A computationally efficient framework for the propagation of interval-process uncertainty

Rongyao Song, Changcong Zhou *
Department of Engineering Mechanics, Northwestern Polytechnical University, Xi'an 710129, China
E-mail: changcongzhou@nwpu.edu.cn

Abstract: Computing the lower and upper boundaries of a response interval process is essential for interval uncertainty propagation over a continuous evolution variable, but conventional discretization-optimization schemes require repeated inner extremum searches at many state points and are costly for complex models. This paper develops a dual-layer Bi-Extremum Alternating Surrogate (BEAS) framework. In the inner layer, the expected-improvement criteria for minimization and maximization are coupled within a shared Kriging-based Efficient Global Optimization framework, so the minimum and maximum responses at a fixed state point are found using one surrogate model and one training set. In the outer layer, an alternating strategy updates the two boundary surrogates in half-steps: the by-product extremum of every inner call is delivered to the opposite boundary and, subject to a distance-based screening rule, injected into its training set, with a buffer as a zero-cost fallback, so both extrema of each expensive call are exploited. On two analytical examples, BEAS reproduces the boundaries of conventional double-loop calculations while markedly reducing inner-layer calls and performance-function evaluations.
Keywords: Interval process, Bi-extremum optimization, boundary surrogate, alternating surrogate.

1. Introduction

In engineering systems, input parameters are often uncertain because of manufacturing errors and material variations (Wei et al., 2017). When data are insufficient to fix an exact probability distribution, interval models give a non-probabilistic description that needs only the parameter bounds, with no assumption on distribution type or statistical moments (Faes et al., 2017; Guo and Lu, 2015). They therefore suit cases such as manufacturing and assembly tolerances in moving mechanisms (Tang et al., 2021; Wu and Rao, 2007) and constraints in optimization design (Wang et al., 2021; Niu et al., 2024), and have become an effective tool for cognitive uncertainty.

To describe how the output evolves as one input varies continuously within its range, Chang et al. (2022) introduced the interval process: the other inputs are fixed as interval variables while the target input varies continuously, so the output becomes an evolving interval band rather than a static interval. Song et al. (2024) extended this to time-varying systems. In such a response interval process the upper and lower boundaries are the maximum and minimum of all possible outputs at each state point, and computing them efficiently remains an open problem.

The discretization-optimization method is the general framework for interval-process boundaries: the evolution domain is discretized into state points and a global optimizer is run at each point to obtain the minimum and maximum outputs. For finite-element and other expensive models the number of simulations becomes prohibitive, which motivates surrogate modeling. Kriging is widely used because it provides both a predictive mean and a variance (Kleijnen, 2009). Kriging-based efficient global optimization (EGO), with improved learning functions, has been applied successfully to expensive black-box minimization (Shahriari et al., 2016; Mockus, 2012; Forrester and Keane, 2009); at each iteration EGO adds the point of maximum expected improvement (EI), reaching the optimum with few evaluations.

The standard EGO still struggles on complex problems, and several improvements exist. Mohammadi et al. (2016) used a small-scale Kriging ensemble to adapt the sampling scale for parallel expensive optimization. Liu et al. (2023) modified the EI criterion with a balancing factor to escape local optima on multimodal objectives. For high-dimensional problems, KT-EGO (Wang et al., 2023) extends EGO beyond twenty dimensions through knowledge transfer, reducing calls to expensive simulations. These works broaden the applicability and efficiency of EGO in engineering.

Progress has also been made on interval uncertainty using surrogates and adaptive sampling. Yao et al. (2024) proposed an interval finite element method based on a bilevel Kriging model for structures with spatially uncertain parameters. Ni et al. (2020) combined the interval process with Kriging and a variance-

based adaptive sampling scheme, but updated the two boundaries independently using only prediction variance, ignoring the nature of extremum solving. Zhan and Xiao (2025) introduced an active-learning surrogate for time- and space-dependent reliability. These works mainly target failure probability rather than response boundaries. Two limitations thus remain: at the inner layer the two extrema are found by separate optimizations that duplicate expensive evaluations, and at the outer layer the two boundary surrogates are built independently, so extremum information for one boundary is not reused by the other.

To address these issues, this paper develops a dual-layer Bi-Extremum Alternating Surrogate (BEAS) method. The first contribution is an inner-layer bi-extremum strategy: within a Kriging-based EGO framework, the conventional EI criterion for minimization is paired with a directly formulated EI criterion for maximization, so both extrema at a state point are found from one surrogate and one training set. The second is an outer-layer alternating strategy: each iteration has two half-steps, one per boundary, in which state points are chosen by predictive variance and an inner call is spent only after absorbing the free by-product from the opposite half-step. Each by-product is injected into the opposite training set under a distance-based screening rule that protects the Kriging conditioning, with screened-out values buffered for later zero-cost retrieval. The two surrogates are refined alternately until both meet a normalized convergence criterion.

The remainder of this paper is organized as follows. Section 2 defines the response interval process and reviews existing boundary-evaluation frameworks. Section 3 formulates the inner-layer bi-extremum optimization strategy at a fixed state point. Section 4 develops the alternating boundary surrogate framework and assembles the complete BEAS method. Section 5 presents numerical examples and discussion, and Section 6 concludes the paper.

2. Response interval process and existing computational frameworks

This section defines the response interval process and reviews the main frameworks for its boundary evaluation. Here the term denotes an interval-valued response evolving with respect to a continuous independent variable, which may be physical time, spatial position, a design parameter, or the nominal value of an interval input; the formulation is thus not restricted to evolving systems.

2.1 Mathematical definition of the response interval process

A computational model is rigorously defined through a deterministic continuous real-valued function, with its mathematical formalism expressed as follows,

$$Y = f\left(\boldsymbol{X}, s\right), s \in \left[s_0, s_e\right] \tag{1}$$

where Y represents the response of interest in the model, $f\left(\boldsymbol{X}, s\right)$ is the performance function being analyzed. The vector $\boldsymbol{X} = \left(X_1, X_2, \cdots, X_n\right)^{\mathrm{T}}$ denotes the n-dimensional input variables, and the variable s denotes a continuous evolution variable. It is important to note that s and $\boldsymbol{X}$ are independent of each other.

Assuming that the model input $\boldsymbol{X}$ is uncertain and its available information is insufficient or incomplete, the input uncertainty can be quantified through the interval model. The uncertainty of an input X_i is represented as an interval variable as follows,

$$X_i^I = \left[X_i^l, X_i^u\right] = \left[X_i^c - X_i^r, X_i^c + X_i^r\right] \tag{2}$$

where the superscripts I, l and u denote the interval, lower bound and upper bound of X_i, respectively, and X_i^c is a nominal value of x_i within this interval. X_i^c is the midpoint of X_i^I, and X_i^r is the radius of X_i^I.

As shown in Fig. 1(a), when input uncertainty is represented using interval variables, the model response will also give rise to an interval at any given state point, i.e., $Y^I\left(s_j\right)$, which can be formulated as follows,

$$Y^I\left(s_j\right) = \left[Y^l\left(s_j\right), Y^u\left(s_j\right)\right] = \left[\min_{\boldsymbol{X} \in \boldsymbol{X}^I} f\left(\boldsymbol{X}, s_j\right), \max_{\boldsymbol{X} \in \boldsymbol{X}^I} f\left(\boldsymbol{X}, s_j\right)\right] \tag{3}$$

where $Y^l\left(s_j\right)$ and $Y^u\left(s_j\right)$ represent the minimum and maximum values of the output at the given state point s_j. Note that when the evolution variable s is fixed at a special state point s_j, the performance function depends solely on the vector $\boldsymbol{X}$.

Since the evolution variable s maintains continuity throughout the entire evolution process, the output uncertainty of the model is an interval process **Error! Reference source not found.**, which is shown in Fig. 1(b). The mathematical formula for this interval process is defined as follows,

$$Y^I(s)=\left[Y^l(s),Y^u(s)\right],s\in[s_0,s_e] \tag{4}$$

where $Y^l(s)$ is the lower boundary function, representing the minimum output of all possible outputs throughout the evolution process, $Y^u(s)$ is the upper boundary function, representing the maximum output of all possible outputs throughout the evolution process. As shown in Fig. 1(c), the interval process is surrounded by two consecutive boundary lines, representing that all possible outputs always fall within the region of the interval process.

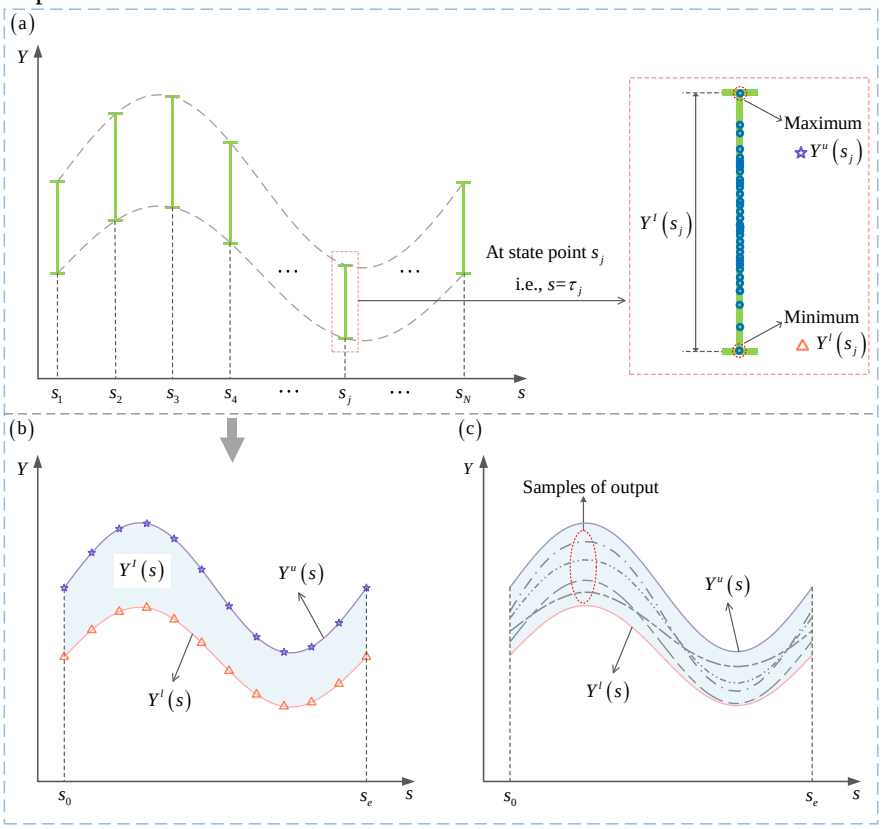

Fig. 1 Illustration of the response interval process

The response uncertainty is thus the region enclosed by the two boundaries. This interval process is related to the non-probabilistic convex model process (Jiang et al., 2014) for time-variant analysis, but here the focus is the interval process induced in the response rather than the correlation structure of interval inputs across states.

2.2 Discretization-optimization method

A direct way to compute the boundaries of a response interval process is the discretization-optimization method. This method discretizes the evolution domain into a set of state points and solves two static optimization problems at each state point to obtain the lower and upper response bounds.

Let the evolution domain $[s_0,s_e]$ be a closed and continuous interval. A finite set of state points is constructed via the discretization as follows,

$$\boldsymbol{S}=\left\{s_j\,\middle|\,s_j=s_0+(j-1)\Delta s,\ j=1,\cdots,N\right\},\Delta s=\frac{s_e-s_0}{N-1} \tag{5}$$

where N is the number of discretized state points, and Δs is the discretization step size. At each discretized state point s_j, two static optimization problems are solved to obtain the minimum value $Y^l(s_j)$ and the maximum value $Y^u(s_j)$ of the response, i.e.,

$$Y^l\left(s_j\right)=\min_{\boldsymbol{X}\in\boldsymbol{X}^I} f\left(\boldsymbol{X},s_j\right) \tag{6}$$

and

$$Y^u\left(s_j\right)=\max_{\boldsymbol{X}\in\boldsymbol{X}^I} f\left(\boldsymbol{X},s_j\right) \tag{7}$$

They can be solved by existing optimization algorithms (e.g., particle swarm optimization (PSO) **Error! Reference source not found.** or the genetic algorithm (GA) **Error! Reference source not found.**). As the discretization density increases, i.e., $m\to\infty$, the two sets of extrema asymptotically converge to the two boundaries, i.e.,

$$\lim_{m\to\infty} Y^l\left(s_j\right)=Y^l\left(s\right),\quad \lim_{m\to\infty} Y^u\left(s_j\right)=Y^u\left(s\right) \tag{8}$$

where $Y^l\left(s_j\right)$ and $Y^u\left(s_j\right)$ represent the values of the true lower boundary function $Y^l\left(s\right)$ and the true upper boundary function $Y^u\left(s\right)$ at the state point s_j. This methodology transforms the continuous boundary computation into $2m$ parameterized static optimization problems. Assuming that computing the minimum and maximum values at a single state point requires N_j^l and N_j^u evaluations of the performance function, the computational costs for determining the lower and upper boundaries are $N_l=\sum_{j=1}^{m}N_j^l$ and $N_u=\sum_{j=1}^{m}N_j^u$, respectively. However, in the context of engineering problems involving finite element analysis, each evaluation of the performance function may correspond to a time-consuming simulation, and the total number of evaluations required by the repeated inner optimizations quickly becomes unacceptable. To alleviate this computational burden, surrogate-based boundary approximation methods have been developed in recent studies, which are reviewed in the following subsection.

2.3 Surrogate-based boundary approximation

Although straightforward, the discretization-optimization method grows costly as the number of state points and the cost per inner optimization increase. Surrogate-based boundary approximation instead treats the lower and upper boundaries as unknown functions of the evolution variable and approximates them adaptively with surrogates such as Kriging (Kleijnen, 2009), radial basis functions (Buhmann, 2003), or neural networks (Hornik et al., 1989).

Building on the boundary function definitions given above, surrogate-based boundary approximation aims to learn the functional dependence of the two response boundaries on the evolution variable, rather than evaluating them at all discretized state points. Let $\hat{Y}^l\left(s\right)$ and $\hat{Y}^u\left(s\right)$ denote the surrogate models of the lower and upper response boundaries, respectively. They are constructed from two sets of boundary samples,

$$\boldsymbol{D}_l=\left\{\left(s_j^l,Y^l\left(s_j^l\right)\right)\right\}_{j=1}^{N_l},\quad \boldsymbol{D}_u=\left\{\left(s_j^u,Y^u\left(s_j^u\right)\right)\right\}_{j=1}^{N_u} \tag{9}$$

and provide the approximations as follows.

$$Y^l\left(s\right)\approx\hat{Y}^l\left(s\right),\quad Y^u\left(s\right)\approx\hat{Y}^u\left(s\right), s\in\boldsymbol{S} \tag{10}$$

A typical framework is two-layered. In the inner layer, an optimizer searches the interval input space at a fixed state point for the lower or upper extremum; in the outer layer, the boundary surrogate is updated over the evolution domain, with new state points chosen by an adaptive criterion, commonly the predictive variance (Jones et al., 1998), and true boundary values supplied by the inner-layer optimization. For the two boundaries the adaptive enrichment selects a new state point by a learning function as follows,

$$s_{N_l+1}^l=\arg\max_{s\in\boldsymbol{S}_l}\alpha_l\left(s\right),\quad s_{N_u+1}^u=\arg\max_{s\in\boldsymbol{S}_u}\alpha_u\left(s\right) \tag{11}$$

where $\boldsymbol{S}_l$ and $\boldsymbol{S}_u$ are the candidate sets for the lower and upper boundary surrogates, $\alpha_l\left(s\right)$ and $\alpha_u\left(s\right)$ are the corresponding learning functions. In variance-based adaptive sampling, the learning functions are commonly defined from the predictive uncertainty of the surrogate model, i.e.,

$$\alpha_l\left(s\right)=\sigma_l^2\left(s\right),\quad \alpha_u\left(s\right)=\sigma_u^2\left(s\right) \tag{12}$$

where $\sigma_l^2\left(s\right)$ and $\sigma_u^2\left(s\right)$ denote the predictive variances of the two boundary surrogates. After a new state point is selected, the corresponding true boundary value is obtained by invoking the inner-layer optimization, and the sample is added to $\boldsymbol{D}_l$ or $\boldsymbol{D}_u$ for surrogate updating.

In most implementations the two boundary surrogates are built independently and sequentially (Fig. 2): one is refined to convergence, then the other, so their sample sets and updates are separated and the minimum and maximum searches are performed as two separate optimizations.

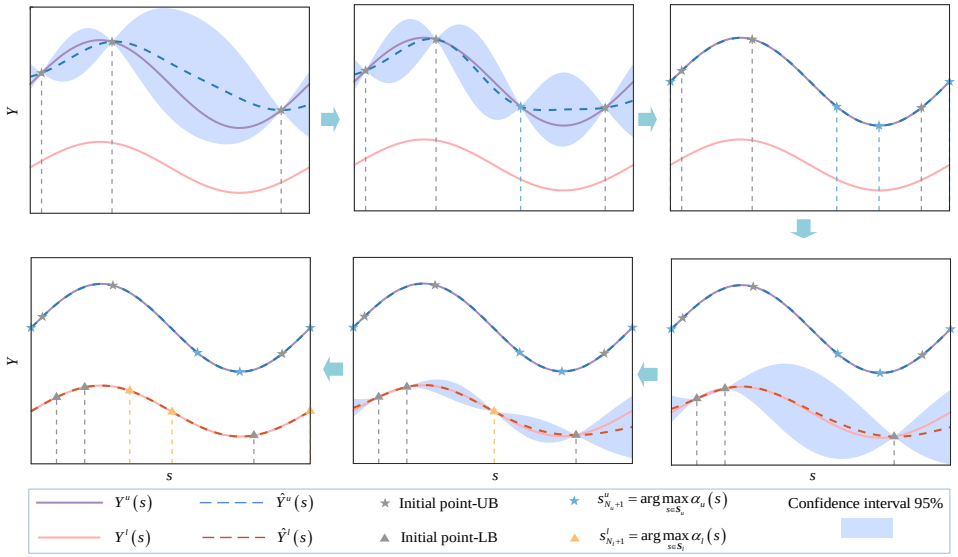

Fig. 2 Schematic of the independent sequential boundary surrogate strategy

In terms of computational cost, the surrogate models reduce the number of state points at which the inner optimization must be performed from the m grid points of section 2.2 to the adaptively selected ones, but each training sample of either boundary still requires one dedicated single-extremum optimization run. With N_0 initial state points per boundary, $N_{seq,l}$ and $N_{seq,u}$ adaptively added points, and $N_s^{(k)}$ performance-function evaluations consumed by the k-th run, the total cost of the sequential strategy is expressed as follows,

$$N_{eval}^{seq} = \sum_{k=1}^{2N_0+N_{seq,l}+N_{seq,u}} N_s^{(k)} = \left(2N_0 + N_{seq,l} + N_{seq,u}\right) \times \bar{N}_{single} \tag{13}$$

where $\bar{N}_{single}$ denotes the average number of evaluations of one single-extremum run.

This sequential strategy is simple but ignores the coupling between the two boundaries. At a given state point both extrema come from the same model and input space, so extremum information from one boundary calculation is wasted by the other. These limitations motivate the two improvements below: a bi-extremum inner strategy (Section 3) and an alternating outer framework (Section 4) in which the two boundary models are kept separate but updated cooperatively over the whole evolution domain.

3. Bi-extremum optimization at a fixed state point

This section addresses the inner-layer problem. Once the evolution variable is fixed at a state point, the model reduces to a static mapping of the interval inputs, and the lower and upper bounds follow from a pair of static extremum searches.

3.1 Problem definition at a fixed state point

Let the evolution variable be fixed at a state point s_j. The model response then depends only on the interval input vector $\boldsymbol{X}$, and the reduced performance function is denoted as $f_j(\boldsymbol{X}) \equiv f(\boldsymbol{X}, s_j)$. The inner-layer task is to solve the following pair of optimization problems.

$$Y^l(s_j) = \min_{\boldsymbol{X} \in \boldsymbol{X}^I} f_j(\boldsymbol{X}), \quad Y^u(s_j) = \max_{\boldsymbol{X} \in \boldsymbol{X}^I} f_j(\boldsymbol{X}) \tag{14}$$

The two extrema correspond to one sample of the lower boundary and one sample of the upper boundary of the response interval process, respectively.

In this work, the two extrema are searched within a Kriging-based efficient global optimization (EGO) framework (Jones et al., 1998). The Kriging model (Sacks et al., 1989) is adopted because its predictive mean and variance naturally support expected-improvement type acquisition functions.

3.2 Expected improvement for minimization

The conventional expected improvement criterion, denoted as $EI_{\min}$ in this work, is used for the minimization problem. As the core acquisition function of the EGO algorithm, it evaluates the expected amount by which an untried point can improve the currently observed minimum, while accounting for the predictive uncertainty of the surrogate model.

Here, an untried point refers to a candidate input vector in the interval input space that has not yet been evaluated by the true model. At any such point $\boldsymbol{x} \in \boldsymbol{X}^I$, the Kriging model provides a Gaussian predictive distribution with mean $\hat{y}(\boldsymbol{x})$ and standard deviation $\sigma(\boldsymbol{x})$.

$$Y(\boldsymbol{x}) \sim N\left(\hat{y}(\boldsymbol{x}), \sigma^2(\boldsymbol{x})\right) \tag{15}$$

If the current best observed value is $f_{\min}$, the improvement of an untried point $\boldsymbol{x}$ with respect to $f_{\min}$ is a random variable.

$$I_{\min}(\boldsymbol{x}) = \max\left(0, f_{\min} - Y(\boldsymbol{x})\right) \tag{16}$$

The $EI_{\min}$ function is defined as the mathematical expectation of this improvement.

$$EI_{\min}(\boldsymbol{x}) = E\left[I_{\min}(\boldsymbol{x})\right] \tag{17}$$

Solving the above expectation yields the following closed-form expression. i.e.,

$$EI_{\min}(\boldsymbol{x}) = \left(f_{\min} - \hat{y}(\boldsymbol{x})\right)\Phi\left(\frac{\left(f_{\min} - \hat{y}(\boldsymbol{x})\right)}{\sigma(\boldsymbol{x})}\right) + \sigma(\boldsymbol{x})\phi\left(\frac{f_{\min} - \hat{y}(\boldsymbol{x})}{\sigma(\boldsymbol{x})}\right) \tag{18}$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ denote the cumulative distribution function and probability density function of the standard normal distribution, respectively. The $EI_{\min}$ function can be understood as a nonlinear combination of exploitation and exploration. The first term of the $EI_{\min}$ function increases as $\hat{y}(\boldsymbol{x})$ decreases, which means it tends to select points with lower predicted response values, whereas the second term increases with $\sigma(\boldsymbol{x})$, encouraging exploration of regions with large predictive uncertainty.

At each iteration, the point maximizing $EI_{\min}$ over the interval input space is selected as the new training sample.

$$\boldsymbol{x}_{\min}^{*} = \arg\max_{\boldsymbol{X} \in \boldsymbol{X}^I} EI_{\min}(\boldsymbol{x}) \tag{19}$$

The $EI_{\min}$ function is continuous, equals zero at evaluated sample points, and is strictly positive elsewhere. Consequently, the point selected by maximizing $EI_{\min}$ never coincides with an existing sample. Under mild regularity conditions, the sequential infill strategy governed by the expected improvement criterion converges to the global optimum as the number of samples increases (Törn and Žilinskas, 1989). The stopping rule adopted in this work is defined in section 3.4.

3.3 Expected improvement for maximization

A maximization problem can be equivalently transformed into a minimization problem by changing the sign of the objective function, i.e., $\max_{\boldsymbol{x} \in \boldsymbol{X}^I} f_j(\boldsymbol{x}) \Leftrightarrow \min_{\boldsymbol{X} \in \boldsymbol{X}^I}\left(-f_j(\boldsymbol{x})\right)$. This transformation is mathematically precise and commonly used. However, in a conventional implementation, the minimization and maximization problems are addressed through two separate EGO runs. This means that two surrogate models are trained on distinct sample sets, and the costly model evaluations gathered during one search cannot be utilized by the other.

To enable the two searches to share a single Kriging model and a single training dataset, an expected improvement criterion for maximization, denoted as $EI_{\max}$, is formulated directly in the original response space as the counterpart of $EI_{\min}$.

Let $f_{\max}$ denote the currently observed maximum value. The improvement of an untried point $\boldsymbol{x}$ is defined as the amount by which its response exceeds $f_{\max}$.

$$I_{\max}(\boldsymbol{x}) = \max\left(0, Y(\boldsymbol{x}) - f_{\max}\right) \tag{20}$$

The $EI_{\max}$ function is defined as the mathematical expectation of this improvement.

$$EI_{\max}(\boldsymbol{x}) = E\left[I_{\max}(\boldsymbol{x})\right] \tag{21}$$

Evaluating the expectation gives the following expression.

$$EI_{\max}(\boldsymbol{x}) = \left(\hat{y}(\boldsymbol{x}) - f_{\max}\right)\Phi\left(\frac{\hat{y}(\boldsymbol{x}) - f_{\max}}{\sigma(\boldsymbol{x})}\right) + \sigma(\boldsymbol{x})\phi\left(\frac{\hat{y}(\boldsymbol{x}) - f_{\max}}{\sigma(\boldsymbol{x})}\right) \tag{22}$$

$EI_{\max}$ balances exploitation and exploration in the same manner as $EI_{\min}$, with the roles of the two terms mirrored for the maximization problem.

$$\boldsymbol{x}^*_{\max} = \arg\max_{\boldsymbol{X}\in\boldsymbol{X}^I} EI_{\max}(\boldsymbol{x}) \tag{23}$$

The characteristics of $EI_{\max}$ parallel function closely align with those of $EI_{\min}$ as it disappears at specific samples while remaining strictly positive in all other areas.

The $EI_{\max}$ criterion is mathematically equivalent to the conventional EI criterion applied to the sign-transformed objective. This equivalence guarantees that $EI_{\max}$ inherits the convergence properties of the standard expected improvement criterion, while keeping both the minimum and maximum searches formulated on the same surrogate model in the original response space, which is the prerequisite of the unified algorithm developed in the next subsection.

3.4 Unified bi-extremum optimization algorithm

The $EI_{\min}$ and $EI_{\max}$ criteria are coupled within a single EGO framework to search for the minimum and maximum responses at a fixed state point. Both searches share one Kriging model and one training dataset, rather than being performed by two isolated EGO runs. The algorithm consists of three stages, i.e., initialization, collaborative update, and separate refinement. The overall procedure is illustrated in Fig. 3.

To obtain a dimensionless stopping rule, the convergence indicators of the two searches are normalized by the currently observed interval width, i.e.,

$$Cr_{\min} = \frac{\max_{\boldsymbol{x}\in\boldsymbol{X}^I} EI_{\min}(\boldsymbol{x})}{f_{\max} - f_{\min}}, \quad Cr_{\max} = \frac{\max_{\boldsymbol{x}\in\boldsymbol{X}^I} EI_{\max}(\boldsymbol{x})}{f_{\max} - f_{\min}} \tag{24}$$

where $f_{\min}$ and $f_{\max}$ are the minimum and maximum observed responses in the current training dataset. A search is regarded as converged when its indicator falls below a prescribed threshold ε ($\varepsilon = 0.01$ in this work), i.e., the maximum expected improvement is less than 1% of the current interval width. To prevent premature termination caused by a transient underestimation of the expected improvement, the convergence condition is required to hold in two consecutive iterations. Note that this normalization is naturally available in the bi-extremum setting, since both extrema are tracked simultaneously; in addition, a maximum iteration number $N_{\max}$ is imposed as a safeguard.

1. Initialization stage

Step 1.1 Generate a candidate pool X_{pool} of N_{pool} points in the interval input space $\boldsymbol{X}^I$ by Latin hypercube sampling (LHS) (McKay et al., 1979), and randomly select N_0 initial training samples from the LHS portion of the pool.

Step 1.2 Evaluate the true responses of the initial samples, $y_i = f_j(\boldsymbol{x}_i)$ and form the initial training dataset $\boldsymbol{D} = \left\{(\boldsymbol{x}_i, y_i)\right\}, i = 1, \cdots, N_0$.

Step 1.3 Construct the initial Kriging model using $\boldsymbol{D}$, and set $f_{\min} = \min\{y_i\}$ and $f_{\max} = \max\{y_i\}$.

2. Collaborative update stage

Step 2.1 Evaluate $EI_{\min}(\boldsymbol{x})$ and $EI_{\max}(\boldsymbol{x})$ over the candidate pool and compute $Cr_{\min}$ and $Cr_{\max}$ by the convergence-indicator equation above. If $Cr_{\min} \le \varepsilon$ and $Cr_{\max} \le \varepsilon$, go to Step 4.1. If exactly one indicator satisfies its threshold, go to Step 3.1. Otherwise, continue with Step 2.2.

Step 2.2: Select $\boldsymbol{x}^*_{\min} = \arg\max_{\boldsymbol{X}\in\boldsymbol{X}^I} EI_{\min}(\boldsymbol{x})$ and $\boldsymbol{x}^*_{\max} = \arg\max_{\boldsymbol{X}\in\boldsymbol{X}^I} EI_{\max}(\boldsymbol{x})$ as the new samples for the minimum and maximum searches, respectively.

Step 2.3: Evaluate the true responses $f_j(\boldsymbol{x}^*_{\min})$ and $f_j(\boldsymbol{x}^*_{\max})$, and add both samples to $\boldsymbol{D}$. The two evaluations are independent and can be executed in parallel.

Step 2.4: Retrain the Kriging model with the updated $\boldsymbol{D}$, refresh $f_{\min} = \min\{y_i\}$ and $f_{\max} = \max\{y_i\}$, and return to Step 2.1.

3. Separate refinement stage

This stage is executed only for the extremum whose indicator has not yet converged. The procedure is stated for the minimum search, and the maximum search is refined in the mirrored manner using $EI_{\max}$ and $Cr_{\max}$.

Step 3.1: Select $\boldsymbol{x}^*_{\min} = \arg\max_{\boldsymbol{X} \in \boldsymbol{X}^l} EI_{\min}(\boldsymbol{x})$ as the new sample.

Step 3.2: Evaluate the true response $f_j(\boldsymbol{x}^*)$ and add it to $\boldsymbol{D}$.

Step 3.3: Retrain the Kriging model and refresh $f_{\min} = \min\{y_i\}$. Both $f_{\min}$ and $f_{\max}$ are refreshed, since a refinement sample may incidentally improve the other extremum.

Step 3.4: Recompute $Cr_{\min}$ by the convergence-indicator equation above. If $Cr_{\min} > \varepsilon$, return to Step 3.1. Otherwise, go to Step 4.1.

Output

Step 4.1: Return the observed extrema $Y^l(s_j) = f_{\min}$ and $Y^u(s_j) = f_{\max}$ as the lower and upper bounds at the state point s_j.

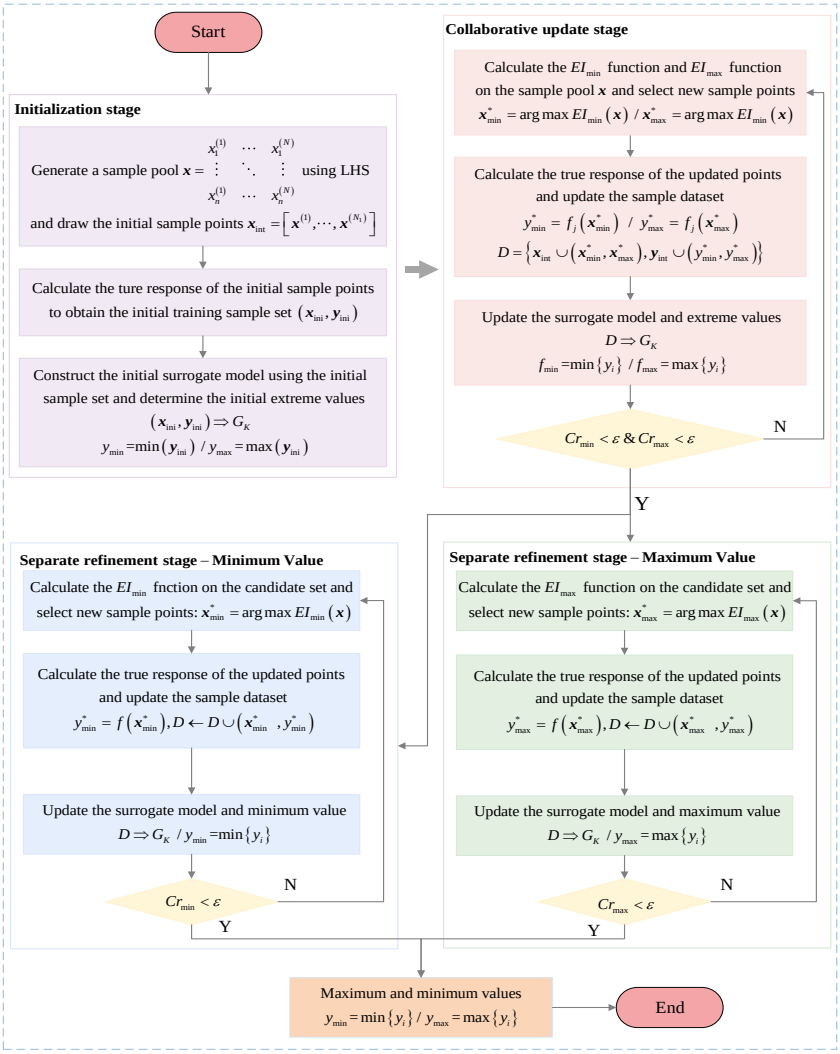

Fig. 3 Flowchart of the unified bi-extremum optimization algorithm at a fixed state point

The computational cost of the algorithm is dominated by the number of true model evaluations. Compared with two independent EGO runs, the unified strategy lets every evaluated sample inform both searches through the shared Kriging model, and the two evaluations in each collaborative-update iteration

can be executed in parallel, which further reduces the wall-clock time for expensive simulations. Moreover, since $f_{\min}$ and $f_{\max}$ are taken from evaluated samples, the returned interval $[f_{\min}, f_{\max}]$ is an inner estimate of the true response interval, and the residual normalized indicators bound the expected further widening: upon convergence, the expected underestimation of the interval width does not exceed ε times the current width. Finally, because the returned extrema are true model responses rather than surrogate predictions, they provide noise-free training data for the outer boundary surrogate framework developed in section 4.

4. Alternating surrogate modeling for response interval process boundaries

This section develops the outer layer. The bi-extremum algorithm of Section 3 returns both extrema at every visited state point, whereas the sequential strategy of Section 2.3 keeps only one and discards the other. The alternating strategy below ensures that both outputs of every inner-layer call enter the training data of the two boundary surrogates; the resulting method is termed the Bi-Extremum Alternating Surrogate (BEAS) method.

4.1 Alternating update strategy with conditional injection

The outer layer trains the two boundary surrogates $\hat{Y}^u(s)$ and $\hat{Y}^l(s)$ by adaptively adding samples to their training sets $\boldsymbol{D}_u$ and $\boldsymbol{D}_l$. Each boundary maintains a candidate pool, $\boldsymbol{S}_u$ and $\boldsymbol{S}_l$. Each boundary further maintains a buffer, $\boldsymbol{B}_u$ and $\boldsymbol{B}_l$, initially empty, which stores by-product values delivered to that boundary but not yet admitted into its training set.

The central mechanism of the strategy is the transfer of the by-product between successive half-steps. When a half-step evaluates *Ain* at a selected state point s^*, the operator returns both boundary values simultaneously. One of them is the primary sample requested by the active boundary and is added to its own training set. The other is a by-product, namely a true sample of the opposite boundary obtained without any additional execution of *Ain*, and is transferred to the opposite boundary. The transferred by-product serves as the update information with which the opposite boundary begins its own half-step. Successive half-steps are thereby linked into a chain, i.e., the upper-boundary half-step supplies a by-product that updates the lower boundary, the lower-boundary half-step supplies a by-product that updates the upper boundary, and so on in alternation.

Each outer iteration consists of two symmetric half-steps, one for each boundary. Taking the upper boundary as the active boundary, a half-step comprises the following five steps. The lower-boundary half-step is obtained by interchanging $u \leftrightarrow l$.

(i) **Step 1 (Injection and update)**. The by-product transferred from the preceding half-step, if admitted by the conditional injection rule as specified in case 1 below, is added to the active training set, and the active surrogate is refitted,

$$\boldsymbol{D}_u \leftarrow \boldsymbol{D}_u \cup \left\{s', Y^u(s')\right\}, \quad Y^u(s) \leftarrow GK\left(\boldsymbol{D}^u\right) \tag{25}$$

where s' is the state point of the transferred by-product and $GK(\cdot)$ denotes Kriging model fitting. The information carried by the by-product is thus incorporated before any decision of the current half-step is made.

(ii) **Step 2 (Convergence assessment)**. The normalized indicator is defined as follows,

$$Cr_u = \frac{\max_{s \in \boldsymbol{S}_u} \sigma_u(s)}{W}, W = \frac{1}{N_s} \sum_{s \in \boldsymbol{S}} \left(\hat{Y}^u(s) - \hat{Y}^l(s)\right) \tag{26}$$

where $\sigma_u(s)$ is the predictive standard deviation of $\hat{Y}^u(s)$. If $Cr_u \le \varepsilon_{\text{out}}$, the upper boundary is flagged converged and the half-step terminates without executing *Ain*. The algorithm stops when both boundaries are converged; a maximum iteration number $N_{\max}$ is imposed as a safeguard.

(iii) **Step 3 (State-point selection)**. The next state point is selected as the pool point of maximum predictive variance,

$$s^* = \arg\max_{s \in \boldsymbol{S}_u} \sigma_u^2(s) \tag{27}$$

(iv) **Step 4 (Inner-layer evaluation)**. If s^* coincides with a buffered state point of the active boundary, its stored value is admitted directly and no execution of A_{in} is performed, as specified in case 3 below. Otherwise, a single inner-layer execution is carried out, i.e.,

$$\left[Y^l\left(s^*\right), Y^u\left(s^*\right)\right] = Ain\left(s^*\right) \tag{28}$$

after which the primary sample is added to the active training set and the state point leaves the active pool.

$$\boldsymbol{D}_u \leftarrow \boldsymbol{D}_u \cup \left\{s^*, Y^u\left(s^*\right)\right\}, \quad \boldsymbol{S}_u \leftarrow \boldsymbol{S}_u \setminus \left\{s^*\right\} \tag{29}$$

(v) **Step 5 (By-product transfer)**. The by-product $\left(s^*, Y^u\left(s^*\right)\right)$ is transferred to the opposite boundary, where its destination is determined by the conditional injection rule based on its distance to the receiving training set.

$$d\left(s^*, \boldsymbol{D}_l\right) = \min_{s_j \in \boldsymbol{S}_l} \left|s^* - s_j\right| \tag{30}$$

The rule assigns the by-product to exactly one of three cases, and which is illustrated in Fig. 4.

(i) **Case 1 (Direct injection)**. If $d\left(s^*, \boldsymbol{D}_l\right) > d_{\min}$, the by-product is admitted into the lower training set and its state point leaves the lower pool.

$$\boldsymbol{D}_l \leftarrow \boldsymbol{D}_l \cup \left\{s^*, Y^l\left(s^*\right)\right\}, \quad \boldsymbol{S}_l \leftarrow \boldsymbol{S}_l \setminus \left\{s^*\right\} \tag{31}$$

(ii) **Case 2 (Screened buffering)**. If $d\left(s^*, \boldsymbol{D}_l\right) < d_{\min}$, the by-product is stored in the buffer and its state point is retained in the pool,

$$\boldsymbol{B}_l \leftarrow \boldsymbol{B}_l \cup \left\{s^*, Y^l\left(s^*\right)\right\}, \quad \boldsymbol{S}_l \leftarrow \boldsymbol{S}_l \tag{32}$$

Retaining s^* in $\boldsymbol{S}_l$ is essential that the lower-boundary value at s^* has not entered $\boldsymbol{D}_l$, so by the pool invariant the point remains a legitimate candidate for later selection. The threshold $d_{\min} = 1.5\Delta s$ is adopted in this work.

(iii) **Case 3 (Buffer retrieval)**. If a selection s^* made by operation (iii) of a later half-step coincides with a buffered state point, the stored value is admitted with no execution of Ain.

$$\boldsymbol{D}_l \leftarrow \boldsymbol{D}_l \cup \left\{s^*, Y^l\left(s^*\right)\right\}, \quad \boldsymbol{B}_l \leftarrow \boldsymbol{B}_l \cup \left\{s^*, Y^l\left(s^*\right)\right\}, \quad \boldsymbol{S}_l \leftarrow \boldsymbol{S}_l \setminus \left\{s^*\right\} \tag{33}$$

Delivery in case 1 or case 2 takes place even if the receiving boundary is already converged, since registering a by-product requires no execution of Ain and, by the variance-monotonicity property discussed below, never increases the predictive variance of the receiving surrogate.

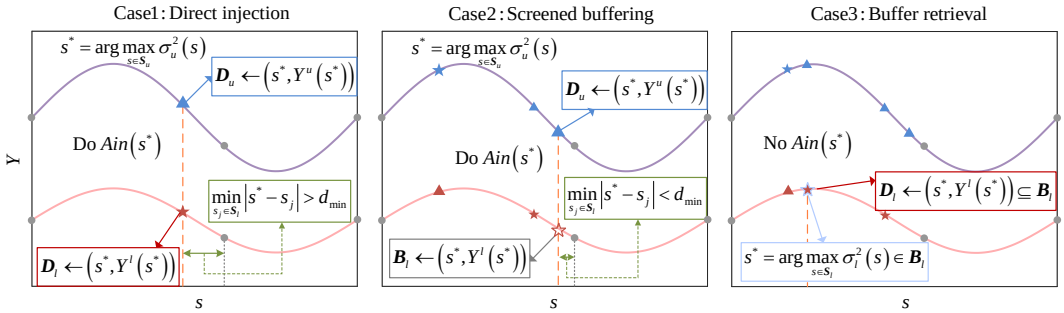

Fig. 4 Three by-product-handling cases in the alternating update strategy

Two properties of the Kriging predictive variance underlie the rule. The distance screening is applied only to a transferred by-product, because the predictive variance vanishes at every training point and the selection of step 3 is thereby kept away from existing samples, whereas a by-product is located by the opposite variance field and may fall arbitrarily close to a sample of the receiving set and render its correlation matrix ill-conditioned. Retrieval in case 3 is feasible because geometric distance and correlation distance differ, so that a by-product screened out as geometrically close may still carry a high predictive variance when its separation exceeds the correlation length and be selected in a later half-step. Retrieval is therefore governed by the ratio of $d_{\min}$ to the correlation length and does not occur under the default setting.

4.2 The complete BEAS procedure

The complete Bi-Extremum Alternating Surrogate (BEAS) method nests the bi-extremum inner-layer algorithm of section 3 within the alternating outer strategy of section 4.1. Following an initialization that executes *Ain* at N_0 uniformly spaced state points to form the two initial training sets and surrogates, the boundaries are updated in alternating half-steps of the five-step form of section 4.1. The first upper-boundary half-step starts from the initial surrogate with no by-product to incorporate, and every subsequent half-step incorporates the transferred by-product, assesses convergence, and, if not converged, selects the state point of maximum predictive variance and evaluates it by buffer retrieval or by one execution of *Ain*. The procedure terminates when both boundaries are converged or when $k > N_{\max}$. The complete procedure is summarized in Algorithm 1.

Algorithm 1: Alternating update with conditional injection (outer layer of BEAS)

Require: candidate grid $\boldsymbol{S}$ over the evolution domain, number of initial state points N_0, inner bi-extremum algorithm $Ain(\cdot)$, screening distance $d_{\min}$, threshold ε_{out}, maximum iterations $N_{\max}$.

Ensure: boundary surrogates $\hat{Y}^l(s)$ and $\hat{Y}^u(s)$

1: Select N_0 initial state points; call $Ain(\cdot)$ at each to obtain both extrema.

2: $\boldsymbol{D}_u, \boldsymbol{D}_l$ initial samples, $\boldsymbol{S}_u, \boldsymbol{S}_l$ initial points, $\boldsymbol{B}_u, \boldsymbol{B}_l \leftarrow \varnothing$, $conv_u, conv_l \leftarrow false$, $k = 1$.

3: while $conv_u = false \vee conv_l = false$ and $k \le N_{\max}$ do

4: for $side \in \{u, l\}$ do // two half-steps per iteration

5: if $conv_{side} = ture$ then continue

6: Retrain $\hat{Y}_{side}$ on $\boldsymbol{D}_{side}$ if $\boldsymbol{D}_{side}$ has changed. // absorbs injections

7: Compute Cr_{side}, if $Cr_{side} \le \varepsilon_{out}$ then $conv_{side} \leftarrow ture$, continue. // convergence gate

8: $s^* \leftarrow \arg\max_{s \in \boldsymbol{S}_{side}} \sigma^2_{side}(s)$.

9: if $s^* \in \boldsymbol{B}_{side}$ then // Case 3: retrieval, no inner call

10: Retrieve $Y^{side}(s^*)$ from $\boldsymbol{B}_{side}$, $\boldsymbol{B}_{side} \leftarrow \boldsymbol{B}_{side} \setminus \{s^*\}$.

11: else

12: $\left[Y^l(s^*), Y^u(s^*)\right] = Ain(s^*)$. // one inner call, both extrema

13: if $\min\left(d\left(s^*, \boldsymbol{D}_{other}\right)\right) \ge d_{\min}$ then // Case 1: injection

14: $\boldsymbol{D}_{other} \leftarrow \boldsymbol{D}_{other} \cup \left\{\left(s^*, Y^{other}(s^*)\right)\right\}$, $\boldsymbol{S}_{other} \leftarrow \boldsymbol{S}_{other} \setminus \{s^*\}$.

15: else $\boldsymbol{B}_{other} \leftarrow \boldsymbol{B}_{other} \cup \left\{\left(s^*, Y^{other}(s^*)\right)\right\}$. // Case 2: buffering

16: end if

17: $\boldsymbol{D}_{side} \leftarrow \boldsymbol{D}_{side} \cup \left\{\left(s^*, Y^{side}(s^*)\right)\right\}$, $\boldsymbol{S}_{side} \leftarrow \boldsymbol{S}_{side} \setminus \{s^*\}$.

18: end for

19: $k \to k+1$.

20: end while

21: Final retraining of both surrogates; return $\hat{Y}^l(s)$ and $\hat{Y}^u(s)$.

4.3 Computational cost analysis

The computational cost of BEAS is measured by the number of executions of the inner-layer algorithm *Ain* and by the resulting number of performance-function evaluations. For each boundary, the executions consumed by its own half-steps equal the number of adaptively added state points minus the number of buffer retrievals, since a retrieved point is admitted without executing *Ain*. These counts are $N_{new,u} - N_{ret,u}$ for the upper boundary and $N_{new,l} - N_{ret,l}$ for the lower boundary. Together with the N_0 initializing executions, the total number of inner-layer executions is expressed as follows.

$$N_{call} = N_0 + \left(N_{new,u} - N_{ret,u}\right) + \left(N_{new,l} - N_{ret,l}\right) \tag{34}$$

Denoting by N_{Ain}^k the number of performance-function evaluations consumed by the *k*-th execution, the total cost of the boundary evaluation is expressed as follows,

$$N_{eval}^{\mathrm{BEAS}} = \sum_{k=1}^{N_{call}} N_{Ain}^{(k)} = N_{call} \times \bar{N}_{bi} \tag{35}$$

where $\bar{N}_{bi}$ is the average cost of one bi-extremum execution.

Compared with the sequential strategy of section 2.3, whose cost is in the Eq. (13), BEAS reduces both factors of the product. The number of executions is reduced because every transferred by-product enters the training set of the receiving boundary without an execution of Ain, so the adaptive additions $N_{new,u}$ and $N_{new,l}$ required to resolve the two boundaries are fewer than the corresponding numbers of the sequential construction, and the buffer retrievals of case 3 remove further executions. The cost per execution satisfies $\bar{N}_{bi} < 2\bar{N}_{single}$, as established in section 3, because the two extremum searches share one surrogate model and one training dataset.

5. Examples and discussion

Two analytical examples validate the method. For each, the boundary iteration is shown for the proposed method, and efficiency is compared among three schemes: sequential single-boundary iteration with a PSO inner solver, the same with an EGO inner solver, and the proposed alternating-boundary iteration with the bi-extremum solver (BEAS). The total adaptively added state points, equal to the number of inner-layer executions, measures the outer-layer cost.

5.1 Experimental setup

Two analytical examples are used to evaluate the accuracy and efficiency of the proposed BEAS framework. Example 1 is a smooth double-boundary case that examines the baseline behavior of the algorithm, and its performance function is defined as follows,

$$\begin{aligned} f_1(\boldsymbol{X}, s) = X_1^2 + X_2^2 + X_3X_4 + X_1X_2\left(1.2\sin(s+0.8) + 0.3\sin(2s+1.5)\right) + \\ X_3X_4\left(0.9\cos(s-1.2) + 0.4\cos(1.5s-0.7)\right) + 0.25X_1X_3\sin(0.8s+2.1) \end{aligned} \tag{36}$$

where the interval input vector is $\boldsymbol{X}^I = \left(X_1^I, X_2^I, X_3^I, X_4^I\right)$ with $n = 4$ interval inputs. All inputs share the common midpoint $X_i^c = 0.5$ with radii $X_i^r = (0.07, 0.175, 0.35, 0.28)$, the evolution domain is $s \in [2,10]$. The harmonic terms of incommensurate frequencies produce smooth and slowly varying boundaries.

Example 2 augments the base function with an oscillatory term,

$$Y = f_2(\boldsymbol{X}, s) = f_1(\boldsymbol{X}, s) + 4(X_3 - 0.5)^2(1.5 + \sin 3s) \tag{37}$$

which is non-negative and vanishes at $X_3 = 0.5$ and oscillates for about 3.8 periods over the evolution domain. The upper boundary of Example 2 is therefore highly oscillatory while the lower boundary remains smooth, which produces an asymmetric sample demand between the two boundaries.

The reference boundaries are obtained by the discretization-optimization framework of section 2.2, with the inner optimization at each grid point replaced by an exhaustive evaluation over a fixed input set consisting of 10^5 uniformly distributed samples and the 2^n vertices of $\boldsymbol{X}^I$. At each grid point the reference boundaries are taken as the extrema of the performance function over this fixed set, $Y_{\mathrm{ref}}^l(s_j) = \min_{\boldsymbol{x} \in \boldsymbol{X}_{\mathrm{ref}}} f(\boldsymbol{x}, s_j)$ and $Y_{\mathrm{ref}}^u(s_j) = \max_{\boldsymbol{x} \in \boldsymbol{X}_{\mathrm{ref}}} f(\boldsymbol{x}, s_j)$, where $\boldsymbol{X}_{\mathrm{ref}}$ denotes the union of the sample set and the vertex set. The vertices are included because a finite random sample contains vertex-type extrema with probability zero, so purely random sampling would bias the reference interval inward. The reference thus realizes the discretization-optimization evaluation of section 2.2 with an exhaustive inner search and serves only for error assessment. All boundary errors are normalized by the reference mean interval width,

$$Err_u = \frac{\max_{s \in \boldsymbol{S}} \left|\hat{Y}^u(s) - Y_{\mathrm{ref}}^u(s)\right|}{\bar{W}_{\mathrm{ref}}}, \quad Err_l = \frac{\max_{s \in \boldsymbol{S}} \left|\hat{Y}^l(s) - Y_{\mathrm{ref}}^l(s)\right|}{\bar{W}_{\mathrm{ref}}} \tag{38}$$

where $\bar{W}_{\mathrm{ref}}$ is the mean width of the reference interval process, equal to 1.096 for Example 1 and 2.633 for Example 2. The parameters used in the experiments are listed in Table 1.

Table 1 Parameters used in the numerical experiments

Parameter	Value	Remark
Number of grid points N_s	500	Same for both examples
Initial state points N_0	4 (Example 1), 6 (Example 2)	Uniformly spaced
Screening distance $d_{\min}$	$1.5\Delta s$	Same for both examples
Outer threshold ε_{out}	0.01	Same for both examples
Inner candidate pool	10^5 LHS samples	Same for both examples
Inner threshold ε	0.01	Two consecutive iterations

5.2 Example 1 results

Fig. 6 shows the boundary iteration process of the sequential strategy for Example 1, and Fig. 7 shows that of the proposed alternating strategy.

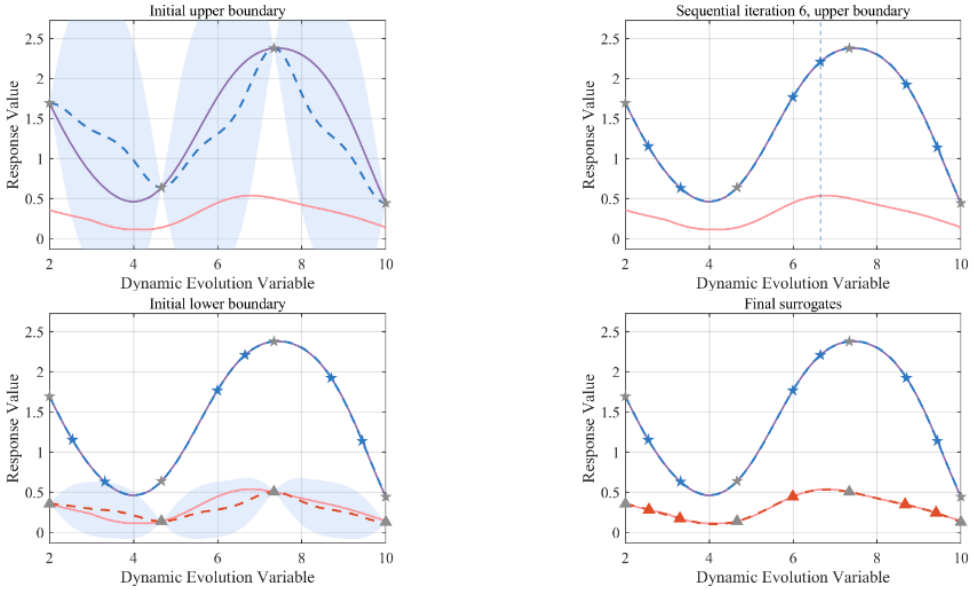

Fig. 6 Boundary iteration process of the sequential strategy for Example 1

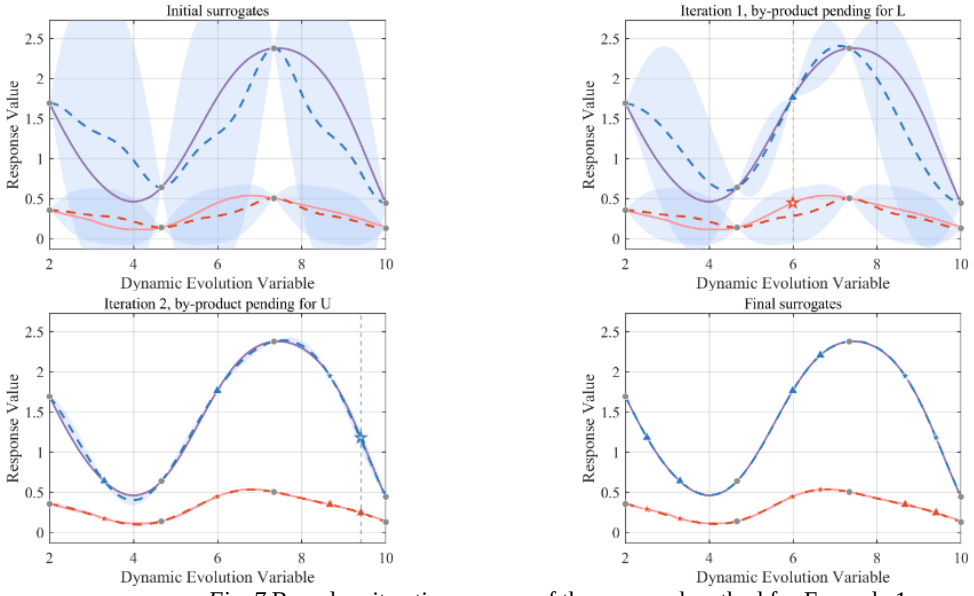

Fig. 7 Boundary iteration process of the proposed method for Example 1

The added state points in Table 2 are visible in the two figures. In Fig. 6 each colored marker is one of the 19 single-extremum runs of the sequential strategy; in Fig. 7 the stars mark the 10 bi-extremum executions of BEAS (4 for initialization) and the triangles are by-products obtained without any execution. The shrinking prediction bands and the matching final panels confirm the equal accuracy reported in Table 2.

Table 2 compares the three schemes for Example 1. For the sequential schemes the added state points are counted per state point at which an inner-layer optimization is executed, and the lower and upper responses are obtained by two separate single-extremum runs, whereas for the proposed method each added state point corresponds to a single bi-extremum execution that returns both responses; the by-products transferred between boundaries are reused without recomputation and are not counted.

Table 2 Efficiency and cost comparison for Example 1

Scheme	Outer strategy	Inner solver	Total state points	N_{eval}	Err_u (%)	Err_l (%)
S-PSO	sequential	PSO	19.0	16968.0	0.398	1.170
S-EGO	sequential	single-extremum EGO	19.0	325.6	0.398	1.177
BEAS	alternating	bi-extremum	10.0	191.0	0.391	1.201

The proposed method attains the same boundary accuracy as the two sequential baselines while reducing the cost on two independent levels. The upper- and lower-boundary errors of the three schemes are at the same level for both examples, so the cost comparison is made at equal accuracy.

The first level compares the two inner solvers. S-PSO and S-EGO share the sequential outer strategy and the same added state points, differing only in the inner solver. Since particle swarm optimization needs about a thousand evaluations per search, S-PSO costs roughly fifty times the evaluations of S-EGO, confirming the value of a Kriging surrogate in expensive-simulation settings.

The second level compares alternating reuse against sequential independent construction. S-EGO and BEAS use the same Kriging inner solver and differ only in the outer strategy. Because BEAS returns both extrema per execution and reuses the transferred by-product without recomputation, the inner-layer executions drop by about 47% and the evaluations by about 41%. This gain is independent of the inner solver and comes solely from by-product reuse.

5.3 Example 2 results

Fig. 8 shows the boundary iteration process of the sequential strategy for Example 2, and Fig. 9 shows that of the proposed alternating strategy in the representative run.

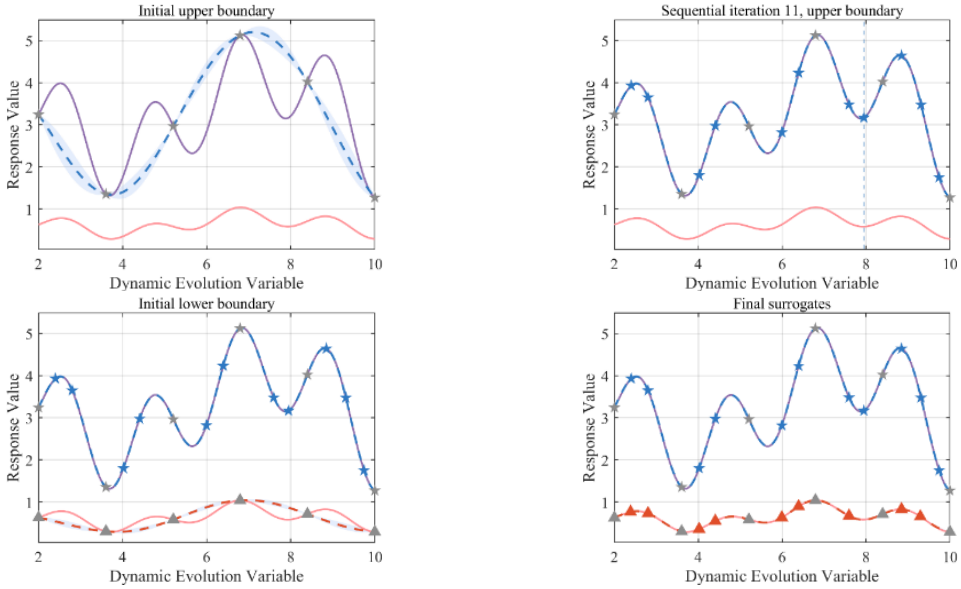

Fig. 8 Boundary iteration process of the sequential strategy for Example 2

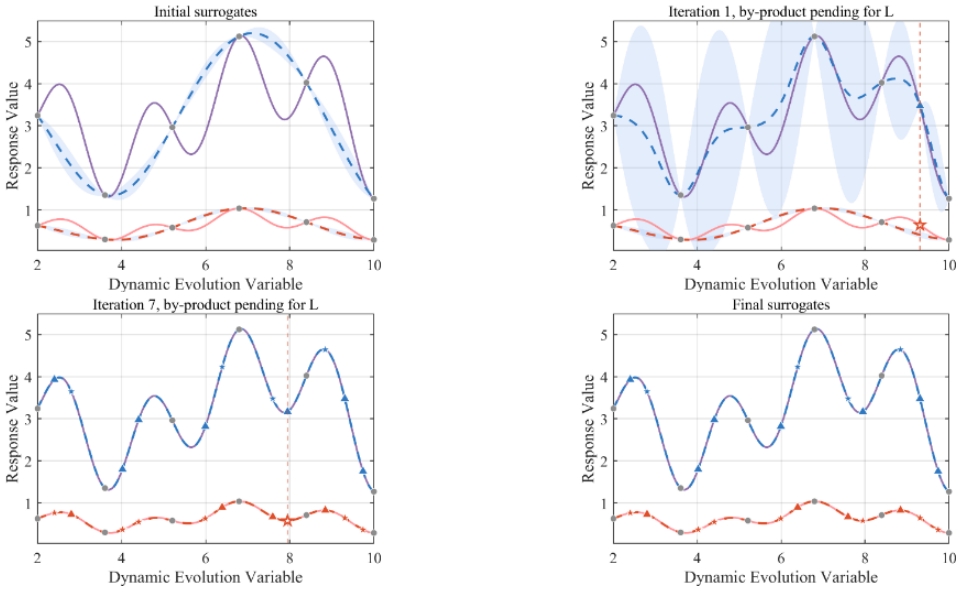

Fig. 9 Boundary iteration process of the proposed method for Example 2

The two figures make the outer-layer cost of Table 3 visible. In Fig. 8 every marker beyond the initial samples is one single-extremum run, with the oscillatory upper boundary consuming most of the 32 runs. In Fig. 9 the smooth lower boundary converges after its fourth half-step, while the remaining upper-boundary half-steps still deliver by-products to it. The eleven adaptive executions split seven to four between the boundaries and, with the six initial ones, give the seventeen added state points of Table 3. The triangles mark samples obtained without any inner execution, the source of the cost reduction.

Table 3 compares the three schemes for Example 2.

Table 3 Efficiency and cost comparison for Example 2

Scheme	Outer strategy	Inner solver	Total state points	N_{eval}	Err_u (%)	Err_l (%)
S-PSO	sequential	PSO	32.0	28568.0	0.310	0.288
S-EGO	sequential	single-extremum EGO	32.0	544.8	0.237	0.330
BEAS	alternating	bi-extremum	17.0	323.2	0.382	0.067

For Example 2 the same two-level reduction holds, with about 41% fewer performance-function evaluations than the single-extremum sequential scheme. Because the upper boundary is oscillatory and the lower one smooth, the added state points concentrate on the upper boundary and the absolute number of executions exceeds that of Example 1; yet the relative reduction of inner-layer executions is again about 47%. This cross-example consistency shows that the benefit of by-product reuse is insensitive to boundary symmetry.

6. Conclusion

This paper developed a dual-layer Bi-Extremum Alternating Surrogate (BEAS) method for the boundaries of a response interval process. At the inner layer, an EI criterion for maximization was formulated directly in the original response space and coupled with the minimization criterion in a single Kriging-based EGO framework, so both extrema at a state point are found from one surrogate and one training set, returning true, noise-free responses for the outer layer. At the outer layer, an alternating strategy updates the two boundary surrogates in half-steps: each inner call's by-product extremum is delivered to the opposite boundary and, under a distance-based screening rule protecting the Kriging conditioning, injected into its training set, with a buffer as a zero-cost retrieval path.

Two conclusions follow on cost. First, at equal accuracy BEAS reduces evaluations on two independent levels: replacing the population-based inner solver by the Kriging surrogate cuts evaluations by about fifty times, and replacing the sequential outer construction by alternating reuse cuts inner-layer executions by about 47% and evaluations by a further 41%. Second, this reduction is close for the smooth and oscillatory examples, so the benefit of by-product reuse is insensitive to boundary symmetry.

The main limitation is the common blind spot of variance-based sampling: boundary features narrower than the sample spacing may go undetected, and the outer indicator is a probabilistic rather than deterministic error measure. Future work includes applying BEAS to engineering problems with expensive simulations and extending the alternating framework to vector-valued responses and higher-dimensional evolution variables.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant No. NSFC52475283).

References

Buhmann, M. D. Radial Basis Functions: Theory and Implementations, Cambridge University Press, 2003.

Chang, Q., C. Zhou, M. A. Valdebenito, H. Liu, and Z. Yue. A novel sensitivity index for analyzing the response of numerical models with interval inputs, Comput. Methods Appl. Mech. Eng. 400 (2022) 115509.

Faes, M., J. Cerneels, D. Vandepitte, and D. Moens. Identification and quantification of multivariate interval uncertainty in finite element models, Comput. Methods Appl. Mech. Eng. 315 (2017) 896–920.

Forrester, A. I. J. and A. J. Keane. Recent advances in surrogate-based optimization, Prog. Aerosp. Sci. 45 (2009) 50–79.

Goldberg, D. E. Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

Guo, S. X. and Z. Z. Lu. A non-probabilistic robust reliability method for analysis and design optimization of structures with uncertain-but-bounded parameters, Appl. Math. Model. 39 (2015) 1985–2002.

Hornik, K., M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators, Neural Netw. 2 (1989) 359–366.

Jiang, C., X. Han, G. Y. Lu, J. Liu, Z. Zhang, and Y. C. Bai. Non-probabilistic convex model process: A new method of time-variant uncertainty analysis and its application to structural dynamic reliability problems, Comput. Methods Appl. Mech. Eng. 268 (2014) 656–676.

Jones, D. R., M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions, J. Glob. Optim. 13 (1998) 455–492.

Kennedy, J. and R. Eberhart. Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, 1995, pp. 1942–1948.

Kleijnen, J. P. C. Kriging metamodeling in simulation: A review, Eur. J. Oper. Res. 192 (2009) 707–716.

Liu, Z. C., H. Y. Huang, X. Y. Xu, M. Xiong, and Q. Z. Li. An efficient global optimization algorithm combining revised expectation improvement criteria and Kriging, Eng. Optim. 56 (2023) 608–624.

McKay, M. D., R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, Technometrics 21 (1979) 239–245.

Mockus, J. Bayesian Approach to Global Optimization, Springer, 2012.

Mohammadi, H. and R. Le Riche. Small ensembles of Kriging models for optimization, arXiv preprint arXiv:1603.02638, 2016.

Ni, B. Y., C. Jiang, J. W. Li, and W. Y. Tian. Interval K-L expansion of interval process model for dynamic uncertainty analysis, J. Sound Vib. 474 (2020) 115254.

Niu, Z., C. Yan, and Y. Li. Non-probabilistic credible reliability analysis of the composite laminate, Aerosp. Sci. Technol. 144 (2024) 108774.

Sacks, J., W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments, Stat. Sci. 4 (1989) 409–423.

Shahriari, B., K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization, Proc. IEEE 104 (2016) 148–175.

Song, R., C. Zhou, J. Rui, H. Li, and J. Li. A volume-ratio index for sensitivity analysis of time-dependent models with interval uncertainty inputs, Aerosp. Sci. Technol. 153 (2024) 109495.

Tang, T., H. Luo, Y. Song, H. Fang, and J. Zhang. Chebyshev inclusion function based interval kinetostatic modeling and parameter sensitivity analysis for Exechon-like parallel kinematic machines with parameter uncertainties, Mech. Mach. Theory 157 (2021) 104209.

Törn, A. and A. Žilinskas. Global Optimization, Springer, 1989.

Wang, L., Y. Liu, D. Liu, and Z. Wu. A novel dynamic reliability-based topology optimization (DRBTO) framework for continuum structures via interval-process collocation and the first-passage theories, Comput. Methods Appl. Mech. Eng. 386 (2021) 114107.

Wang, Q. N., L. M. Song, Y. Chen, G. J. Ma, Z. D. Guo, and J. Li. KT-EGO: a knowledge transfer assisted efficient global optimization algorithm for solving high-dimensional expensive black-box problems, Eng. Optim. 55 (2023) 2015–2033.

Wei, P., Y. Wang, and C. Tang. Time-variant global reliability sensitivity analysis of structures with both input random variables and stochastic processes, Struct. Multidisc. Optim. 55 (2017) 1883–1898.

Wu, W. and S. S. Rao. Uncertainty analysis and allocation of joint tolerances in robot manipulators based on interval analysis, Reliab. Eng. Syst. Saf. 92 (2007) 54–64.

Yao, Z. Y., S. H. Wang, P. G. Wu, B. Y. Ni, and C. Jiang. An interval finite element method based on bilevel Kriging model, Chin. J. Aeronaut. 37 (2024) 1–11.

Zhan, H. Y. and N. C. Xiao. A new active learning surrogate model for time- and space-dependent system reliability analysis, Reliab. Eng. Syst. Saf. (2025) 110536.