known, it would be trivial to select the appropriate KF; unfortunately, uncertainty in the yield stress causes erroneous Kalman updates of the parameters and hidden state.

Two solutions are implemented to handle the discontinuity in response and the switching between elastic and plastic modes. The first sets a convergence tolerance criteria on the parameters. This *ad hoc* version of the EKF sets parameter sensitivities to zero when convergence is detected. This masks the sensitivity matrix A with a diagonal matrix M that has unit entries for parameters that have not converged and zeros for ones that have. This masking has the effect of fixing the converged parameters at their current mean. In particular, it allows the elastic parameters to be calibrated first while the material state is still within the yield surface, and then the calibration of the plastic parameters ensues, with fixed elastic parameters, once yield is encountered. For this simple method, convergence is assessed with a Cauchy convergence criterion on the mean $\boldsymbol{\mu}$ for the particular parameter and a check that the current diagonal entry of the covariance $\boldsymbol{\Sigma}$ has decreased from its previous value.

Alternatively, an extended *switching* Kalman filter (SKF) is a more rigorous approach and can deal with discontinuous parameters, such as elastic vs. plastic response, by simultaneously competing multiple models (in this case, material modes). This competition allows for a more accurate prediction of material behavior when there is a latent variable with abrupt or discontinuous changes. We implement a generalized pseudo-Bayesian (GPB) algorithm [48], which takes Gaussian distributions associated with each material mode and collapses them into a Gaussian mixture of (at most) two previous history steps. The second-order generalized pseudo-Bayesian algorithm (GPB2) is a good balance between accuracy and the complexity of longer views of the history. Full details of the GPB2 algorithm applied to the present context are given in "Appendix C".

3 Deep reinforcement learning for experimental design with extended Kalman filter

This section describes the incorporation of the extended Kalman filter (EKF) into deep reinforcement learning (DRL) for experimental designs that maximize information gain. Based on the reward hypothesis of reinforcement learning [50], we assume that the goal of a calibration experiment can be formalized as the outcome of maximizing a cumulative reward defined by the information gain estimated from the EKF. As such, the design of the experiment that generates a model can be viewed as a game where an agent seeks

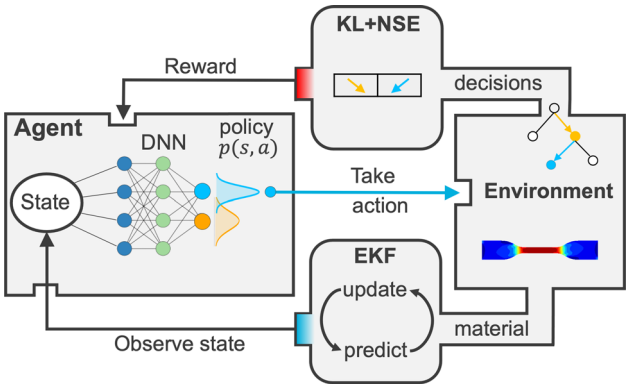

Fig. 2 Workflow of the KF-based deep reinforcement learning

feasible actions in an experiment to maximize the total information gain, and hence the best-informed model parameters, as shown in Fig. 2 which illustrates the workflow of the proposed RL approach.

In this section, we have two objectives: (1) to describe how an experiment is run as an MDP in an RL framework enhanced by the EKF, and (2) to provide some key highlights on how the DRL agent that runs the experiment is trained. Section 3.1 first formulates a family of mechanical tests as an MDP. Then, the policy that gives the action selection is described in Sect. 3.2 and the action of the experiments represented by a decision tree is described in Sect. 3.3. The action-state-reward relationship quantified by the EKF is described in Sect. 3.4. With these ingredients properly defined, the Monte Carlo tree search (MCTS) used to update the policy and improve the DRL agent's decision-making process is provided in Sect. 3.5.

3.1 Experiments as a Markov decision process

Here, we consider an experiment conducted by an agent as a single-player game formulated as MDP where the agent interacts with the environment based on its state in a sequential manner. This MDP can be a tuple $(\mathcal{S}, \mathcal{A}, \boldsymbol{s}, \gamma)$ consisting of state set $\mathcal{S}$, action set $\mathcal{A}$, and joint probability of reward $\mathcal{R}$ for a given state $\boldsymbol{s}$ and discount factor γ that balances the relative importance of earlier and later rewards.

For the experiment we set the RL *actions* with all the allowed experimental control actions that affect the strain $\boldsymbol{\epsilon}$, and the RL *state* set as all the results of possible actions over a sequence of steps. To accommodate history effects within the MDP we expand the notion of state to include all prior loading history. Even for a discrete set of n actions the state space increases exponentially with time starting from the reference/initial state and growing n-fold with every step.

In an *episode*, the agent makes a sequence of decisions of (discrete) actions $\Delta\boldsymbol{\epsilon}_k$ to take. An episode is a complete traversal of the decision tree from the root node to a leaf node that corresponds to the end state of an experiment after a fixed number of steps. In effect an episode is a particular experiment defined by selected control actions $\{\mathbf{x}_i, i = 1, k\}$. For each decision within the same episode, an agent takes an action based on the policy $\boldsymbol{p}(\boldsymbol{s}, \boldsymbol{a}) \in [0, 1]$ that suggests the probability of the preferred choice. Each action taken by the agent must lead to an update of the (observable) state $\mathbf{y}_k$. This updated state is then used as a new input to generate the next policy value for the available actions. This feedback loop continues until the particular experiment/episode concludes.

The RL *environment* is defined as any source of observations that provides sufficient information to update the state. In this work, our goal is to introduce the KF as a component of the environment where the estimation of the KF state is used to: (1) provide an enhanced state representation, as well as (2) constitute the reward of the experiments to redefine the objective of the game, which is now formulated to maximize the information gain of the experiments. Finally, learning occurs whenever the deep neural network that predicts the policy is retrained in an RL *iteration* [59, 66]. An iteration can be called upon whenever a sufficient amount of new state-action pair labels is collected during the experiments. The ratio between the number of iterations and episodes as well as the architecture of the neural network itself are both hyperparameters that can be fine-tuned for optimal performance.

3.2 Policy represented by deep neural network

In principle, the RL *policy* that guides the agent to select rewarding actions can be determined by directly sampling the states, actions, and rewards among all the existing options. However, such an approach is not feasible when the number of possible paths to conduct experiments becomes too large. A classical example includes the game of Go [60] where exhausting all the possible moves is intractable. This large number of paths is also common in experiments where a sequence of decisions has to be made both before and during the experiments. More discussion of this point is given in Sect. 4.

As shown in, for instance Silver et al. [58–60], an option to manage this curse of high dimensionality is to approximate the policy (and potentially other sets in the tuple) via a trained neural network and use MCTS to improve the efficiency of the sampling. In our work, the deep neural network is solely designed for the purpose of generating the policy value when given a particular combination of state $\boldsymbol{s} \in \mathcal{S}$ and action $\boldsymbol{a} \in \mathcal{A}$. Note that policy function can also be approximated by other techniques, including mathematical expressions obtained from symbolic regression [39] and the choice of the approximation may affect the difficulty of the representation problem. In this paper, a standard overparameterized deep neural network is adopted throughout the entire training process. The neural network follows a standard mul-

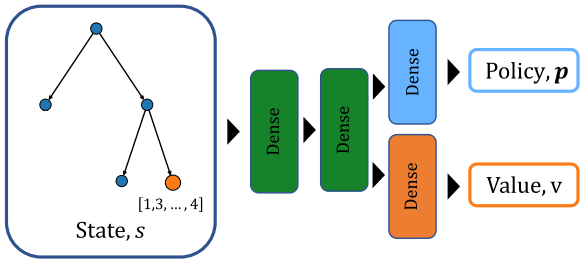

Fig. 3 A schematic of the reinforcement learning policy-value neural network architecture. The network inputs a state s corresponding to a node of the decision tree and outputs a policy $\boldsymbol{p}$ and value v

tilayer perceptron feed-forward architecture. The network takes RL state $\boldsymbol{s}$ that corresponds to a node of the decision tree as inputs and is subsequently fed into a series of dense hidden layers. The network has two outputs: a policy vector $\boldsymbol{p}$ representing the probability of taking the action $\boldsymbol{a}$ from the current state $\boldsymbol{s}$ and a predicted scalar value v estimating the reward from the state (see Fig. 3). A policy is a probabilistic function in part to allow for exploration as well as exploitation of previous information such as the rewards for previous actions. Since the policy represents a probability, a softmax layer is used for this output, while a tanh layer is used for the continuous value output. Additional specifics of the network's architecture and training hyperparameters are provided in Sect. 4.

The mapping between the input state $\boldsymbol{s}$ and the output policy $\boldsymbol{p}$ and value v is, thus, represented by a neural network approximator $\widehat{f}$. The approximation is defined such that $(\widehat{\boldsymbol{p}}, \widehat{v}) = \widehat{f}(\boldsymbol{s} \mid \boldsymbol{W}, \boldsymbol{b})$, where $\widehat{\boldsymbol{p}}$ and $\widehat{v}$ are the approximated values of the policy vector and value, respectively, and $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weight matrices and bias vectors of the architecture, respectively, to be optimized with stochastic gradient descent during the network training. The training objective for the training samples $i \in [1, \ldots, N]$ is to reduce the mean squared loss:

$$\boldsymbol{W}', \boldsymbol{b}' = \underset{\boldsymbol{W},\boldsymbol{b}}{\operatorname{argmin}} \left(\frac{1}{N} \sum_{i=1}^{N} \left(\| \boldsymbol{p}_i - \widehat{\boldsymbol{p}}_i \|_2^2 + \| v_i - \widehat{v}_i \|_2^2 \right) \right). \tag{16}$$

Remark Note that this policy corresponds to the reward but is not the reward itself, as the policy must be a trade-off between exploration and exploitation [63]. This balance will be further discussed in Sect. 3.5.

Remark The application in this current work is developed for calibrating plasticity models for metals where negligible measurement noise (on the order of 0.01%) is observed in the target data. As such, we do not investigate the impact of noise on the DRL algorithm's convergence in this work. It is expected that observed noise will not significantly affect the algorithm's convergence. It is worth noting that the reinforcement learning states refer to discrete decisions in the experiment setup and there is no direct definition of noise in the context of the experimental design. Nevertheless, our application aims to train the algorithm offline and deploy it in a lab setting for real-time decision-making, where noise is not expected to significantly impact the algorithm's performance. The policy-value network $\widehat{f}(\mathbf{s})$ inputs only the current state, which represents the history of previous decisions, to make a forward prediction for the optimal next experimental step and is not affected by the observed material response.

3.3 Action representation: decision tree for experiments

In a Markov decision process, a state from an earlier decision is connected to all of the corresponding possible child states through an action. Furthermore, an earlier action may affect the latter states but a latter decision has no effect on the prior state. Hence, all the possible actions and states together are connected in a directed and acyclic manner, which, in graph theory [68] is referred to as a *poly-tree*. We will employ the terminology of graph theory and refer the initial state of an experiment as the *root* (the vertex with only outgoing edged in the tree) and the end of the experiment as the *leaf* (a vertex with only an incoming edge). As such, the decision tree of an experiment is a specific poly-tree that represents all the possible states at vertices, each connected by edges representing the corresponding actions available during an experiment. The role of the policy of the RL agent (the actor) is to determine the action when a state is given as input such that an RL agent may create a path that started from the root and end at one of the leaves of the decision tree.

3.4 Environment: states and rewards of the design-of-experiment problem

As pointed out by Reda et al. [54], a key ingredient to generating the effective learned policy and value of the states that often get overlooked is the parameterization of the environment, in which the DRL agent/algorithm interacts. In the design-of-experiment problem, the environment provides feedback caused by actions selected by the DRL agent. This feedback can be in the form of a state or reward.

We formulate the design-of-experiment as a multi-objective problem in which we want to (1) minimize the expected value of the discrepancy between the predictions made by the calibrated models and the ground truth (by maximizing the Nash–Sutcliffe efficiency (NSE) index of the **calibrated model**, see (23)) and (2) improve the efficiency

of the **experiments** by maximizing the Kullback–Leibler (KL) divergence, and hence maximizing the information gain between states. While both measurements provide a valuation of the actions through measuring the improvement of the model due to additional data gained from new actions in the experiment, the KL divergence does not require additional sampling, and hence is more cost-efficient. However, the extended Kalman filter (EKF) used to calculate the KL divergence also has well-known limitations, such as the need to make a sufficiently close initial guess to avoid divergence triggered by the linearization process and the consistency issue due to the underestimation of the true covariance matrix.

As such, we employ a mixed strategy in which we only approximate the NSE index by under-sampling a few states outside of the training data region. This cheaper approximated NSE index is augmented with the KL divergence as the combined reward to circumvent the inconsistency and divergence issue of the EKF, whereas the KF-predicted mean and variance of the calibrated parameters are used, in addition to the loading history of the experiments, to represent the enhanced state of the experiment.

To fully assess the usefulness of the NSE and KL divergence reward metrics for experiment design, we conduct a comprehensive study of each reward metric individually and in combination. In Sect. 4, our numerical experiments are designed to validate the capacity of each reward metric for designing experiments and their ability to work together to produce optimal experiment designs. We want to use the KL divergence reward metric because of its cost efficiency; however, we also incorporate the NSE index in our mixed reward when necessary to address the limitations of the EKF calibration. This approach enables us to achieve a balance between experiment efficiency and prediction accuracy, and it underscores the flexibility and versatility of our proposed algorithm for designing optimal experiments in a variety of settings.

For brevity and to avoid confusion with the estimated added reward within each state update, we would use the term *game score* to refer to the total reward accumulated within an experiment game, while the history of the performances is measured by monitoring the distribution of the game score against the policy neural network iteration at which the policy neural network is re-trained.

3.4.1 Information-gain reward: Kullback–Leibler divergence

The KL divergence has a lower-bound value of 0 when the model perfectly describes the data, meaning that no information was gained. Similarly, a perfect NSE score has an upper-bound value of 1 when the model perfectly replicates the data. There is no direct relationship between the NSE score and the KL divergence, but both measure aspects of model accuracy and parameter uncertainty.

Generally speaking, the KL divergence is a measure of the statistical change between two probability distributions. Denoted as $D(\pi_1||\pi_0)$ where π_0 is a reference prior probability distribution and π_1 is the updated posterior probability distribution. The difference is calculated as the expectation of the logarithmic difference between distributions with respect to the posterior probabilities $\pi_1(x)$:

$$D_{\mathrm{KL}}(\pi_1||\pi_0) = \int_{\mathcal{X}} \pi_1(x) \log\left(\frac{\pi_1(x)}{\pi_0(x)}\right) \mathrm{d}x, \tag{17}$$

where $x \in X$ represents a shared probability space.

By formulating a reward based on the KL divergence, we can get a measure of how informative new data d_{n+1} is in updating the distribution of the estimated parameters $\pi(\boldsymbol{\theta}|D)$ (see Fig. 4):

$$\begin{aligned}\Delta\mathrm{KL} = &\int \pi(\boldsymbol{\theta}|D_{n+1}) \log \frac{\pi(\boldsymbol{\theta}|D_{n+1})}{\pi(\boldsymbol{\theta})} \mathrm{d}\boldsymbol{\theta} \\ &- \int \pi(\boldsymbol{\theta}|D_n) \log \frac{\pi(\boldsymbol{\theta}|D_n)}{\pi(\boldsymbol{\theta})} \mathrm{d}\boldsymbol{\theta},\end{aligned} \tag{18}$$

where $D_n = \{\mathsf{d}_k, k \le n\}$. For a multivariate Gaussian distribution, like the one we have with the KF, the KL divergence is analytic:

$$\begin{aligned}\mathrm{KL}_k = \frac{1}{2}\Bigg(&\log\frac{\det\boldsymbol{\Sigma}_0}{\det\boldsymbol{\Sigma}_k} + \mathrm{tr}\left(\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\Sigma}_k\right) \\ &+(\boldsymbol{\mu}_k - \boldsymbol{\mu}_0)\boldsymbol{\Sigma}_0^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_0) - n_{\boldsymbol{\theta}}\Bigg),\end{aligned} \tag{19}$$

where $n_{\boldsymbol{\theta}}$ is the number of parameters. A reward for the experiment over n steps can be formulated as

$$\mathcal{R}_{\mathrm{KL}} = \sum_{k=1}^{n} \Delta\mathrm{KL}_k, \tag{20}$$

where $\Delta\mathrm{KL}_k \equiv \mathrm{KL}_k - \mathrm{KL}_{k-1}$.

For example, the reward for a single step in a linear elastic material with bulk modulus K and shear modulus G has covariance

$$\boldsymbol{\Sigma}_k = \begin{bmatrix} \mathrm{var}(K)_k & \mathrm{cov}(K,G)_k \\ \mathrm{cov}(G,K)_k & \mathrm{var}(G)_k \end{bmatrix} \tag{21}$$

and mean vector of the estimated bulk and shear moduli

$$\boldsymbol{\mu}_k = \begin{bmatrix} K_k \\ G_k \end{bmatrix}. \tag{22}$$

As illustrated in Fig. 4, the distribution of the parameters begins with a pre-selected *prior* value for the mean $\boldsymbol{\mu}_0$ and covariance $\boldsymbol{\Sigma}_0$ before measuring any data. The KL divergence initially increases since the new data is more

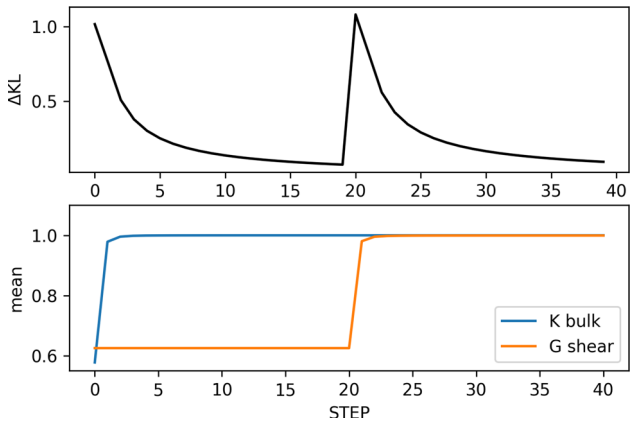

Fig. 4 Calibration of the isotropic elastic material model (bottom), and the incremental KL-based reward along the calibration path (top) that starts with a volumetric deformation and then switches to shear

informative, but then the rewards tail off as less information is gained from subsequent samples and the mean parameter values converge. When the material response exhibits a discontinuity, as in the onset of plastic deformation, the Kalman switching filter Sect. 2.3 can be used to select the deformation mode (elastic or plastic) that best captures the data at the current step. Section 4.2 provides a detailed illustration of this switching mechanism.

3.4.2 State represented by action history and Kalman filter prediction

The RL *state* represents the complete information necessary to describe the consequences of actions taken by the agent from the beginning of the game to the current step.

In this work, the state of the experiment contains two components: (1) the entire loading history of the experiments since the beginning of the test and (2) the calibrated mean and covariance of the material parameters (see, for instance, Eqs. (21) and (22)). Depending on the types of experimental tests, the loading history can be represented via different parametrizations [21]. For a strain-controlled test, the available action choices are the increments of individual components of the strain tensor, and hence the loading history can be represented by a stack of these strain increments stored in the Voigt notation, which leads to a matrix of dimension $6 \times N_{\text{step}}$ where 6 is the number of independent components of the symmetric strain tensor and N_{step} is the total number of strain increments. For future time steps that are not yet executed, the corresponding columns of the matrix are set to zero. For the case where the loading combination is more limited, such as a shear box apparatus, loading history can be sufficiently represented by a vector.

This state is augmented by the mean and covariance of the material parameters predicted by the EKF. For instance, in the case of linear elasticity, the state can be represented by the mean of two elastic material parameters and the components of the symmetric covariance matrix, which contains three independent components. In other words, the state representation of the experiment employs both a representation of the loading path and the parameter distribution provided by the EKF. Together, they form the RL state, which is used as the input for the policy neural network shown in Fig. 3.

Remark The decision points for the MDP can be on a larger step size than the KF, i.e., the KF can be sub-cycled for additional stability and accuracy.

Remark Note that while incorporating more sensory information may provide more information to learn the policies and values, in practice adding more information may also increase the dimensionality of the state representation, and hence significantly increase the difficulty of the DRL. The exploration of more efficient state representation methods and the implications of more efficient state representations are active research areas [57], but are out of the scope of this study.

3.4.3 Forecast prediction reward via an under-sampled Nash–Sutcliffe efficiency index

The Nash–Sutcliffe efficiency index is a simple normalized measure of the discrepancy between model predictions and ground truth data. The NSE index is:

$$\mathcal{R}_{\text{NSE}} = 1 - \frac{\sum_{k=1}^{N_{\text{data}}} |\mathsf{d}_k - \mathbf{m}(\boldsymbol{\theta})|}{\sum_{k=1}^{N_{\text{data}}} |\mathsf{d}_k - \text{mean}(\mathsf{d})|} \in (-\infty, 1], \tag{23}$$

where d_k is the data point at step k, $\mathbf{m}(\boldsymbol{\theta})$ is the calibrated model at the end of an episode and $\text{mean}(\mathsf{d})$ is the mean value of the dataset. To measure the mean of the forecast accuracy via the NSE index, the number of data points N_{data} must be sufficiently large such that empirical loss and population loss lead to a sufficiently small difference (refer to Gnecco et al. [18]). Hence, this sampling requirement can be costly, especially for physical experiments that require labor, time and material costs. As such, we propose an under-sampling and static strategy where the data points d_k collected to calculate the NSE reward are sampled prior to the training of the DRL agent [66, 67]. This reduction in the sampling size may, in principle, leads to increasing bias by missing data of statistical significance. However, since the sub-reward $\mathcal{R}_{\text{NSE}}$ introduced here merely functions as a regularization term for the KL reward in (20) to guide the early exploration of the DRL agent, no significant issues manifested in the numerical experiments showcased in Sect. 4.

3.4.4 Combined reward for multi-objective experiments

As mentioned earlier, approximating the parameter distributions of highly nonlinear function via the standard EFK may lead to inaccurate state estimation. This inaccuracy, in turn, can inflate the KL divergence and leads to an over-optimistic reward for the DRL agent [40]. While there are alternatives, such as unscented Kalman filter [31] and the ensemble Kalman filter Evensen [13] that can circumvent the loss of accuracy due to the linearization of the nonlinear function, a simpler implementation approach is to add the NSE reward into the DRL framework during the training process to modify the exploitation behavior of the DRL agent.

As such, we introduce a weighted average reward that augments the KL divergence with the under-sampled NSE index:

$$\mathcal{R}_{\text{total}} = w_{\text{NSE}}\mathcal{R}_{\text{NSE}} + w_{\text{KL}}\mathcal{R}_{\text{KL}}, \tag{24}$$

where w_{NSE} and w_{KL} are weighting factors for the two rewards. To promote the neural network representation training, the rewards in the following applications are rescaled to be order 1 based on an estimate of the range of expected values. When using the mixture of the two rewards, $\mathcal{R}_{\text{NSE}}$ and $\mathcal{R}_{\text{KL}}$, we rescaled each to the range of [0, 1] and choose the weights such that $w_{\text{NSE}} + w_{\text{KL}} = 1$ so that the total reward is in the same range. Unless stated otherwise, all the numerical experiments are conducted with equally weighted sub-reward, i.e., $w_{\text{NSE}} = w_{\text{KL}} = 1/2$.

In order to implement this joint reward during the training of the DRL algorithm in practice, we perform the tasks described in Sects. 3.4.1 and 3.4.3 sequentially to calculate the sub-rewards in Eq. (20) and Eq. (23). For Eq. (20), we update the KL divergence reward with the information gain metrics as the EKF model is calibrated on new experiment design paths from the tree search. For Eq. (23), we test the newly calibrated model in a forward prediction against a subsample of blind data d_k—this data has been sampled prior to the start of the DRL training. The two sub-rewards are scaled, weighted, and summed to provide the joint reward in Eq. (24).

Remark Note the assumption of a fixed-step budget is not overly constraining. If the incremental reward tails off, and the parameter uncertainties are acceptable, the experiment can be truncated early, potentially with significant cost savings.

Remark There is distinction between (a) the reward calculated from the KL divergence, NSE index or a combination of the two, and (b) the state value and expected cumulative reward predicted by the policy. As in the RL literature, we use *reward* to refer to the former and *value* to reference the latter.

3.5 A Monte Carlo tree search with Kalman reward estimator

Here, we consider the case where planning and learning are both needed to maximize the information gain within a limited number of actions. The learning objective is the optimal calibration of a material model. For costly physical experiments, the goal of designing an experiment is not just to finish a task (e.g., calibration, discovery and uncertainty quantification), but to finish the task within the allocated resources. The multitude of decisions complicates this goal. For instance, in a biaxial compression/extension test where the specimen can be compressed/extended in two directions, there will be $4^{50} \approx 10^{30}$ possible ways to run an experiment with 50 incremental time steps. Sampling all the available paths and selecting actions that maximize the optimal reward can be a feasible strategy only if all paths can be visited. The classical Monte Carlo simulation is not feasible as random sampling is not sufficiently efficient to discover the optimal policy given limited opportunities to visit only a small fraction of the possible paths. As such, a tactic to balance exploitation and exploration is necessary [47].

In this work, Monte Carlo tree search (MCTS) is used to enable estimation of the policy value $\boldsymbol{p}(\boldsymbol{s}, \boldsymbol{a})$ by visiting state-action pairs according to an optimality Eq. (25), that balances exploitation and exploration of the decision tree. It is necessary to setup a material simulator and model calibrator before beginning the policy search in algorithm 1. The output of these components constitute the observable variables in the RL environment that help the agent learn and improve its policy. Code implementations of the simulator and calibrator are not covered here in detail, but the material exemplar is found in "Appendix A". An KF is used for both evaluating the reward and calibrating material parameters. The EKF and the SKF methods we empolyed are described in Sects. 2.2 and 2.3.

Once the policy DNN is initialized (see Sect. 4.1), the iterator i starts the outer loop. Each iteration executes a policy update after an inner loop (iterator j) over a number of episodes, where the outcome of each episode represents the result of the current policies for the design of the experiment. Update of the policy DNN is accomplished with a stochastic gradient descent optimizer, see Sect. 3.2. During an episode, the decision tree is populated with state and action pairs until the end of the tree is reached. Each action is selected according to a probability of moving to a state of maximum value. This value is estimated by a policy $\boldsymbol{p}_i(\boldsymbol{s}, \boldsymbol{a})$ at the i^{th} iteration.

At the end of each episode, the history of control actions taken are fed as input into the material simulator and calibrator. The reward $\mathcal{R}_{KL}$ assigned to an episode is calculated

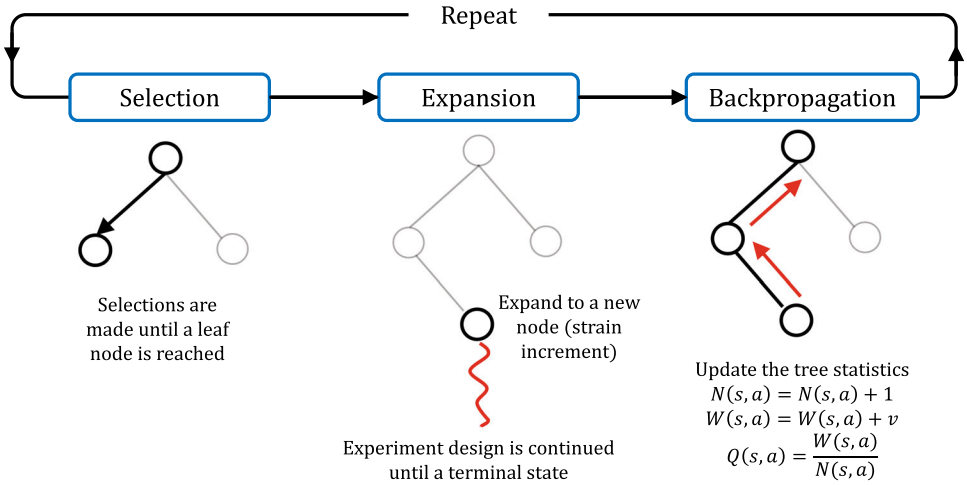

Fig. 5 Monte Carlo tree search steps

using the KL divergence (20) which is based on the mean and covariance posterior updates. The KL divergence is a measure of the expected amount of information gained about the system state. A higher reward indicates that the agent is learning more about the system as it informs the model calibration. The reward can also be calculated according to (23) or a mixture, (24). At the conclusion of each iteration, the policy is updated using the training examples generated during the episode simulations (step 26, algorithm 1). After several iterations, the policy becomes a good estimator of action value and the exploitation of high value actions will be balanced by (25) which diminishes the probability of selecting repetitive pathways.

Algorithm 2 details and the MCTS in the inner loop in Algorithm 1 (7–13). Within one episode, the MCTS we employed repeatedly performs the three steps illustrated in Fig. 5) [58]:

Once the search is complete, the search probabilities/policies π are evaluated based on how often a state was traversed:

$$\pi(a \mid s) = \frac{N(s,a)^{1/\tau}}{\sum_b N(s,b)^{1/\tau}}, \tag{27}$$

where N is the visit count of each move from the root node and τ is a temperature parameter, which is another parameter controlling the exploration. Thus, the discovered policies are proportional to the number of visits in each state.

After a fixed number of episodes, there will be enough labeled data of state, action, policy, and reward to update the policy network. At this point, an *iteration* of the policy network is conducted, and all the collected data within the episode will be used to retrain the policy neural network. In this context, iteration refers to a policy update which is conducted after a certain number of episodes have collected sufficient new reward data.

4 Numerical experiments

In this section, we introduce three design-of-experiments examples to: (1) validate the implementation of the EKF-DRL algorithm, (2) provide benchmarks against the classical DRL approach that employs the Nash–Sutcliffe sampling to estimate the rewards, and (3) showcase the potential applications of the EKF-DRL algorithm for designing mechanical experiments for models of high-dimensional spaces that require a significantly larger decision tree for long-term planning due to history dependence and other complications.

We focus on the calibration of traditional physical models whose limited parameterization (relative to potentially more expressive models) helps to control the growth of the decision trees. We also assume that have noiseless observations of experimental data, which is motivated by the low measurement noise of modern testing equipment. These simplifying assumptions allow us to focus on the primary challenge of designing a UQ-driven reinforcement learning design of experiments strategy for history-dependent materials.

4.1 Implementation verification 1: experiment for linear isotropic elastic materials

For a linear isotropic elasticity model, there are two independent elastic moduli. Hence two linearly independent observations are sufficient to provide the necessary information to determine the model parameters [3]. For example, a single step in a uniaxial test should be sufficient to identify any pairs of independent linear elastic parameters if the stress is observed in independent directions (e.g., the Poisson effect is observed as a supplement to the surface traction and uniaxial stress along the loading direction).

In this numerical experiment, we introduce a virtual test where the EKF-DRL agent observes the volumetric and deviatoric stress of a linear isotropic elastic material whenever the

Algorithm 1 Reinforcement learning for Design of Experiments

Require: The definitions of the experiment game: environment, states, actions, rewards.

Initialize the experimentalist policy/value network DNN. For fresh learning, the network is randomly initialized. For transfer learning, load pre-trained network instead.
for i in iterations **do**
  Initialize empty sets of the training examples $trainExamples \leftarrow \{\}$.
  for j in episodes **do**
    Initialize the starting game state vector $\boldsymbol{s}$ (container for experiment control history).
    Initialize empty tree of the Monte Carlo Tree search (MCTS), by setting containers for edge visits $N(\boldsymbol{s}, \boldsymbol{a})$, and mean action values $Q(\boldsymbol{s}, \boldsymbol{a})$
    while True **do**
      Check for all allowed actions at the current state $\boldsymbol{s}$ according to the games rules.
      Get the action probabilities $\boldsymbol{p}(\boldsymbol{s}, .)$ for all allowed actions by performing repeated MCTS simulations.
      Sample action $\boldsymbol{a}$ from the probabilities $\boldsymbol{p}(\boldsymbol{s}, .)$
      Modify the current game state to a new state $\boldsymbol{s}$ by taking the action $\boldsymbol{a}$.
      if $\boldsymbol{s}$ is the end state of the game of the experimentalist **then**
        break
    Calibrate material model using EKF with the selected paths in the decision tree.
    if the information-gain reward (Sect. 3.4.1) is used **then**
      Evaluate $\mathcal{R}_{\text{KL}}$ from the model calibration.
      Evaluate the total reward $\mathcal{R}_{\text{total}} = \mathcal{R}_{\text{KL}}$ of this gameplay.
    if the Nash-Sutcliffe efficiency index reward (Sect. 3.4.3) is used **then**
      Test calibrated model against set of blind experiments and evaluate $\mathcal{R}_{\text{NSE}}$.
      Evaluate the total reward $\mathcal{R}_{\text{total}} = \mathcal{R}_{\text{NSE}}$.
    if the combined reward (Sect. 3.4.4) is used **then**
      Evaluate $\mathcal{R}_{KL}$ from the model calibration.
      Test calibrated model against set of blind experiments and evaluate $\mathcal{R}_{\text{NSE}}$.
      Evaluate the total reward $\mathcal{R}_{\text{total}} = w_{\text{NSE}}\mathcal{R}_{\text{NSE}} + w_{\text{KL}}\mathcal{R}_{\text{KL}}$ of this gameplay.
    Append the gameplay history $[\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{p}(\boldsymbol{s}, .), \mathcal{R}_{\text{total}}]$ to $trainExamples$
  Train the policy/value network DNN with $trainExamples$
Use the trained network DNN of the last iteration to select the optimal experiments for model calibration.
Exit

Algorithm 2 Monte Carlo tree search

1. Selection. A path is determined by picking action according to the estimated policy $\boldsymbol{p}(\boldsymbol{s}, \boldsymbol{a})$ (until a leaf of the tree (the state node with no child) is reached), i.e.,

$$\boldsymbol{a}_t = \underset{\boldsymbol{a}\in\mathcal{A}}{\text{argmin}} \left(Q(\boldsymbol{s}, \boldsymbol{a}) + c_{puct}\, \boldsymbol{p}(\boldsymbol{s}, \boldsymbol{a}) \frac{\sqrt{\sum_{a'} N(\boldsymbol{s}, \boldsymbol{a}')}}{1 + N(\boldsymbol{s}, \boldsymbol{a})} \right), \tag{25}$$

where $\boldsymbol{a}_t$ is the chosen action, c_{puct} is a parameter that controls the degree of exploration. $N(\boldsymbol{s}, \boldsymbol{a})$ denotes the number of visit/time action a is taken at state $\boldsymbol{s}$. In (25), the first term is the Q value in our case, which is the expected value of the reward, i.e.,

$$Q(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}[\mathcal{R}_{\text{total}} | \boldsymbol{a}_t = \boldsymbol{a}, \boldsymbol{s}_t = \boldsymbol{s}], \tag{26}$$

whereas the second term is the upper confidence bound, which can be derived from Hoeffding's inequality [63]. Here we simply average the action value we have collected from $N(\boldsymbol{s}, \boldsymbol{a})$ sampling as the $Q(\boldsymbol{s}, \boldsymbol{a})$ value. Meanwhile, the policy values $\boldsymbol{p}(\boldsymbol{s}, \boldsymbol{a})$ are estimated from the deep neural network trained in the last iteration. The key feature of this action selection model is that it reduces the value of the second term for a given action when it is visited more frequently. Consequently, this reduction triggers a mechanism for the agent to explore actions with high uncertainty if given the same expected return.
2. Expansion. The available options of the experiment for a given state is added to the tree. Generally speaking, options may vary for different states. In this work, the available options of actions are identical for each state. The allowed actions from every state are the same increments of the strain tensor component. The number of allowed actions/strain component increment options depends on the complexity of the respective material and respective decision tree.
3. Back-propagation. At the terminated state, the KL divergence and other feasible indices that yield the reward are calculated (see Fig. 2). All the policies between the root node and leaf nodes will be updated. The visit count $N(s, a)$ at every node traversed is increased by 1 and the action value $Q(\boldsymbol{s}, \boldsymbol{a})$ is updated to the mean value.

corresponding volumetric and shear strains are prescribed. This agent is then tasked with designing a strain-controlled mechanical test of a specimen to identify the bulk K and shear G moduli. The decision tree that includes the possible paths for two incremental steps is shown in Fig. 6.

Since the specimen is strain-controlled in two directions, there are two optimal strategies: (1) first shear then compress the specimen or (2) first compress then shear the specimen. As with model-free Q learning [19], one can simply visit all the possible states and the optimal strategy will be learned. As such, the goal of this numerical example is to verify the implementation where the optimal design is trivial in this sense.

A synthetic measurement model was used to verify convergence to the correct policy as a benchmark for the RL algorithm. A training experiment was performed for 10 policy training iterations. Every iteration has 10 game episodes. As mentioned, each episode corresponds to a complete traversal of the decision tree from the root node to a leaf node to design the experiment strain path. This includes gathering the linear elasticity data, calibrating the Kalman filter (KF) model on that data and calculating the information gain reward. During each episode, we gather information for the states $\boldsymbol{s}$ traversed as well as the corresponding policies $\boldsymbol{p}$ and values v. At the end of every iteration, the RL neural network is trained on the $(\boldsymbol{s}, \boldsymbol{p}, v)$ data collected as described in Sect. 3.2. The exploration parameter (25) is set to linearly reduce every iteration, starting at $c_{puct} = 10$ and being equal to $c_{puct} = 1$ at iteration 10. This parameter was chosen to encourage exploration more in earlier iterations, sample the

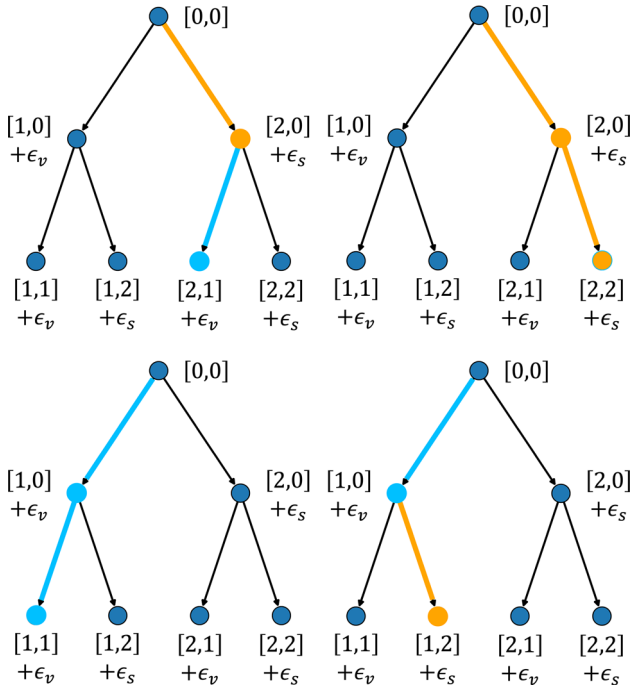

Fig. 6 All possible paths (orange/blue lines) of the decision tree for the volumetric/deviatoric test within two steps. The weight of the nodes stores the state, which represents the current strain of the specimen, and the edge weight is the strain increment. The integers in the state vector indicate 0 for no action, 1 for compression and 2 for shear, where the first and second components of the state vector record the choice made by the agent

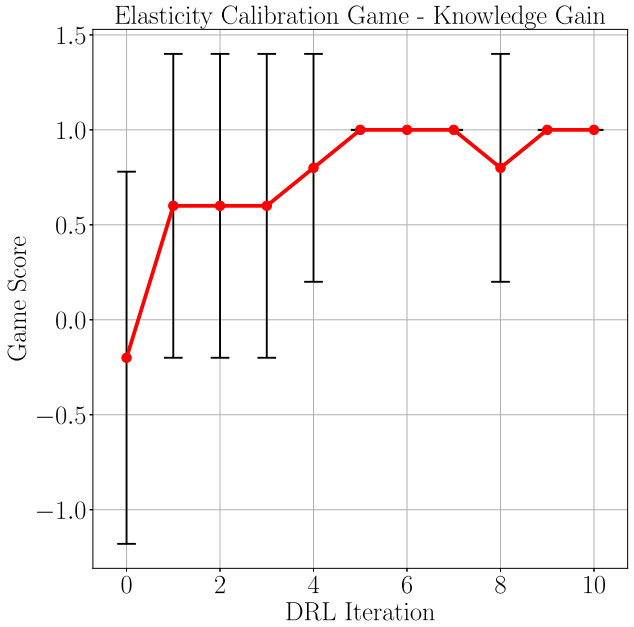

Fig. 7 Convergence over training iterations of a RL training experiment for the linear elastic calibration game with the information gain reward. The error bars indicate the standard deviation of the rewards over all episodes

tree choices more uniformly, and avoid accumulating bias towards one decision tree path.

The policy-value neural network utilized for this example had two hidden dense layers with a width of 50 neurons each and Rectified Linear Unit (ReLU) activation functions. The policy vector output layer had two neurons (equal to the maximum number of allowed actions on the tree) and a softmax activation function. The (scalar) value output layer had one neuron and a tanh activation function. The kernel weight matrix was initialized with a Glorot uniform distribution and the bias vector with a zero distribution for every layer. At the end of every iteration, the model architecture and optimized weights from the previous iteration are reloaded and trained for 100 epochs with a batch sample size of 32 using an Adam optimizer Kingma and Ba [35].

The total CPU time for the offline training of the algorithm for the elasticity decision tree, including playing the game episodes and training the policy network, was about 7 min. In deployment, the neural network can traverse the elasticity tree and design the optimal experiment in about 0.046 s.

The convergence of a training experiment is demonstrated in Fig. 7. The RL reward for the experiments is based on the information gain described in (20). The mean episode reward converges to a maximum value, with essentially zero standard deviation, of zero after 10 training iterations. Note that for this small decision tree, the rewards are scaled to be exactly 0 for the cases where the model is not calibrated successfully and 1 for the cases it is.

Since the RL algorithm applied to this problem is fast to converge and test, we also set up a numerical experiment to test the algorithm's repeatability and the effect the random initialization of the MCTS and neural network has on convergence. Recall that this simple decision tree is composed of two actions; volumetric strain and shear strain were employed to generate the training data. Figure 8 shows the distribution of converged policies $\boldsymbol{p}$ for 100 trials to the expected policy, which should favor paths that take one of each of the independent actions. Upon the first step, there is little to distinguish between the value of taking a shear or compression step, but after the second step, the policy is essentially binary. The final policy assigns the highest value to the shear-compression ([2, 1][2, 0]) and compression-shear ([1, 2][1, 0]) experiments; effectively zero value is assigned to experiments that take repeated steps because they cannot calibrate the linear elastic model. Note that the observed symmetry of tree policies was ensured by selecting a high value for the exploration variable in earlier iterations. For smaller values of the exploration parameter c_{puct}, the algorithm was still observed to converge but randomly biasing towards one winning path over the other.

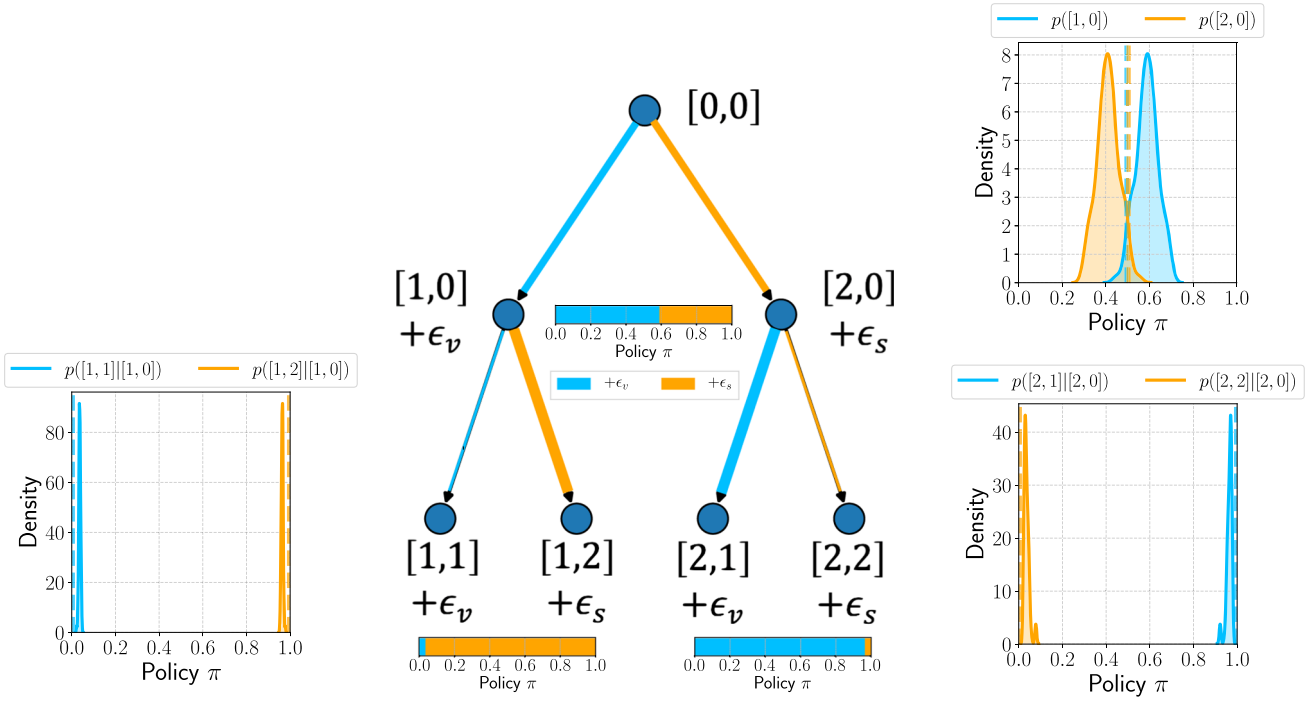

Fig. 8 Linear elastic policy tree (two actions and two steps) for the information gain reward

4.2 Implementation verification 2: experiment for identifying von Mises yield function and hardening

In this example, we introduce an experimental design problem with a history-dependent model and a significantly larger total number of possible paths but also a known optimal design. In this numerical experiment, the EKF-DRL agent is tasked with determining elastic and plastic parameters given the prior knowledge that the yield surface is of von Mises type and isotropic, which is embedded in the chosen model.

First, we illustrate the performance of the EKF and compare the two variants of the switching algorithm.

For both some trial and error in choosing the prior covariances was necessary; however, having an $\mathcal{O}(1)$ response and noiseless observations helped. With these assumptions choosing the observation covariance was simply a matter of numerical conditioning i.e. small but not so small to create underflow.

Figure 9 illustrates the efficacy of the *ad hoc* masked Kalman filter and the more formally rigorous switching Kalman filter (SKF) described in Sect. 2.3. Figure 9a,b show that the two methods both converge on the true parameters (K bulk modulus, G shear modulus, Y yield strength, H hardening modulus) with 100 steps; however, the convergence has different characteristics. With the masked method, prior to yield the elastic parameters (K bulk modulus, G shear modulus) have converged and are fixed throughout the loading steps. Furthermore, the plastic parameters (Y yield strength, H hardening modulus) are effectively fixed prior to encountering yield. The abrupt change also affects the reward shown in Fig. 9c, and in particular, leads to the reward jumping when yield is encountered before leveling off again. On the other hand, the SKF displays much smoother behavior in both the mean parameter convergence (Fig. 9b) and the reward (Fig. 9d). Figure 9d shows the KL divergence for both elastic and plastic modes, but only the reward from the plastic mode is selected as the most probable reference of information gain. Figure 9f demonstrates that the method switches effectively between the two models and chooses the elastic model in the elastic region and the plastic model post yield. It should be noted that the convergence behavior is sensitive to the step size, i.e., a certain number of elastic samples are needed for convergence of the elastic parameters; yield is best detected over a moderate interval, and the estimate of the hardening improves at larger strains. The covariance shown in Fig. 9e drives these changes. The relatively slow convergence of the hardening parameter H for both methods is likely due to a lower sensitivity of the output to this parameter.

Next, we design the action-state space based on physical considerations. Since the von Mises model is pressure-independent, we can constrain the experimental design search to the π-plane, i.e., the plane perpendicular to the pressure and passing through the origin of the three principal stress axes. The underlying elasticity model is still linear

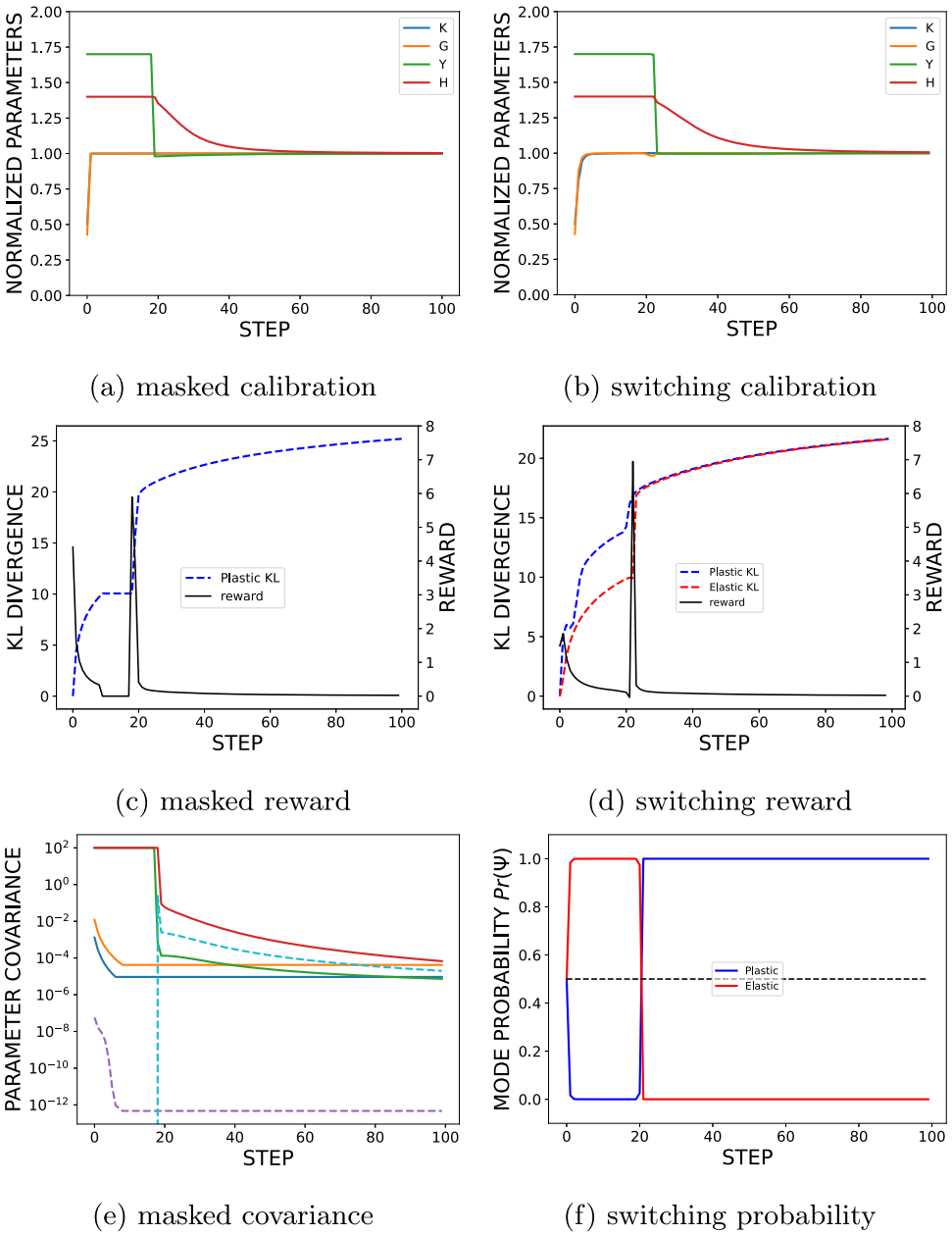

Fig. 9 Comparison of the switching Kalman filter to the masking approach for a von Mises calibration. For **c** and **d**, the dashed Kullback–Leibler divergence lines represent the integral of the reward curve. In **d**, the reward curve corresponds to the Plastic KL only since that is the relevant material model for experimental design

elastic. Thus, we can constrain the exploration of strain and stress on the π-plane by designing strain increment decisions in the two principal directions $\pm\Delta\epsilon_1$ and $\pm\Delta\epsilon_2$ and the increment in the third direction being calculated to have a volumetric strain increment equal to 0.

The decision tree that describes this process is shown in Fig. 10. From every state in the decision tree, there can be four allowed actions: increase $+\Delta\epsilon_1$, increase $+\Delta\epsilon_2$, decrease $-\Delta\epsilon_1$ or decrease $-\Delta\epsilon_2$. The magnitude of the strain increment is chosen to be $\Delta\epsilon = 0.04$ in each direction. The number of options/layers in the decision tree is $n_{\text{opt}} = 6$ which was deemed enough to explore strain–stress cases that exceed the yielding point and demonstrate hardening behavior. The state vector has a length of $n_{\text{opt}} = 6$ as each component corresponds to a selected action. In the root state of the tree, all the components are equal to 0. An enumeration is used for all the actions. For example, selecting the action to increase $+\Delta\epsilon_1$, the component corresponding to this action would be 1, selecting $+\Delta\epsilon_2$ it would be 2, and so on. In Fig. 10, a final state that corresponds to a leaf node in the decision tree is also shown along with the corresponding stress path on the π-plane. The number of all possible configurations/states in the tree is 5461, while the number of final states/experiments is 4096.

Thus, we can define the RL algorithm environment to make experimental decisions (decision tree), generate exper-

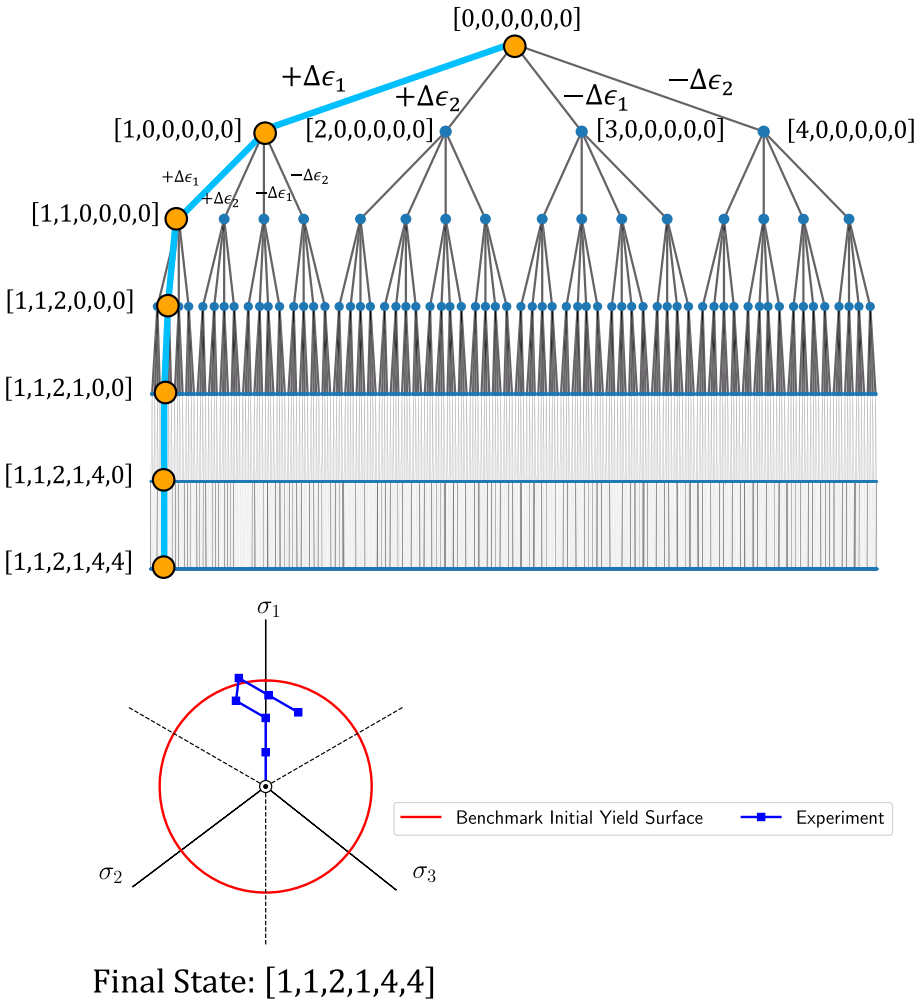

Fig. 10 Decision tree for the exploration of the principal stress space on the π-plane. A complete path from the root note to a leaf node is shown for example along with the corresponding experiment stress path on the π-plane

imental data (a von Mises benchmark model in this synthetic experiment) and calibrate the plasticity model (another von Mises model). In this RL benchmark experiment, we train a RL neural network to explore the π-plane stress space to design experiments that optimize the Kalman filter's discovery of the plastic parameters: yield stress Y and hardening modulus H. Here, the underlying elastic model parameters are considered known ($K = 1.0$, $G = 0.7$). The RL algorithm was performed for 20 training iterations. Each iteration has 10 game episodes. Each game episode involves traversing through the decision tree, root to a leaf node, to design an experiment, collect the training data for this experiment, calibrate the KF model and calculate a reward for this episode. The exploration parameter was again set to a high value ($c_{puct} = 10$) in the first iteration and linearly reduce to 1 in the final iteration to encourage exploration of the decision tree and to avoid converging to a local maximum of the reward.

The neural network architecture used in this experiment is based on the architecture described in Sect. 3.2. Similar to elasticity example, the network inputs a six-component state vector, has two hidden layers (100 neurons each and ReLU activations), and two output layers: the policy output (six neurons and softmax activation) and value output (one neuron and tanh activation). As with the previous example, the kernel weight matrix of every layer was initialized with a Glorot uniform distribution and the bias vector with a zero distribution. The model architecture and optimized weights from the previous iteration are reloaded and trained for 500 epochs at the end of every iteration with a batch sample size of 32 using the Adam optimizer, set with default values.

The total CPU time for the offline training of the algorithm for the isotropic plasticity decision tree was about 20 min and the inference time for a complete tree traversal is about 0.138 s. Note the response time for policy evaluation resulting in a control action is fast but may not be fast enough for high-rate experiments probing rate-effects, in this case we can make control decisions over longer intervals that the data sampling interval.

The RL experiment was performed once with an efficiency index reward based on (23) as described in Sect. 3.4.3 and once with an information gain reward based on (20) as

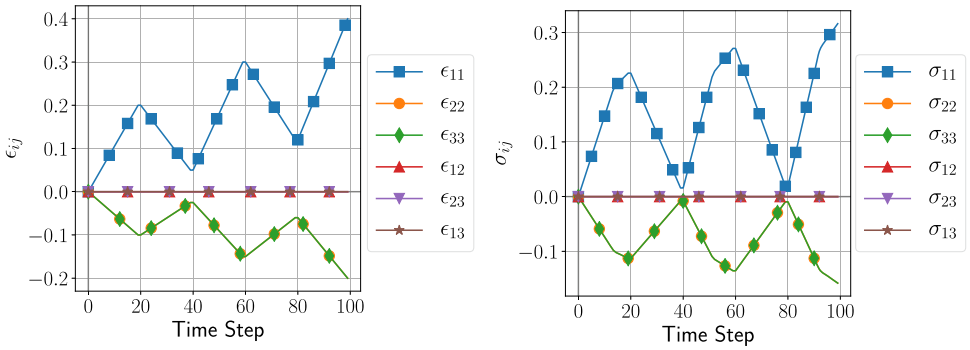

Fig. 11 The strain and stress path of the blind test experiment used to calculate the efficiency index reward for the von Mises dataset

described in Sect. 3.4.1. The efficiency index reward requires an additional sampling of test data points to be calculated for every episode. The additional test curve selected is shown in Fig. 11. We note that the information gain reward does not require additional sampling of the material response space, and it is calculated on the collected experimental data after calibration is complete. The convergence of these two numerical experiments is showcased in Fig. 12, showing the mean episode reward and the episode reward standard deviation for every game iteration. The effect of a higher exploration parameter and the random initialization of the RL neural network in the first game iterations are visible as the strain paths are sampled randomly leading to correspondingly poor mean rewards and high standard deviations. The RL network begins to encourage the sampling of experiments that provide increasingly higher rewards for both the efficiency index and information gain reward before converging to a maximum score at the last few iterations of playing.

The behavior of this convergence is reflected in the optimal experiment predicted by the RL neural network at the end of every game iteration. In Fig. 13, we demonstrate the designed experiments at the end of iterations 1, 10 and 20. Figure 13a and b show the designed experiments for the efficiency index and information gain rewards, respectively. These paths are selected by making a forward prediction with the trained RL neural network at the end of each training iteration. We traverse through the decision tree starting from the root node [0, 0, 0, 0, 0, 0] and selecting the action with the highest policy/probability as predicted by the RL network. By the last iteration, the networks have converged to predict the final state [1, 1, 1, 1, 1, 1] that corresponds to a monotonically increasing strain–stress curve in the direction of the first principal stress axis. This radial loading choice is expected as the yield surface model is isotropic. Furthermore, as previously mentioned, the model sensitivities to moduli, such as H, increase with increased strain, so the radial directions

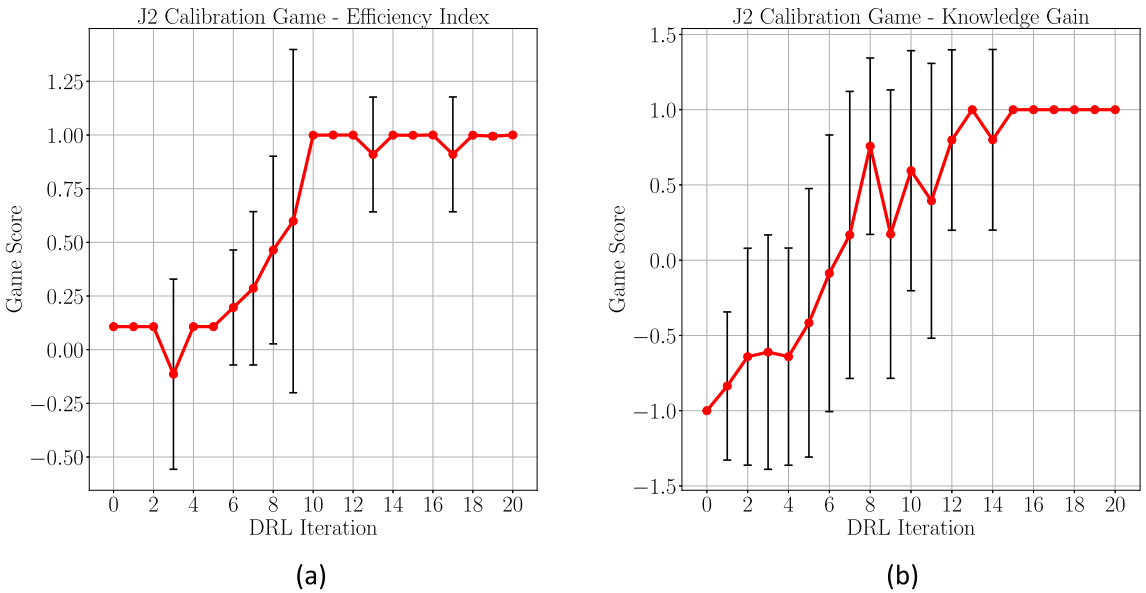

Fig. 12 Game score versus iteration for the von Mises plasticity calibration game calibrating with the Kalman filter model and using **a** an efficiency index reward (NSE) and **b** an information gain reward (KF/KL). The red lines and the error bars are the mean and ±1 standard derivation over the episodes. Note the change in scale

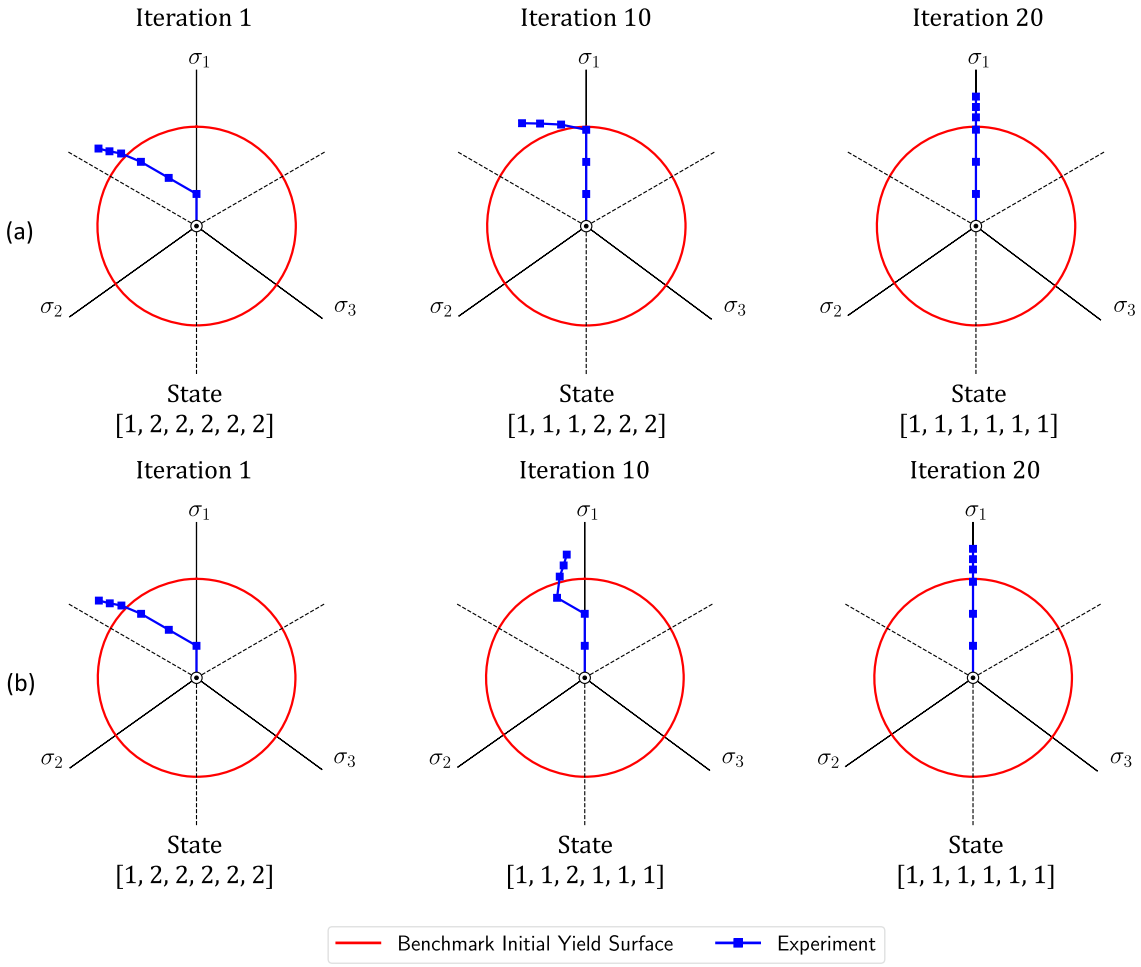

Fig. 13 Designed experiments for the von Mises plasticity calibration game at the end of iterations 1, 10 and 20, calibrating with the Kalman filter model and using **a** an efficiency index reward and **b** an information gain reward

Table 2 The RL-discovered final state and corresponding experimental design for the von Mises calibration game along with the benchmark and KF-calibrated plasticity parameters

Final state	Experimental design
[1, 1, 1, 1, 1, 1]	$+\Delta\epsilon_1, +\Delta\epsilon_1, +\Delta\epsilon_1, +\Delta\epsilon_1, +\Delta\epsilon_1, +\Delta\epsilon_1$
Benchmark parameters	**Calibrated parameters**
$Y_0 = 0.3$	$Y_0 = 0.30261$
$H = 1.0$	$H = 0.9578$

tend to be the most informative. Thus, the KF optimizes the calibration/maximizes the information gain when the data density in one direction is maximized. The converged final states and calibrated KF parameters are shown in Table 2.

4.3 Anisotropic plasticity

In the last numerical experiment, we demonstrate the capacity of the KF model and RL algorithm to calibrate the anisotropic modified Hill model described in "Appendix A". In this numerical experiment, we are setting up the RL environment to calibrate both the elastic parameters (E, ν, $\nu_\perp$) and plastic parameters (B, Y_0, H). Given these complexities the ideal path is not known *a priori*.

For this material model, the π-plane is not enough to adequately describe the anisotropy of the data; thus, we opted for full control of the general strain–stress space. In Fig. 14, the decision tree for the exploration of this parametric space is illustrated. From every state in the decision tree, there are 12 allowable actions to select from. Every choice corresponds to either increasing or decreasing of one of the six symmetric strain tensor components ϵ_{ij}. The number of options in the decision tree is set to be $n_{\text{opt}} = 5$, which is also the length of the state and policy vectors. As a result, the initial state/root node of the decision tree corresponds to the zero vector of [0, 0, 0, 0, 0]. Selecting to increase $+\Delta\epsilon_{33}$ would correspond to state [3, 0, 0, 0, 0], then increasing $+\Delta\epsilon_{11}$ would correspond to state [3, 1, 0, 0, 0], and so on. In Fig. 14, we also

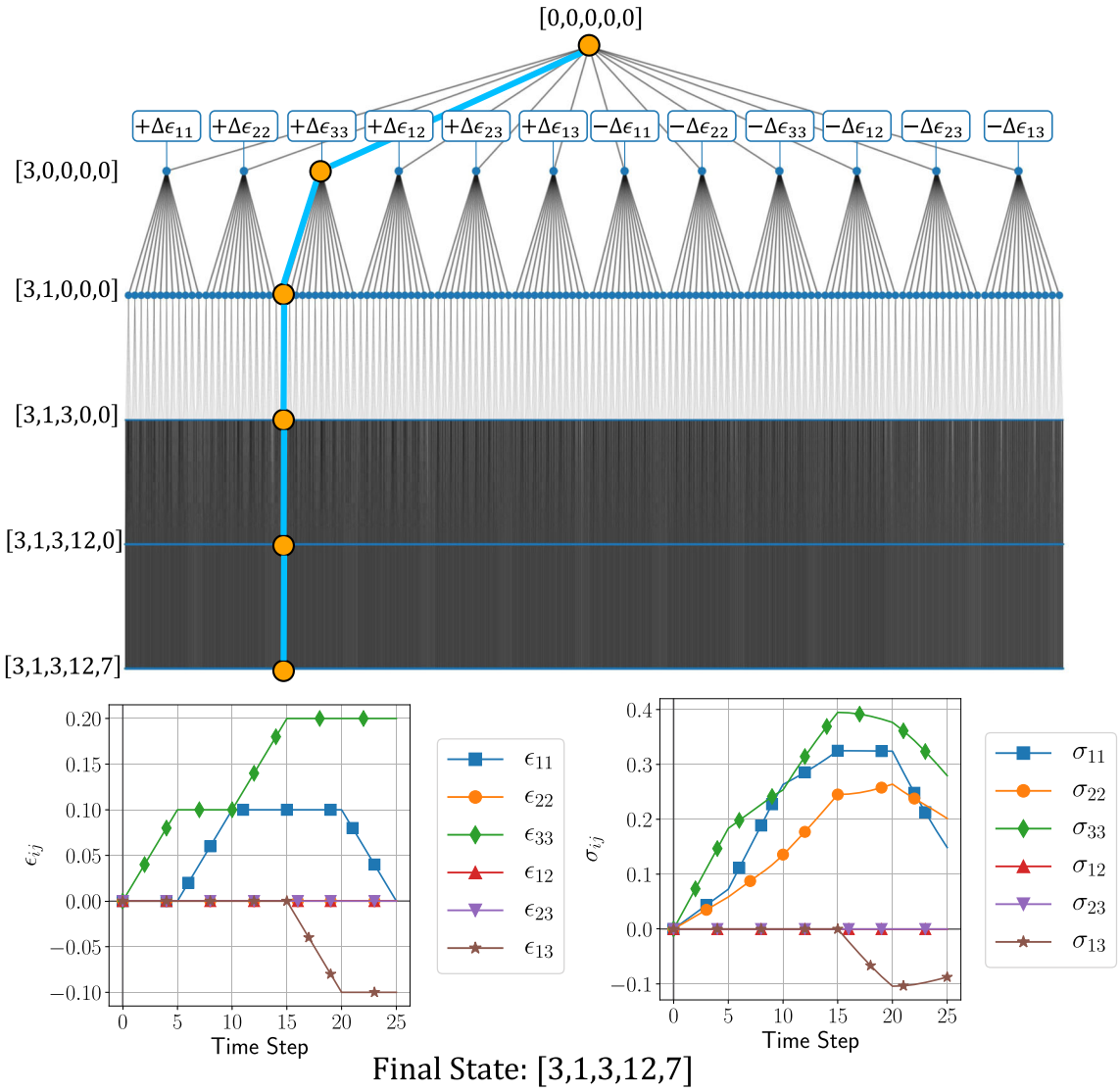

Fig. 14 Decision tree for the exploration of the strain space. A complete path from the root note to a leaf node is shown along with the corresponding experiment strain–stress paths for the Hill model ($B = 0.5$)

illustrate a final leaf state and the corresponding training strain–stress paths that will be used to calibrate the KF model. The number of all possible configurations/states in the tree is 271,453, while the number of final states/experiments is 248,832.

We can thus define the environment to design the experiments (decision tree), gather the experimental data and calibrate an anisotropic elastoplasticity model (KF). In this section, we conduct three numerical experiments to observe the capacity of the network to collect experimental data to fully calibrate the KF model for different anisotropic parameters in the dataset.

For the first two experiments, the RL algorithm is tasked with designing an experiment and calibrating the model for yield surface data with different degrees of anisotropy, $B = 0.5$ and 2. A full description of the parameters of the dataset is shown in Table 3. The algorithm was performed for 30 training iterations. In each iteration, there are 10 game episodes, each involves designing an experiment, calibrating the KF model and calculating the episode reward. In these experiments, the mixed reward is calculated through (24) where the normalized efficiency index and information gain rewards of (23) and (20), respectively, are weighted by $w_{\text{NSE}} = w_{\text{KL}} = 0.5$. The efficiency index is calculated against a blind test experiment strain–stress curve illustrated in Fig. 15. The exploration parameter c_{puct} was tuned to 5 in the first iteration and linearly reduce to 1 in the last iteration. The RL neural network architecture, hyperparameters and training procedures are identical to the ones described in Sec. 4.2. The total CPU time for the offline training of the

Table 3 The RL-discovered final state and corresponding experimental design for the Hill model calibration game along with the benchmark and KF-calibrated parameters for $B = 0.5$ and 2

Modified hill model (B = 0.5)	
Final state	Experimental design
[1, 1, 4, 1, 1]	$+\Delta\epsilon_{11}, +\Delta\epsilon_{11}, +\Delta\epsilon_{12}, +\Delta\epsilon_{11}, +\Delta\epsilon_{11}$
Benchmark parameters	Calibrated parameters
$E = 1.5$	$E = 1.5013$
$\nu = 0.3$	$\nu = 0.2992$
$\nu_\perp = 0.2$	$\nu_\perp = 0.2014$
$B = 0.5$	$B = 0.4996$
$Y_0 = 0.1$	$Y_0 = 0.9999$
$H = 0.1$	$H = 0.1001$
Modified hill model (B = 2.0)	
Final state	Experimental design
[1, 1, 1, 4, 1]	$+\Delta\epsilon_{11}, +\Delta\epsilon_{11}, +\Delta\epsilon_{11}, +\Delta\epsilon_{12}, +\Delta\epsilon_{11}$
Benchmark parameters	Calibrated parameters
$E = 1.5$	$E = 1.4893$
$\nu = 0.3$	$\nu = 0.2984$
$\nu_\perp = 0.2$	$\nu_\perp = 0.2109$
$B = 2.0$	$B = 1.9889$
$Y_0 = 0.15$	$Y_0 = 0.1505$
$H = 0.2$	$H = 0.1999$

anisotropic plasticity decision tree algorithm was about 3 h and the inference time from root to leaf node of the decision tree is about 0.115 s.

The convergence of the two experiments is demonstrated in Fig. 16. The figure demonstrates that the mean values of the mixed reward for the game episodes per iteration are maximized and converged by iteration 30. In an experimental space this large, it is difficult to perfectly scale the rewards to be exactly maximized at unity. Therefore, we empirically tune the scaling so that the rewards are roughly in the range of [0,1]. The RL algorithm still discovers a complex path that leads to a maximum reward, and the calibrated parameters shown in Table 3 are deemed adequate.

In Fig. 17, we present the curve discovered by the RL algorithm for the Hill model dataset with parameter $B = 0.5$. The figure shows the corresponding full strain tensor components ϵ_{ij} time history for the predicted converged final state [1, 1, 3, 1, 1] at iteration 30. It is observed that the KF model calibrates very well to that data with the predicted stress component paths matching the benchmark data. The good approximation of the Hill model plasticity parameters is also observed in predicting the anisotropic yield surface and its evolution with hardening as shown in Fig. 18 (Table 4).

In a third experiment, we investigate the capacity of the DRL algorithm to converge and calibrate the model for different levels of model discrepancy. Specifically, we test two cases where the DRL algorithm is guided by a model that covers a smaller and greater range of behaviors than those existing in the database respectively. For the former, we test a case where the underlying constitutive model is not expressive enough to capture the range of behaviors in the data set. Specifically, we are attempting to calibrate the model with an isotropic von Mises yield surface to anisotropic elastoplastic data from a Hill model with $B = 0.5$. We allow the KF to calibrate five parameters $\boldsymbol{\theta} = \{E, \nu, \nu_\perp, Y_0, H\}$ while $B = 1$ is fixed. For the latter, we test the capacity of the RL algorithm to design an experiment that can reduce the degree of anisotropy of the Hill model to a simpler model. We allow the KF to calibrate all six anisotropic parameters $\boldsymbol{\theta} = \{E, \nu, \nu_\perp, B, Y_0, H\}$ while providing data for an isotropic elastic and von Mises plasticity model. The decision tree, neural network setup, and RL hyperparameters are identical to the previous two experiments. The RL algorithm is run for 20 training iterations and 20 game episodes each for both cases. The mixed reward is utilized in this experiment as well with $w_{\text{NSE}} = w_{\text{KL}} = 0.5$.

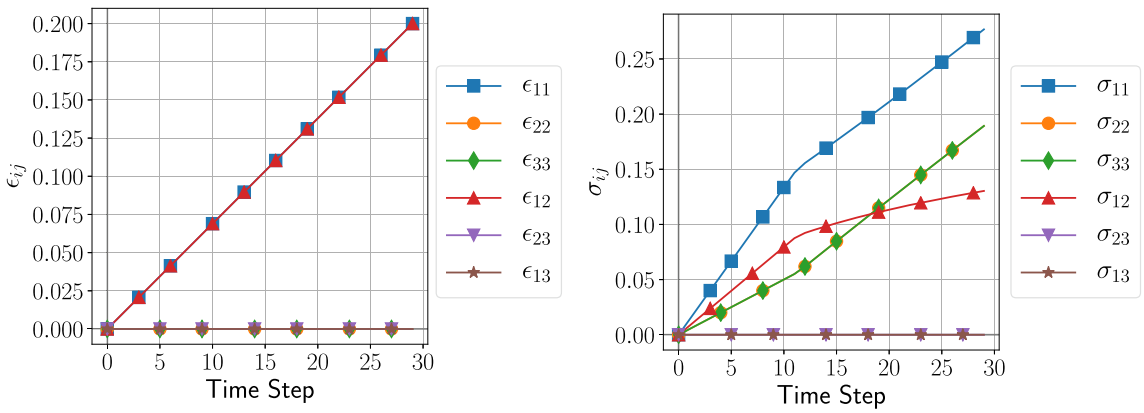

Fig. 15 The strain and stress path of the blind test experiment used to calculate the efficiency index reward for the $B = 0.5$ modified Hill model dataset

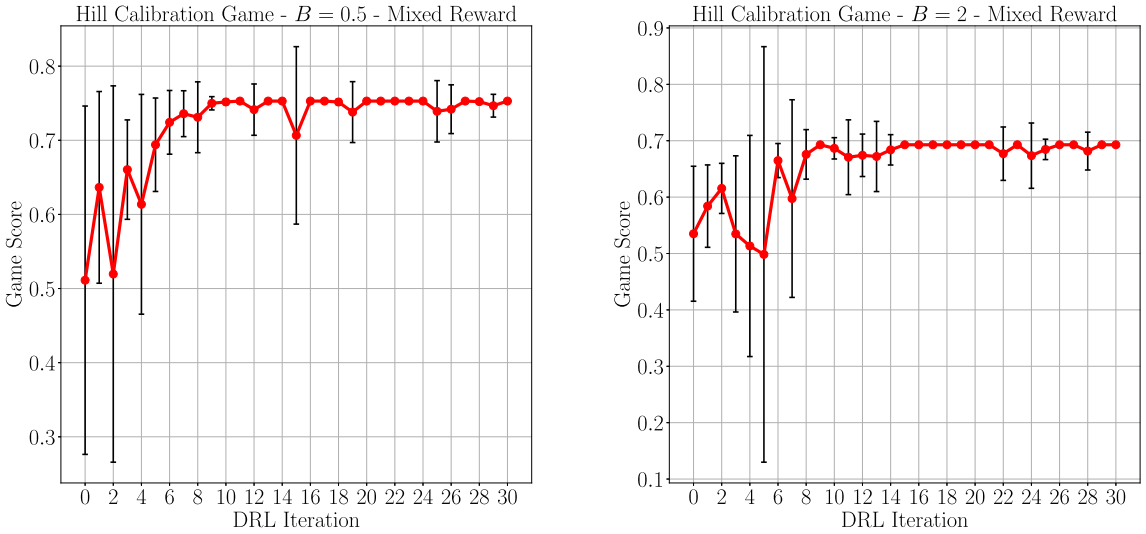

Fig. 16 Game score versus iteration for the Hill plasticity calibration game for $B = 0.5$ and 2 calibrating with the KF model and using a mixed reward combining the efficiency index and information gain reward. The red lines and the error bars are the mean and ± 1 standard derivation of the rewards over the episodes

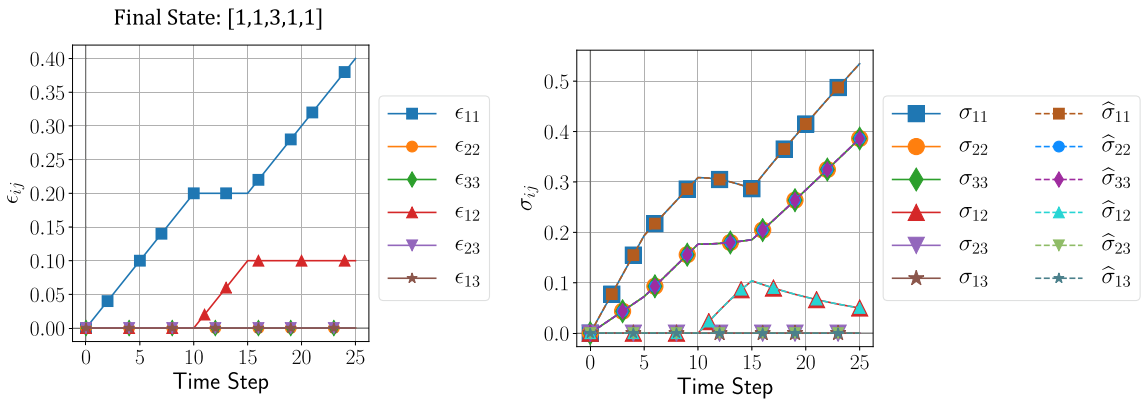

Fig. 17 **a** Discovered experiment strain path for the Hill model dataset with $B = 0.5$ that corresponds to the tree state [1, 1, 3, 1, 1]. **b** The corresponding data stress components σ_{ij} and the KF model prediction $\widehat{\sigma}_{ij}$

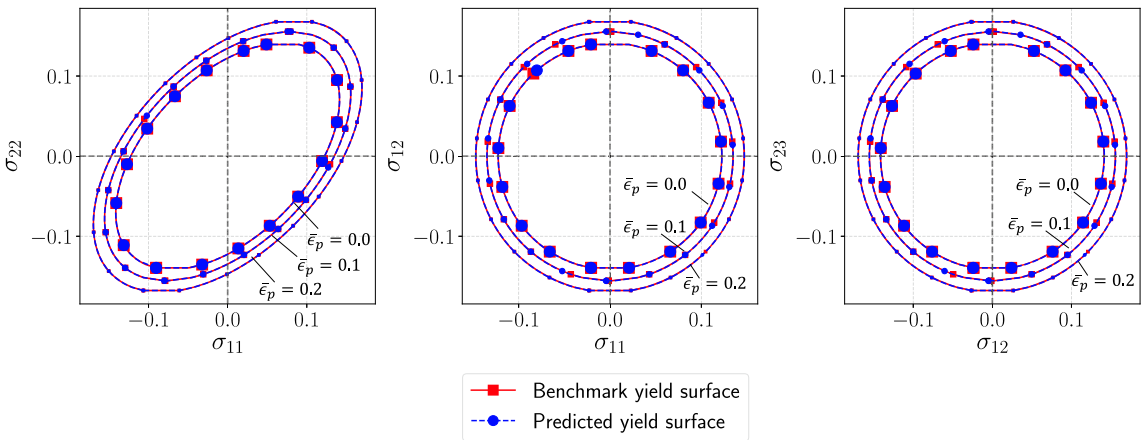

Fig. 18 Predicted yield surface for the Hill model $B = 0.5$ for the KF-calibrated model for accumulated plastic strain $\bar{\epsilon}^p = 0$, 0.1 and 0.2

Table 4 The RL-discovered final state and corresponding experimental design for the isotropic von Mises plasticity model calibration game along with the benchmark and KF-calibrated parameters calibrated on the anisotropic elastoplasticity Hill data ($B = 0.5$)

von Mises Plasticity model	
Final State	Experimental Design
[2, 9, 1, 9, 9]	$+\Delta\epsilon_{22}, -\Delta\epsilon_{33}, +\Delta\epsilon_{11}, -\Delta\epsilon_{33}, -\Delta\epsilon_{33}$
Benchmark parameters	Calibrated parameters
$E = 1.5$	$E = 1.3713$
$\nu_{xy} = 0.3$	$\nu_{xy} = 0.2982$
$\nu_{zy} = 0.2$	$\nu_{zy} = 0.2700$
$B = 0.5$	$B = 1.0$ (fixed)
$Y_0 = 0.1$	$Y_0 = 0.1014$
$H = 0.1$	$H = 0.1011$

The efficiency index for the two cases is calculated against the strain–stress paths shown in Fig. 15 and Fig. 11 respectively.

The convergence of the mixed reward for the two cases is shown in Fig. 19, in which the algorithm reaches a maximum score by the 20th iteration. For the model discrepancy case, the model can be seen to represent the anisotropic behavior with a best fit using the initial size of the isotropic yield surface Y_0 and hardening parameter H. The discrepancy can be observed in Fig. 20 where the isotropic axes of the yield function are perfectly captured (the yielding behavior in the $\sigma_{11} - \sigma_{22}$ axes is independent of B) but in the shear stress axes the anisotropy cannot be captured. For the model simplification case, the discovered parameters of the Kalman model calibration are shown in Table 5 where the two calibrated Poisson ratios are equal, indicating isotropy, and the anisotropy parameter B is close to unity, which coincides with the von Mises yield surface. Thus, we observe that the DRL algorithm has the capacity to converge regardless of model discrepancy to a state that maximizes the calibration objective. Note that, due to the usage of the upper confident bound to balance exploitation and exploration, the game score is not necessarily improving monotonically and may dip locally in between an iteration due to the exploration. Nevertheless, the trend of the game score is improving until the game equilibrium is reached.

5 Conclusion

In this paper, we introduced an integrated framework that combines the strengths of Kalman filters (KF) and model-based deep reinforcement learning (DRL) to design experiments for calibrating material models. The enhanced Kalman filter provides the means to estimate the information gain corresponding to each individual action in an experiment without further sampling. As shown in the numerical examples, this estimated information gain, quantified by the Kullback–Leibler (KL) divergence, can help us improve the efficiency of the DRL by either replacing the more expensive Nash–Sutcliffe efficiency (NSE) index (which can be

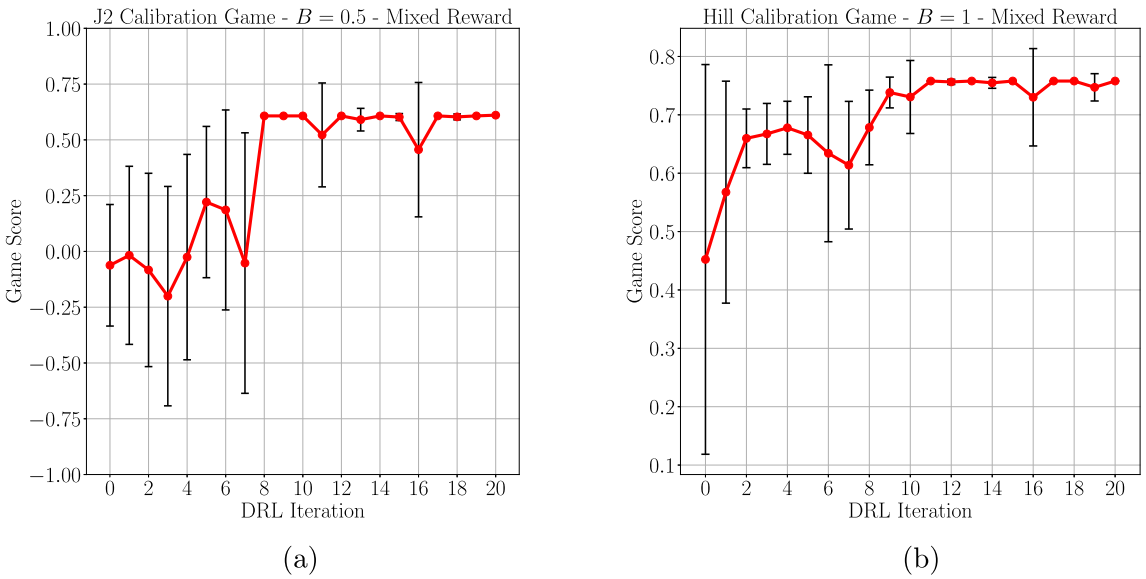

Fig. 19 **a** Game score vs. iteration for the J2 plasticity calibration game model given a dataset for anisotropic elastoplasic behavior from the Hill model ($B = 0.5$). **b** Game score versus iteration for the Hill plasticity calibration game model given a dataset for isotropic elasticity and von Mises yield surface ($B = 1$). The red dots and the error bars are the mean and $\pm$ standard derivation of the rewards of over the episodes respectively

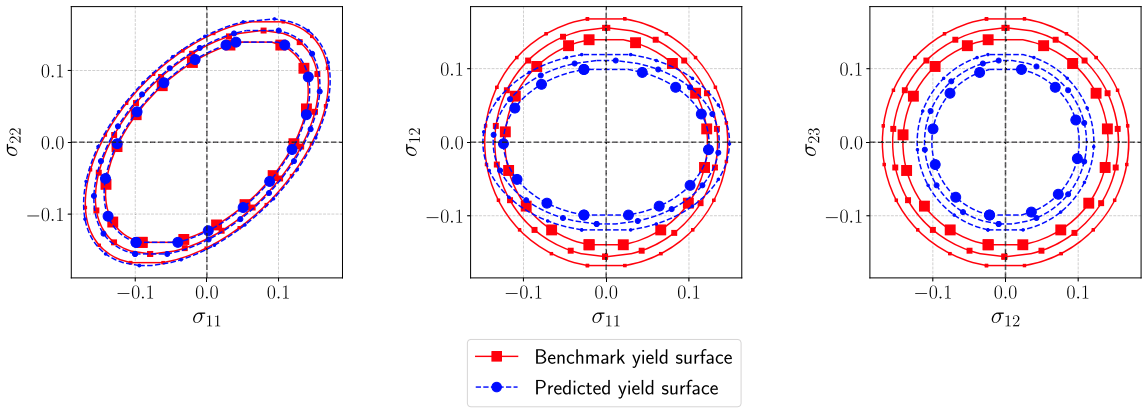

Fig. 20 Predicted yield surface for the KF-calibrated von Mises plasticity model for accumulated plastic strain $\overline{\epsilon}^p = 0, 0.1,$ and 0.2. The model was calibrated through the DRL algorithm in an environment of anisotropic Hill plasticity data ($B = 0.5$)

Table 5 The RL-discovered final state and corresponding experimental design for the Hill model calibration game along with the benchmark and KF-calibrated parameters calibrated on the linear elasticity/von Mises data

Modified hill model (B = 1.0/von Mises)	
Final state	Experimental design
[1, 1, 1, 12, 1]	$+\Delta\epsilon_{11}, +\Delta\epsilon_{11}, +\Delta\epsilon_{11}, -\Delta\epsilon_{13}, +\Delta\epsilon_{11}$
Benchmark parameters	Calibrated parameters
$E = 1.5$	$E = 1.5017$
$\nu = 0.3$	$\nu = 0.2996$
$\nu_\perp = 0.3$	$\nu_\perp = 0.3002$
$B = 1.0$	$B = 0.9992$
$Y_0 = 0.1$	$Y_0 = 0.0999$
$H = 0.1$	$H = 0.1000$

expensive to estimate due to the additional cost for k-fold validation), or enabling us to reduce the sampling size for the NSE reward estimation.

In future work, we will enhance the algorithm to improve its performance. In particular, including a variable step size in the action space or enabling continuous action (cf. Doya [11]) will allow the method to optimize the decisions more precisely near the onset of yield.

We will also pursue the application of the proposed RL methodology to real physical experiments which will inevitably include model-data discrepancy and incomplete observations of material response. A full model of the boundary value problem representing a typical mechanical experiment may enable a richer representation of the state and hence improve the policy predictions at the expense of higher training costs and reaction time. Techniques that can effectively learn policies in state space of higher dimension, such as those in Yang et al. [71] and those that can enable the deployment of multiple agents to explore the decision tree with shared experience, such as Schrittwieser et al. [57], both of which are very active research fields in DRL, will be explored in the future to further improve the robustness and accuracy on the decision knowledge acquired by the experimentalist agent.

Ultimately, the goal is to deploy the method as a real-time decision-making process for concurrent experiment data collection and model calibration. What we have developed is an actor with a policy that is trained to synthetic data that can deployed on an actual experiment and will react to the real time rewards. We rely on similarity between the model used to generate training data and the actual behavior of the material of interest with the assumption that similar behavior is sufficient to generate a good policy. Due to the extensive interactions required to generate good policies, (a) transfer learning between the simulated environmental and physical tests and (b) more effective state representation may both be necessary [9]. Transfer learning would entail retraining the DNN policy starting from the parameters optimized to the synthetic data with limited data from the actual experiment, while keeping as much of the problem the same, e.g., the action set. As mentioned, selecting an ideal state representation is an open question, with downsides to adding too many state variables. Combinatorial studies are brute force means of determining which state representation have advantages over others. Latent encoding method may prove to have applications as well. On a related thrust, the appropriate material model is generally not known in advance, so model selection and/or machine learning/data-driven models to augment the traditional models are likely needed. As we have shown, the SKF is an effective means to perform model selection among a finite set of models.

Real experiments also present the complications of measurement noise, imprecise or incomplete control of the sample, indirect measurements (e.g. observing forces and displacements instead of stresses and strains), multiple mea-

surement methods (strain gauges as well as full-field surface measurements) and potential advantages of multiple protocols on equivalent samples. Each of these challenges will likely engender new developments to the proposed RL algorithm; however, the KF provides means of tackling many of them, such as noise, incomplete control and observation.

Acknowledgements The authors are primarily supported by Sandia National Laboratories Computing and Information Sciences Laboratory Directed Research and Development program, with additional support from the Department of Defense SMART scholarship is provided to support Nhon N. Phan. NAT acknowledges support from the Department of Energy early career research program. This support is gratefully acknowledged. WCS would also like to thank Dr. Christine Anderson-Cook from Los Alamos National Laboratory for a fruitful discussion on the design of experiments in 2019 and the UPS Foundation Visiting Professorship from Stanford University for providing additional funding for this research. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis, which is also archived in the internal Sandia report SAND2022-13022. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. This article has been co-authored by an employee of National Technology and Engineering Solutions of Sandia, LLC under Contract No. DE-NA0003525 with the U.S. Department of Energy (DOE). The employee owns all right, title and interest in and to the article and is solely responsible for its contents. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this article or allow others to do so, for United States Government purposes. The DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan https://www.energy.gov/downloads/doe-public-access-plan.

Data availability The code and data that support the findings of this paper will be posted in an open-source repositories upon the acceptance of the manuscript.

Credit statement The extended Kalman filter was implemented by Dr. Ruben Villareal, whereas the incorporation of the extended Kalman filter and the framework of the deep reinforcement learning was implemented by Dr. Nikolaos Vlassis. The rest of the authors contributed to developing ideas, writing the manuscript and discussions.

Appendix A: Elastoplasticity

One class of particularly technologically important examples of this problem type is the calibration of traditional elastoplasticity models [44, 56, 61]. In these models the observable stress $\boldsymbol{\sigma}$ is a function of elastic strain $\boldsymbol{\epsilon}^e$. Typically, a linear relationship between $\boldsymbol{\sigma}$ and $\boldsymbol{\epsilon}^e$ is assumed:

$$\boldsymbol{\sigma} = \mathbb{C}\boldsymbol{\epsilon}^e, \tag{28}$$

where $\mathbb{C}$ is a fourth-order elastic-modulus tensor. For instance, with transverse isotropy,

$$\begin{aligned}
C_{1111} &= E\frac{(1-\nu_\perp)}{(1-2\nu^2\nu_\perp)},\\
C_{2222} = C_{3333} &= E\frac{(1-\nu^2)}{(1-2\nu^2\nu_\perp)},\\
C_{1122} = C_{1133} &= E\frac{\nu}{(1-2\nu^2\nu_\perp)},\\
C_{2233} &= E\frac{(\nu^2+\nu_\perp)}{(1-2\nu^2\nu_\perp)(1+\nu_\perp)},\\
C_{1212} = C_{1313} &= E\frac{1}{(1-\nu)},\\
C_{2323} &= E\frac{1}{(1-\nu_\perp)},
\end{aligned} \tag{29}$$

where E is an effective Young's modulus, ν is an in-plane Poisson's ratio and $\nu_\perp$ is an out-of-plane Poisson's ratio. History dependence is incorporated via plastic strain $\boldsymbol{\epsilon}^p$, which is a hidden material state variable that elicits dissipative behavior. The elastic strain in (28) is the difference between the controllable, observable total strain $\boldsymbol{\epsilon}$ and the irreversible plastic strain $\boldsymbol{\epsilon}^p$:

$$\boldsymbol{\epsilon}^e = \boldsymbol{\epsilon} - \boldsymbol{\epsilon}^p. \tag{30}$$

A closed, convex yield surface limits the elastic region and demarcates the elastic, reversible behavior in the interior of the surface from the irreversible plastic flow at the limit defined by the surface. For instance, a modified/simplified Hill anisotropic yield surface

$$\begin{aligned}
Y &= \phi(\boldsymbol{\sigma})\\
&\equiv \left(\frac{1}{3}\left((\sigma_{22}-\sigma_{33})^2+(\sigma_{11}-\sigma_{33})^2+(\sigma_{22}-\sigma_{11})^2\right)\right.\\
&\quad \left.+\frac{B}{2}\left(\sigma_{23}^2+\sigma_{13}^2+\sigma_{21}^2\right)\right)^{1/2}
\end{aligned}$$

generalizes the widely-used von Mises yield surface [44]; in fact, Fig. 21 shows that it reduces to von Mises when $B = 1$. The yield surface evolves with hardening of the material

$$Y = Y_0 + h(e^p), \tag{31}$$

where Y_0 is the initial yield strength, h is the hardening function and e^p is the equivalent plastic strain. For instance, $h = He^p$ induces linear hardening. The plastic strain evolves via the (associative) flow rule

$$\dot{\boldsymbol{\epsilon}}^p = \dot{\lambda}\boldsymbol{\partial}_{\boldsymbol{\sigma}}\phi, \tag{32}$$

where the direction of evolution is given by the normal to the yield surface $\boldsymbol{\partial}_{\boldsymbol{\sigma}}\phi$.

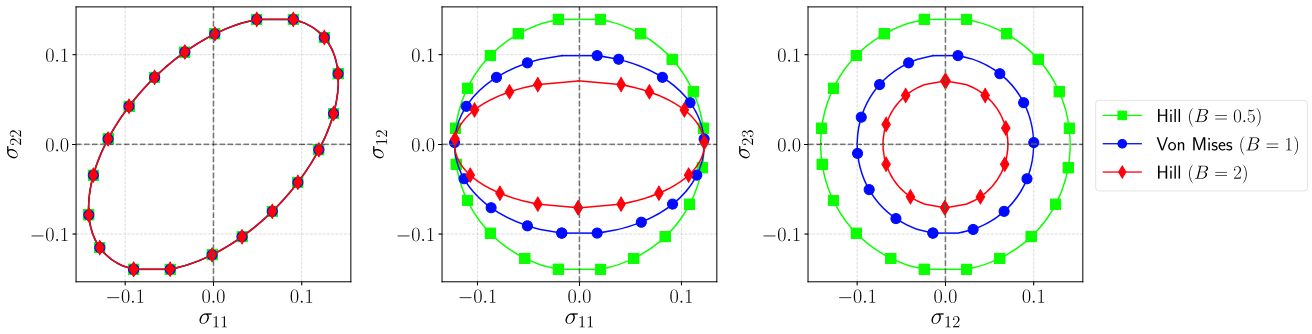

Fig. 21 The modified Hill yield surface model under different stress axes for parameters $B = 0.5$, 1, and 2

The yield surface

$$g \equiv \phi(\boldsymbol{\sigma}) - Y(e^p) \leq 0 \tag{33}$$

constrains the possible response of the material. When $g < 0$, the material is in an elastic state, and the material state variables $\boldsymbol{\epsilon}^p$ and $\lambda = e^p$ are fixed so that the stress at the new state k is

$$\boldsymbol{\sigma}_k = \mathbb{C}(\boldsymbol{\epsilon}_k - \boldsymbol{\epsilon}^p) = \boldsymbol{\sigma}_{k-1} + \mathbb{C}(\Delta\boldsymbol{\epsilon}). \tag{34}$$

where k indexes load steps. Otherwise, the material is in a plastic state; the evolution equations and the constraint $g = 0$ need to be solved through a Newton iteration with increments $\Delta\boldsymbol{\sigma}^{(i)}$

$$\boldsymbol{\sigma}_k = \boldsymbol{\sigma}_{k-1} + \sum_i \Delta\boldsymbol{\sigma}^{(i)}. \tag{35}$$

where i indexes the Newton iterations. This aspect complicates obtaining parameter sensitivities. Further details can be found in Simo and Hughes [61].

For this exemplar the parameters are $\boldsymbol{\theta} = \{E, \nu, \nu_\perp, B, Y_0, H\}$. If $Y_0 \to \infty$ the model reduces to elasticity, and if $\nu_\perp = \nu$ it reduces to isotropic elasticity, $\boldsymbol{\theta} = \{E, \nu\}$. If Y_0 is finite, $\nu_\perp = \nu$ and $B = 1$ it reduces to the widely-used von Mises plasticity model, $\boldsymbol{\theta} = \{E, \nu, Y_0, H\}$.

Appendix B: EKF for state and parameter estimation of DAEs

In this appendix we will derive and discuss the extended Kalman filter (EKF) in the context of semi-explicit index-1 differential algebraic equations (DAEs) for joint state and parameter estimation. The plasticity model described in "Appendix A" is an example of a DAE system with an algebraic stress rule (28) and an ordinary differential equation (ODE) prescribing the flow of the hidden state variables (32) subjected to the algebraic yield constraint (33). In general, these DAEs have the form

$$\dot{\mathbf{z}} = \mathbf{f}\left(\mathbf{z}, \boldsymbol{\sigma}, \boldsymbol{\theta}, \mathbf{x}, t\right) \tag{B.1}$$

$$\mathbf{0} = \mathbf{g}\left(\mathbf{z}, \boldsymbol{\sigma}, \boldsymbol{\theta}, \mathbf{x}, t\right). \tag{B.2}$$

Here $\mathbf{z}$ are unobserved dynamic states, $\boldsymbol{\sigma}$ are unobserved algebraic states, $\boldsymbol{\theta}$ are model parameters, $\mathbf{x}$ are known inputs and t is time. Since we are dealing with index-1 DAEs, we can assume that $g\left(\mathbf{z}, \boldsymbol{\sigma}, \mathbf{x}, t\right) = 0$ is solvable for $\boldsymbol{\sigma}$. In addition to the DAE process model, we assume that there is an observation model for measurement d, given by

$$\mathsf{d} = \mathbf{m}\left(\mathbf{z}, \boldsymbol{\sigma}, \boldsymbol{\theta}, \mathbf{x}, t\right) + \boldsymbol{\varepsilon}, \tag{B.3}$$

where $\boldsymbol{\varepsilon}$ is noise which we assume follows a Gaussian distribution.

Considering that these dynamics are specified using a continuous DAE system, we need to discretize them in time. Further we will assume that the models are not time dependent for simplicity but extending this method to the time-dependent case is straight forward. While there are no closed-form solutions to the DAEs explicitly, for convenience we can define the solution as the function f for the dynamic state update and m for the explicit measurement function when a closed-form solution does not exist (e.g., if it depends on $\boldsymbol{\sigma}$). As a result,

$$\mathbf{z}_k = f\left(\mathbf{z}_{k-1}, \boldsymbol{\theta}, \mathbf{x}_k\right) + \boldsymbol{\eta}_k \tag{B.4}$$

$$\mathsf{d}_k = m\left(\mathbf{z}_k, \boldsymbol{\theta}, \mathbf{x}_k\right) + \boldsymbol{\varepsilon}_k. \tag{B.5}$$

for time t_k. Here the addition of a Gaussian process noise term $\boldsymbol{\eta}$ reflects modeling errors due to the discretization in addition to any intrinsic noise. It is important to note that there are many choices of f depending on the discretization and numerical integration scheme used to solve the DAEs. This explicit construction, though not implementable in a closed form, defines the functions that we need to linearize in order to construct the EKF. We can also augment the state to include fictitious dynamics of the model parameters to aid in model parameter identification:

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \boldsymbol{\delta}_k, \tag{B.6}$$

where $\boldsymbol{\delta}$ is again additive Gaussian noise. For exact parameter estimation, $\boldsymbol{\delta}_k = 0$ because the parameters are fixed; however, in some cases for stability adding small amounts of noise can reduce bias in the estimated parameters at the cost of increased variance and slower convergence of the estimation.

Under this construction, the EKF prediction step has the form

$$\mathbf{z}_{k|k-1} = \boldsymbol{f}\left(\mathbf{z}_{k-1|k-1}, \boldsymbol{\theta}_{k-1|k-1}, \mathbf{x}_k\right) \tag{B.7}$$

$$\boldsymbol{\theta}_{k|k-1} = \boldsymbol{\theta}_{k-1|k-1} \tag{B.8}$$

$$\mathsf{d}_{k|k-1} = m\left(\mathbf{z}_{k|k-1}, \boldsymbol{\theta}_{k|k-1}, \mathbf{x}_k\right), \tag{B.9}$$

while the uncertainty propagation on the prediction has the form

$$\boldsymbol{\Sigma}_{k|k-1} = \mathsf{F}_k \boldsymbol{\Sigma}_{k-1|k-1} \mathsf{F}_k^T + \mathsf{Q}_k \tag{B.10}$$

$$\mathsf{S}_{k|k-1} = \mathsf{A}_k \boldsymbol{\Sigma}_{k|k-1} \mathsf{A}_k^T + \mathsf{R}, \tag{B.11}$$

where $\boldsymbol{\Sigma}$ is the covariance for the joint state $[\mathbf{z}, \boldsymbol{\theta}]^T$, Q is the process and parameter additive uncertainty assumed to be independent and has covariances Q_η and Q_δ, respectively, and R is the measurement noise covariance. We also must construct the linearizations of the dynamics F and the measurement A:

$$\mathsf{Q}_k = \begin{bmatrix} \mathsf{Q}_\eta & \mathbf{0} \\ \mathbf{0} & \mathsf{Q}_\delta \end{bmatrix} \tag{B.12}$$

$$\mathsf{F}_k = \begin{bmatrix} \partial_{\mathbf{z}} \boldsymbol{f} & \partial_{\boldsymbol{\theta}} \boldsymbol{f} \\ \mathbf{I} & \mathbf{0} \end{bmatrix}_{\mathbf{z}_{k|k-1}, \boldsymbol{\theta}_{k|k-1}, \mathbf{x}_k} \tag{B.13}$$

$$\mathsf{A}_k = \begin{bmatrix} \partial_{\mathbf{z}} m & \partial_{\boldsymbol{\theta}} m \end{bmatrix}_{\mathbf{z}_{k-1|k-1}, \boldsymbol{\theta}_{k-1|k-1}, \mathbf{x}_k}. \tag{B.14}$$

Once all these variables have been defined and a measurement d_k has been made, the EKF update is straight forward. The EKF update is given by:

$$\mathsf{r}_k = \mathsf{d}_k - \mathsf{d}_{k|k-1} \tag{B.15}$$

$$\mathsf{K}_k = \boldsymbol{\Sigma}_{k|k-1} \mathsf{A}_k^T \mathsf{S}_{k|k-1}^{-1} \tag{B.16}$$

$$\mathbf{z}_{k|k} = \mathbf{z}_{k|k-1} + \mathsf{K}_k \mathsf{r}_k \tag{B.17}$$

$$\boldsymbol{\Sigma}_{k|k} = \left(\mathbf{I} - \mathsf{K}_k \mathsf{A}_k\right) \boldsymbol{\Sigma}_{k|k-1}. \tag{B.18}$$

Therefore, in order to apply the EKF to the DAEs, we must compute the derivatives: $\partial_{\mathbf{z}} \boldsymbol{f}$, $\partial_{\boldsymbol{\theta}} \boldsymbol{f}$, $\partial_{\mathbf{z}} m$ and $\partial_{\boldsymbol{\theta}} m$.

We will compute these derivatives for the case where the dynamics are implicitly solved using the backward Euler method (as is common for plasticity updates). Thus, returning to the discrete time DAEs, the model becomes

$$\mathbf{z}_k = \mathbf{z}_{k-1} + \Delta_t \mathbf{f}\left(\mathbf{z}_k, \boldsymbol{\sigma}_k, \boldsymbol{\theta}_k, \mathbf{x}_k\right) \tag{B.19}$$

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} \tag{B.20}$$

$$\mathbf{0} = \mathbf{g}\left(\mathbf{z}_k, \boldsymbol{\sigma}_k, \boldsymbol{\theta}_k, \mathbf{x}_k\right) \tag{B.21}$$

$$\mathsf{d}_k = \mathbf{m}\left(\mathbf{z}_k, \boldsymbol{\sigma}_k, \boldsymbol{\theta}_k, \mathbf{z}_k\right). \tag{B.22}$$

Before we begin with the derivation of the derivatives needed for the EKF, we will derive the following useful derivatives: $\partial_{\mathbf{z}_{k-1}} \boldsymbol{\theta}_k$, $\partial_{\boldsymbol{\theta}_{k-1}} \mathbf{z}_{k-1}$, $\partial_{\mathbf{z}_{k-1}} \boldsymbol{\sigma}_k$, $\partial_{\mathbf{z}_k} \boldsymbol{\sigma}_k$ and $\partial_{\boldsymbol{\theta}_{k-1}} \boldsymbol{\sigma}_k$. By inspection, we see that $\partial_{\mathbf{z}_{k-1}} \boldsymbol{\theta}_k = \mathbf{0}$ and $\partial_{\boldsymbol{\theta}_{k-1}} \mathbf{z}_{k-1} = \mathbf{0}$. This realization might seem counter-intuitive because obviously there is a relationship between $\mathbf{z}_{k-1}$ and $\boldsymbol{\theta}_{k-1}$; however, that relationship is already being accounted for via $\boldsymbol{\Sigma}$. To find $\partial_{\mathbf{z}_{k-1}} \boldsymbol{\sigma}_k$ we must implicitly compute the derivative of the algebraic constraint

$$\begin{aligned} \partial_{\mathbf{z}_{k-1}} \mathbf{0} \equiv \mathbf{0} &= \partial_{\mathbf{z}_{k-1}} \mathbf{g}\left(\mathbf{z}_k, \boldsymbol{\sigma}_k, \boldsymbol{\theta}_k, \mathbf{x}_k\right) \\ &= \partial_{\mathbf{z}_k} \mathbf{g}\, \partial_{\mathbf{z}_{k-1}} \mathbf{z}_k + \partial_{\boldsymbol{\theta}_k} \mathbf{g}\, \partial_{\mathbf{z}_{k-1}} \boldsymbol{\theta}_k \\ &\quad + \partial_{\boldsymbol{\sigma}_k} \mathbf{g}\, \partial_{\mathbf{z}_{k-1}} \boldsymbol{\sigma}_k \\ &= \partial_{\mathbf{z}_k} \mathbf{g}\, \partial_{\mathbf{z}_{k-1}} \mathbf{z}_k + \partial_{\boldsymbol{\sigma}_k} \mathbf{g}\, \partial_{\mathbf{z}_{k-1}} \boldsymbol{\sigma}_k \end{aligned} \tag{B.23}$$

$$\implies \partial_{\mathbf{z}_{k-1}} \boldsymbol{\sigma}_k = -\left(\partial_{\boldsymbol{\sigma}_k} \mathbf{g}\right)^{-1} \partial_{\mathbf{z}_k} \mathbf{g}\, \partial_{\mathbf{z}_{k-1}} \mathbf{z}_k. \tag{B.24}$$

Here we used the fact that $\partial_{\mathbf{z}_{k-1}} \boldsymbol{\theta}_k = \mathbf{0}$. Also, we know that $\partial_{\boldsymbol{\sigma}_k} g$ is invertible because it is an index-1 DAE. Following the same argument, we also find that $\partial_{\mathbf{z}_k} \boldsymbol{\sigma}_k = -\left(\partial_{\boldsymbol{\sigma}_k} \mathbf{g}\right)^{-1} \partial_{\mathbf{z}_k} \mathbf{g}$.

Similarly, to find $\partial_{\boldsymbol{\theta}_{k-1}} \boldsymbol{\sigma}_k$ we must implicitly compute the derivative of the algebraic constraint

$$\begin{aligned} \partial_{\boldsymbol{\theta}_{k-1}} \mathbf{0} \equiv \mathbf{0} &= \partial_{\boldsymbol{\theta}_{k-1}} \mathbf{g}\left(\mathbf{z}_k, \boldsymbol{\sigma}_k, \boldsymbol{\theta}_k, \mathbf{x}_k\right) \\ &= \partial_{\mathbf{z}_k} \mathbf{g}\, \partial_{\boldsymbol{\theta}_{k-1}} \mathbf{z}_k + \partial_{\boldsymbol{\theta}_k} \mathbf{g}\, \partial_{\boldsymbol{\theta}_{k-1}} \boldsymbol{\theta}_k \\ &\quad + \partial_{\boldsymbol{\sigma}_k} \mathbf{g}\, \partial_{\boldsymbol{\theta}_{k-1}} \boldsymbol{\sigma}_k \end{aligned} \tag{B.25}$$

$$\implies \partial_{\boldsymbol{\theta}_{k-1}} \boldsymbol{\sigma}_k = -\left(\partial_{\boldsymbol{\sigma}_k} \mathbf{g}\right)^{-1} \left(\partial_{\mathbf{z}_k} \mathbf{g}\, \partial_{\boldsymbol{\theta}_{k-1}} \mathbf{z}_k + \partial_{\boldsymbol{\theta}_k} \mathbf{g}\right). \tag{B.26}$$

Now, using the previous definitions, we can use the same implicit strategy to solve for $\partial_{\mathbf{z}} \boldsymbol{f}$. We can derive it as

$$\begin{aligned} \partial_{\mathbf{z}} \boldsymbol{f} &= \partial_{\mathbf{z}_{k-1}} \mathbf{z}_k \\ &= \partial_{\mathbf{z}_{k-1}} \left(\mathbf{z}_{k-1} + \Delta_t \boldsymbol{f}\left(\mathbf{z}_k, \boldsymbol{\sigma}_k, \boldsymbol{\theta}_k, \mathbf{x}_k\right)\right) \\ &= \mathbf{I} + \Delta_t \left(\partial_{\mathbf{z}_k} \boldsymbol{f}\, \partial_{\mathbf{z}_{k-1}} \mathbf{z}_k + \partial_{\boldsymbol{\theta}_k} \boldsymbol{f}\, \partial_{\mathbf{z}_{k-1}} \boldsymbol{\theta}_k + \partial_{\boldsymbol{\sigma}_k} \boldsymbol{f}\, \partial_{\mathbf{z}_{k-1}} \boldsymbol{\sigma}_k\right) \\ &= \mathbf{I} + \Delta_t \left(\partial_{\mathbf{z}_k} \boldsymbol{f}\, \partial_{\mathbf{z}_{k-1}} \mathbf{z}_k - \partial_{\boldsymbol{\sigma}_k} \boldsymbol{f} \left(\partial_{\boldsymbol{\sigma}_k} \mathbf{g}\right)^{-1} \partial_{\mathbf{z}_k} \mathbf{g}\, \partial_{\mathbf{z}_{k-1}} \mathbf{z}_k\right) \end{aligned} \tag{B.27}$$

$$\begin{aligned} \implies \partial_{\mathbf{z}} \boldsymbol{f} &= \left(\mathbf{I} - \Delta_t \partial_{\mathbf{z}_k} \boldsymbol{f} + \Delta_t \partial_{\boldsymbol{\sigma}_k} \boldsymbol{f} \left(\partial_{\boldsymbol{\sigma}_k} \mathbf{g}\right)^{-1} \partial_{\mathbf{z}_k} \mathbf{g}\right)^{-1}. \end{aligned} \tag{B.28}$$

Using a similar strategy, we can compute $\partial_{\boldsymbol{\theta}} \boldsymbol{f}$:

$$\begin{aligned} \partial_{\theta} \boldsymbol{f} &= \partial_{\boldsymbol{\theta}_{k-1}} \mathbf{z}_k \\ &= \partial_{\boldsymbol{\theta}_{k-1}} \left(\mathbf{z}_{k-1} + \Delta_t \boldsymbol{f}\left(\mathbf{z}_k, \boldsymbol{\sigma}_k, \boldsymbol{\theta}_k, \mathbf{x}_k\right)\right) \\ &= \partial_{\boldsymbol{\theta}_{k-1}} \mathbf{z}_{k-1} + \Delta_t \\ &\quad \left(\partial_{\mathbf{z}_k} \boldsymbol{f}\, \partial_{\boldsymbol{\theta}_{k-1}} \mathbf{z}_k + \partial_{\boldsymbol{\theta}_k} \boldsymbol{f}\, \partial_{\boldsymbol{\theta}_{k-1}} \boldsymbol{\theta}_k + \partial_{\boldsymbol{\sigma}_k} \boldsymbol{f}\, \partial_{\boldsymbol{\theta}_{k-1}} \boldsymbol{\sigma}_k\right) \end{aligned}$$

$$= \Delta_t \left(\partial_{\mathbf{z}_k} \not{f}\, \partial_{\boldsymbol{\theta}_{k-1}} \mathbf{z}_k + \partial_{\boldsymbol{\theta}_k} \not{f} - \partial_{\boldsymbol{\sigma}_k} \not{f} \left(\partial_{\boldsymbol{\sigma}_k} \mathbf{g} \right)^{-1} \left(\partial_{\mathbf{z}_k} \mathbf{g}\, \partial_{\boldsymbol{\theta}_{k-1}} \mathbf{z}_k + \partial_{\boldsymbol{\theta}_k} \mathbf{g} \right) \right) \tag{B.29}$$

$$\implies \partial_{\boldsymbol{\theta}} \not{f} = \left(\mathbf{I} - \Delta_t \partial_{\mathbf{z}_k} \not{f} + \Delta_t \partial_{\boldsymbol{\sigma}_k} \not{f} \left(\partial_{\boldsymbol{\sigma}_k} \mathbf{g} \right)^{-1} \partial_{\mathbf{z}_k} \mathbf{g} \right)^{-1} \left(\Delta_t \partial_{\boldsymbol{\theta}_k} \not{f} - \Delta_t \partial_{\boldsymbol{\sigma}_k} \not{f} \left(\partial_{\boldsymbol{\sigma}_k} \mathbf{g} \right)^{-1} \partial_{\boldsymbol{\theta}_k} \mathbf{g} \right). \tag{B.30}$$

Computing the derivatives for the measurement function is more straight forward because we are now no longer using the implicit integrator, and all the key derivatives have already been defined. We find that

$$\partial_{\mathbf{z}_k} m = \partial_{\mathbf{z}_k} m + \partial_{\boldsymbol{\sigma}_k} m\, \partial_{\mathbf{z}_k} \boldsymbol{\sigma}_k \tag{B.31}$$

$$\partial_{\boldsymbol{\theta}_k} m = \partial_{\mathbf{z}_k} m\, \partial_{\theta} \not{f} + \partial_{\boldsymbol{\theta}_k} m + \partial_{\boldsymbol{\sigma}_k} m\, \partial_{\boldsymbol{\theta}_k} \boldsymbol{\sigma}_k. \tag{B.32}$$

For the special case when $m\left(\mathbf{z}_k, \boldsymbol{\sigma}_k, \boldsymbol{\theta}_k, \mathbf{x}_k\right) = \boldsymbol{\sigma}_k$, as in our models, we can significantly simplify these equations as

$$\partial_{\mathbf{z}_k} m = \partial_{\mathbf{z}_k} \boldsymbol{\sigma}_k = -\left(\partial_{\boldsymbol{\sigma}_k} \mathbf{g} \right)^{-1} \partial_{\mathbf{z}_k} \mathbf{g} \tag{B.33}$$

$$\partial_{\boldsymbol{\theta}_k} m = \partial_{\boldsymbol{\theta}_k} \boldsymbol{\sigma}_k = -\left(\partial_{\boldsymbol{\sigma}_k} \mathbf{g} \right)^{-1} \left(\partial_{\mathbf{z}_k} \mathbf{g}\, \partial_{\theta} \not{f} + \partial_{\boldsymbol{\theta}_k} \mathbf{g} \right). \tag{B.34}$$

Appendix C: GPB algorithm applied to plasticity model calibration

In GPB2, the material response modes occurring at step $k-1$ and k can be assigned discrete switching variables $\alpha, \beta \in \{0, 1\}$ respectively, where 0 represents being in an elastic mode and 1 in a plastic mode. The probability of being in either mode depends on the likelihood of the active mode given the observations, the transition model, and mode probabilities. We partition our exemplar into two modes, $\mathcal{M}^0$ and $\mathcal{M}^1$, which have a corresponding elastic, (34), and plastic, (35), response. This allows the yield criterion (33) to be bypassed and responses from both modes can be simultaneously carried out in order to update the mode probabilities at every step. From the perspective of material science there is a low probability of having any plastic behavior at start of an experiment and this prior knowledge can be incorporated by setting the initial mode probabilities accordingly.

Since we do not know *a priori* when switching occurs, we assign each mode a probability; $\pi(\mathcal{M}_0|\boldsymbol{\theta})$ is the prior probability before we collect any data, and $\pi(\mathcal{M}_k|\mathsf{d}_{1:k};\boldsymbol{\theta})$ is the probability after we collect data, where $\mathsf{d}_{1:k}$ is data observed up to step k and $\boldsymbol{\theta}$ are the model parameters shared between both modes. The mode probability $\pi(\mathcal{M})$ update equation is derived from Bayes rule for the joint probability $\pi(\mathcal{M}^{\alpha}_{k-1}, \mathcal{M}^{\beta}_{k})$ given the data d up to the current step k:

$$\begin{aligned}
&\pi(\mathcal{M}^{\alpha}_{k-1}, \mathcal{M}^{\beta}_{k}|\mathsf{d}_k, \mathsf{d}_{1:k-1}) \\
&\quad \propto \pi(\mathcal{M}^{\alpha}_{k-1}, \mathcal{M}^{\beta}_{k}, \mathsf{d}_k|\mathsf{d}_{1:k-1}) \\
&\quad = \pi(\mathsf{d}_k|\mathcal{M}^{\alpha}_{k-1}, \mathcal{M}^{\beta}_{k}, \mathsf{d}_{1:k-1})\pi(\mathcal{M}^{\alpha}_{k-1}, \mathcal{M}^{\beta}_{k}|\mathsf{d}_{1:k-1}) \\
&\quad = \underbrace{\pi(\mathsf{d}_k|\mathcal{M}^{\alpha}_{k-1}, \mathcal{M}^{\beta}_{k}, \mathsf{d}_{1:k-1})}_{L_k(\alpha,\beta)} \\
&\qquad \times \underbrace{\pi(\mathcal{M}^{\beta}_{k}|\mathcal{M}^{\alpha}_{k-1}, \mathsf{d}_{1:k-1})}_{Z(\alpha,\beta)} \underbrace{\pi(\mathcal{M}^{\alpha}_{k-1}|\mathsf{d}_{1:k-1})}_{W_{k-1|k-1}(\alpha)},
\end{aligned} \tag{C.1}$$

The likelihood function $L_k(\alpha, \beta)$ is a multivariate Gaussian distribution $L = \mathcal{N}(\mathbf{r}, \mathsf{A}\boldsymbol{\Sigma}\mathsf{A}^T + \mathsf{R})$ based on the parameter distributions where $\mathbf{r}$ is the residual error and the covariance is defined in (C.4). The transition matrix $Z(\alpha, \beta)$ contains elements $z_{\alpha\beta}$ which are the probability of transitioning from mode $\mathcal{M}^{\alpha}$ to $\mathcal{M}^{\beta}$. It has the form

$$Z(\alpha, \beta) = \begin{bmatrix} z_{00} & z_{01} \\ z_{10} & z_{11} \end{bmatrix} \tag{C.2}$$

and was initialized with the following values:

$$Z(\alpha, \beta) = \begin{bmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{bmatrix}.$$

The reasoning for these initial values is that on the average we expect most experimental steps to remain sequentially in either the elastic or the plastic region with only a few steps centering around the yield point where switching occurs. The $W_{k|k}(\beta) = \sum_{\alpha} W_{k-1,k|k}(\alpha, \beta)$ is the posterior distribution for $\mathcal{M}$. If the transition is partitioned between modes, the switching becomes soft, and the Kalman filter is effectively a mixture of the two discrete filter modes. Since the material response is either elastic or plastic, we manipulate the transition matrix to be binary. When the material is more likely to deform elastically with a probability of $\pi(\mathcal{M}_k = 0|\mathsf{d}_{1:k}, \boldsymbol{\theta}_k) > 1/2$, the prior parameters $\boldsymbol{\theta}_k$ are updated according to $\mathcal{M}^0$; else, if the material begins to deform plastically with probability $\pi(\mathcal{M}_k = 1|\mathsf{d}_{1:k}, \boldsymbol{\theta}_k) > 1/2$, then the model parameters are updated according to $\mathcal{M}^1$. Treating the calibration as separate modes allows for better estimation of both elastic and plastic parameters considering that new data can be partitioned appropriately by the mode assignment.

The GPB2 algorithm extends the Kalman filter (KF) by incorporating a Markovian jump system that models transitions between discrete behavior modes. At each sequential step $t_{k-1} \rightarrow t_k$, the algorithm generates estimates conditioned on N modes and all possible transitions between them, resulting in N^2 mode-matched KFs. The estimates are then merged using a mixing probability proportional to the likelihood of each KF to obtain the overall estimate of the system state. The interplay between the multiple KFs and the mixing probability is illustrated in Fig. 22, highlighting the algorithm's ability to generate accurate state estimates

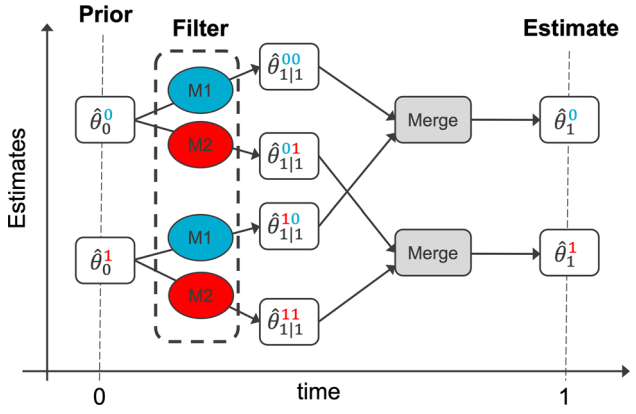

Fig. 22 A schematic of one complete cycle of the GPB2 algorithm

at each time step. The algorithm begins with previous estimates (or priors) W^{α}_{k-1}, $\hat{\theta}^{\alpha}_{k-1}$, and Σ^{α}_{k-1}, which are the mode probabilities, mode conditioned estimates, and covariances, respectively.

A complete recursive cycle of GPB2 is as follows:

1. Mode matched filtering: The N^2 mode matched KFs takes $\mathcal{N}(\hat{\theta}^{\alpha}_{k-1}, \Sigma^{\alpha}_{k-1})$ and outputs $\mathcal{N}(\hat{\theta}^{\beta\alpha}_{k}, \Sigma^{\beta\alpha}_{k})$ where subscript $k|k-1$ denotes predicted statistics and k are updated statistics. Details of the Kalman filter are described in Sec. 2.2. Note that in our exemplars, the material model parameters being estimated are constant and therefore the state transition F_β is the identity matrix. The inclusion of F_β in the following filtering step shows generality for calibrating dynamic states.

$$\hat{\theta}^{\alpha\beta}_{k|k-1} = F_\beta \hat{\theta}^{\alpha}_{k-1}, \tag{C.3}$$

$$\Sigma^{\alpha\beta}_{k|k-1} = F_\beta \Sigma^{\alpha}_{k-1} F^T_\beta, \tag{C.4}$$

$$S^{\alpha\beta}_{k} = A_\beta \Sigma^{\alpha\beta}_{k|k-1} A^T_\beta + R_k, \tag{C.5}$$

$$K^{\alpha\beta}_{k} = \Sigma^{\alpha\beta}_{k|k-1} A^T_\beta (S^{\alpha\beta}_{k})^{-1}, \tag{C.6}$$

$$\hat{\theta}^{\alpha\beta}_{k} = \hat{\theta}^{\alpha\beta}_{k|k-1} + K^{\alpha\beta}_{k}(d_k - A_\beta \hat{\theta}^{\alpha\beta}_{k|k-1}), \tag{C.7}$$

$$\Sigma^{\alpha\beta}_{k} = \Sigma^{\alpha\beta}_{k|k-1} - K^{\alpha\beta}_{k} S^{\alpha\beta}_{k} (K^{\alpha\beta}_{k})^T. \tag{C.8}$$

2. Mixing probabilities: $W^{\alpha\beta}_{k-1|k}$ is interpreted as the probability that mode $\mathcal{M}^\alpha$ was in effect at step k-1 given that $\mathcal{M}^\beta$ is in effect at step k conditioned on data d_k. The likelihood is given by the normal distribution

$$L(\alpha, \beta) = \mathcal{N}(\tilde{r}^{\alpha\beta}_{k}; 0, S^{\alpha\beta}_{k}) \tag{C.9}$$

where $\tilde{r}^{\alpha\beta}_{k} = d_k - A_\beta \hat{\theta}^{\alpha\beta}_{k|k-1}$ is the residual between the prediction and the data d_k and the mixing probabilities are calculated as

$$W^{\alpha\beta}_{k-1|k} = \frac{L(\alpha\beta) Z(\alpha, \beta) W^{\alpha}_{k-1}}{c^{\beta}_{k}} \tag{C.10}$$

where $Z(\alpha, \beta)$ is the transition matrix, W^{α}_{k-1} is a mode probability, and c^{β}_{k} is a normalization constant given by

$$c^{\beta}_{k} \equiv \sum_{\alpha=0}^{N} L(\alpha\beta) Z(\alpha, \beta) Z(\alpha, \beta) W^{\alpha}_{k-1}. \tag{C.11}$$

3. Merging: The previous mode history $\mathcal{M}^\alpha$ of $\hat{\theta}^{\alpha\beta}_{k}$ and $\Sigma^{\alpha\beta}_{k}$ is marginalized out using the mixing probabilities to obtain the conditional posterior estimates and covariances given the current mode $\mathcal{M}^\beta$. These are calculated as

$$\hat{\theta}^{\beta}_{k} = \sum_{\alpha=0}^{N} W^{\alpha\beta}_{k|k-1} \hat{\theta}^{\alpha\beta}_{k} \tag{C.12}$$

$$\Sigma^{\beta}_{k} = \sum_{\alpha=0}^{N} W^{\alpha\beta}_{k|k-1} [\Sigma^{\alpha\beta}_{k} + (\hat{\theta}^{\beta}_{k} - \hat{\theta}^{\alpha\beta}_{k}) \times (\hat{\theta}^{\beta}_{k} - \hat{\theta}^{\alpha\beta}_{k})^T]. \tag{C.13}$$

4. Update mode probabilities: The mode probabilities are updated by the sum of weighted likelihood estimates and are given by

$$W^{\beta}_{k} = \frac{1}{c} \sum_{\alpha=0}^{N} \mathcal{N}(\tilde{r}^{\alpha\beta}_{k}; 0, S^{\alpha\beta}_{k}) Z(\alpha, \beta) W^{\alpha}_{k-1} = \frac{c^{\beta}_{k}}{c} \tag{C.14}$$

where $c \equiv \sum_{\beta=0}^{N} c^{\beta}$.

5. Overall estimate: Combining the mode estimates by the updated mode probabilities (C.14) results in the final state estimate and covariance. These are calculated as

$$\hat{\theta}_{k} = \sum_{\beta=0}^{N} W^{\beta}_{k} \hat{\theta}^{\beta}_{k} \tag{C.15}$$

$$\Sigma_{k} = \sum_{\beta=0}^{N} W^{\beta}_{k} [\Sigma^{\beta}_{k} + (\hat{\theta}_{k} - \hat{\theta}^{\beta}_{k}) \times (\hat{\theta}_{k} - \hat{\theta}^{\beta}_{k})^T]. \tag{C.16}$$

References

1. Ames NM, Srivastava V, Chester SA, Anand L (2009) A thermo-mechanically coupled theory for large deformations of amorphous polymers. Part II: applications. Int J Plast 25(8):1495–1539
2. Baird L (1995) Residual algorithms: reinforcement learning with function approximation. In: Machine learning proceedings 1995. Elsevier, pp 30–37
3. Bower AF (2009) Applied mechanics of solids. CRC Press, Boca Raton
4. Catanach TA (2017) Computational methods for Bayesian inference in complex systems. Ph.D. Thesis, California Institute of Technology

5. Chaloner K, Verdinelli I (1995) Bayesian experimental design: a review. Stat Sci pp 273–304
6. Chatzi EN, Smyth AW (2009) The unscented kalman filter and particle filter methods for nonlinear structural system identification with non-collocated heterogeneous sensing. Struct Control Health Monit 16(1):99–123
7. Darema F (2004) Dynamic data driven applications systems: a new paradigm for application simulations and measurements. In: Computational science-ICCS 2004: 4th international conference, Kraków, Poland, June 6–9, 2004, Proceedings, Part III 4. Springer, pp 662–669
8. Daum F (2005) Nonlinear filters: beyond the Kalman filter. IEEE Aerosp Electron Syst Mag 20(8):57–69
9. De Bruin T, Kober J, Tuyls K, Babuška R (2018) Integrating state representation learning into deep reinforcement learning. IEEE Robot Autom Lett 3(3):1394–1401
10. Ding Z, Huang Y, Yuan H, Dong H (2020) Introduction to reinforcement learning. In: Deep reinforcement learning: fundamentals, research and applications, pp 47–123
11. Doya K (2000) Reinforcement learning in continuous time and space. Neural Comput 12(1):219–245
12. Erazo K, Sen D, Nagarajaiah S, Sun L (2019) Vibration-based structural health monitoring under changing environmental conditions using Kalman filtering. Mech Syst Signal Process 117:1–15
13. Evensen G (2003) The ensemble Kalman filter: theoretical formulation and practical implementation. Ocean Dyn 53(4):343–367
14. Feinberg V, Wan A, Stoica I, Jordan MI, Gonzalez JE, Levine S (2018) Model-based value estimation for efficient model-free reinforcement learning. arXiv:1803.00101
15. Fisher RA et al (1937) The design of experiments. Oliver & Boyd, Edinburgh
16. Fuchs A, Heider Y, Wang K, Sun WC, Kaliske M (2021) DNN2: A hyper-parameter reinforcement learning game for self-design of neural network based elasto-plastic constitutive descriptions. Comput Struct 249:106505
17. Ghanem R, Ferro G (2006) Health monitoring for strongly nonlinear systems using the ensemble Kalman filter. Struct Control Health Monit 13(1):245–259
18. Gnecco G, Sanguineti M et al (2008) Approximation error bounds via Rademacher complexity. Appl Math Sci 2:153–176
19. Gu S, Lillicrap T, Sutskever I, Levine S (2016) Continuous deep q-learning with model-based acceleration. In: International conference on machine learning. PMLR, pp 2829–2838
20. Gu S, Holly E, Lillicrap T, Levine S (2017) Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: 2017 IEEE international conference on robotics and automation (ICRA). IEEE, pp 3389–3396
21. Heider Y, Wang K, Sun WC (2020) So (3)-invariance of informed-graph-based deep neural network for anisotropic elastoplastic materials. Comput Methods Appl Mech Eng 363:112875
22. Heider Y, Suh HS, Sun WC (2021) An offline multi-scale unsaturated poromechanics model enabled by self-designed/self-improved neural networks. Int J Numer Anal Methods Geomech 45(9):1212–1237
23. Hester T, Stone P (2013) Texplore: real-time sample-efficient reinforcement learning for robots. Mach Learn 90:385–429
24. Huan X, Marzouk YM (2013) Simulation-based optimal Bayesian experimental design for nonlinear systems. J Comput Phys 232(1):288–317
25. Huan X, Marzouk YM (2016) Sequential Bayesian optimal experimental design via approximate dynamic programming. arXiv:1604.08320
26. Huang J, Li D, Li H, Song G, Liang Y (2018) Damage identification of a large cable-stayed bridge with novel cointegrated Kalman filter method under changing environments. Struct Control Health Monit 25(5):e2152
27. Huang Y, Jianqi Yu, Beck JL, Zhu H, Li H (2020) Novel sparseness-inducing dual Kalman filter and its application to tracking time-varying spatially-sparse structural stiffness changes and inputs. Comput Methods Appl Mech Eng 372:113411
28. Jazwinski AH (2007) Stochastic processes and filtering theory. Courier Corporation, North Chelmsford
29. Jin C, Jang S, Sun X, Li J, Christenson R (2016) Damage detection of a highway bridge under severe temperature changes using extended Kalman filter trained neural network. J Civ Struct Heal Monit 6(3):545–560
30. Jones RE, Frankel AL, Johnson KL (2022) A neural ordinary differential equation framework for modeling inelastic stress response via internal state variables. J Mach Learn Model Comput 3(3)
31. Julier SJ, Uhlmann JK (1997) New extension of the Kalman filter to nonlinear systems. In: Signal processing, sensor fusion, and target recognition VI, volume 3068. SPIE, pp 182–193
32. Julier SJ, Uhlmann JK (2004) Unscented filtering and nonlinear estimation. Proc IEEE 92(3):401–422
33. Kaelbling LP, Littman ML, Moore AW (1996) Reinforcement learning: a survey. J Artif Intell Res 4:237–285
34. Kalman RE (1960) A new approach to linear filtering and prediction problems. J Basic Eng 82(1):35–45
35. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv:1412.6980
36. Kiumarsi B, Vamvoudakis KG, Modares H, Lewis FL (2017) Optimal and autonomous control using reinforcement learning: a survey. IEEE Trans Neural Netw Learn Syst 29(6):2042–2062
37. Kober J, Andrew Bagnell J, Peters J (2013) Reinforcement learning in robotics: a survey. Int J Robot Res 32(11):1238–1274
38. Kuss M, Rasmussen C (2003) Gaussian processes in reinforcement learning. Adv Neural Inf Process Syst 16
39. Landajuela M, Petersen BK, Kim S, Santiago CP, Glatt R, Mundhenk N, Pettit JF, Faissol D (2021) Discovering symbolic policies with deep reinforcement learning. In: International conference on machine learning. PMLR, pp 5979–5989
40. LaViola JJ (2003) A comparison of unscented and extended Kalman filtering for estimating quaternion motion. In: Proceedings of the 2003 American control conference, 2003, volume 3. IEEE, pp 2435–2440
41. Lee JH, Lawrence Ricker N (1994) Extended Kalman filter based nonlinear model predictive control. Ind Eng Chem Res 33(6):1530–1541
42. Lee S-H, Song J (2020) Regularization-based dual adaptive Kalman filter for identification of sudden structural damage using sparse measurements. Appl Sci 10(3)
43. Li Y (2017) Deep reinforcement learning: an overview. arXiv:1701.07274
44. Lubliner J (2008) Plasticity theory. Courier Corporation, North Chelmsford
45. Ma R, Sun WC (2020) Computational thermomechanics for crystalline rock. Part II: chemo-damage-plasticity and healing in strongly anisotropic polycrystals. Comput Methods Appl Mech Eng 369:113184
46. McCuen RH, Knight Z, Gillian Cutter A (2006) Evaluation of the Nash–Sutcliffe efficiency index. J Hydrol Eng 11(6):597–602
47. Moskovitz T, Parker-Holder J, Pacchiano A, Arbel M, Jordan M (2021) Tactical optimism and pessimism for deep reinforcement learning. Adv Neural Inf Process Syst 34:12849–12863
48. Murphy KP (1998) Switching Kalman filters. Technical report, DEC/Compaq Cambridge Research Labs
49. Nguyen LH, Goulet JA (2018) Anomaly detection with the switching Kalman filter for structural health monitoring. Struct Control Health Monit 25(4):e2136
50. Niv Y (2009) Reinforcement learning in the brain. J Math Psychol 53(3):139–154

51. O'Donoghue B, Osband I, Munos R, Mnih V (2018) The uncertainty bellman equation and exploration. In: International conference on machine learning, pp 3836–3845
52. Ormoneit D, Sen A (2002) Kernel-based reinforcement learning. Mach Learn 49(2–3):161
53. Pukelsheim F (2006) Optimal design of experiments. SIAM, Philadelphia
54. Reda D, Tao T, van de Panne M (2020) Learning to locomote: understanding how environment design matters for deep reinforcement learning. In: Motion, interaction and games. ACM, pp 1–10
55. Ryan EG, Drovandi CC, McGree JM, Pettitt AN (2016) A review of modern computational algorithms for Bayesian optimal design. Int Stat Rev 84(1):128–154
56. Scherzinger WM (2017) A return mapping algorithm for isotropic and anisotropic plasticity models using a line search method. Comput Methods Appl Mech Eng 317:526–553
57. Schrittwieser J, Hubert T, Mandhane A, Barekatain M, Antonoglou I, Silver D (2021) Online and offline reinforcement learning by planning with a learned model. Adv Neural Inf Process Syst 34:27580–27591
58. Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, Lanctot M, Sifre L, Kumaran D, Graepel T, et al (2017a) Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv:1712.01815
59. Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, Lanctot M, Sifre L, Kumaran D, Graepel T, et al (2017b) Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv:1712.01815
60. Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A et al (2017) Mastering the game of go without human knowledge. Nature 550(7676):354
61. Simo JC, Hughes TJR (2006) Computational inelasticity, vol 7. Springer Science & Business Media, Berlin
62. Sun N-Z, Sun A (2015) Model calibration and parameter estimation: for environmental and water resource systems. Springer, Berlin
63. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction. MIT Press, Cambridge
64. Vlassis NN, Sun W (2022) Component-based machine learning paradigm for discovering rate-dependent and pressure-sensitive level-set plasticity models. J Appl Mech 89(2)
65. Wang K, Sun WC (2019) Meta-modeling game for deriving theory-consistent, microstructure-based traction-separation laws via deep reinforcement learning. Comput Methods Appl Mech Eng 346:216–241
66. Wang Kun, Sun WaiChing, Du Qiang (2019) A cooperative game for automated learning of elasto-plasticity knowledge graphs and models with AI-guided experimentation. Comput Mech 1–33
67. Wang K, Sun WC, Qiang D (2021) A non-cooperative meta-modeling game for automated third-party calibrating, validating and falsifying constitutive laws with parallelized adversarial attacks. Comput Methods Appl Mech Eng 373:113514
68. West DB et al (2001) Introduction to graph theory, vol 2. Prentice Hall, Upper Saddle River
69. Williams RJ (1992) Training recurrent networks using the extended Kalman filter. In: [Proceedings 1992] IJCNN international joint conference on neural networks, volume 4. IEEE, pp 241–246
70. Yang JN, Lin S, Huang H, Zhou L (2006) An adaptive extended Kalman filter for structural damage identification. Struct Control Health Monit 13(4):849–867
71. Yang Z, Jin C, Wang Z, Wang M, Jordan MI (2020) On function approximation in reinforcement learning: optimism in the face of large state spaces. arXiv:2011.04622
72. Zhao W, Queralta JP, Westerlund T (2020) Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In: 2020 IEEE symposium series on computational intelligence (SSCI). IEEE, pp 737–744
73. Zhou L, Shinya W, Yang JN (2008) Experimental study of an adaptive extended Kalman filter for structural damage identification. J Infrastruct Syst 14(1):42–51