Cross-Entropy Method Variants for Optimization

Robert J. Moss
Stanford University, Computer Science
Stanford, CA, 94305
mossr@cs.stanford.edu

***Abstract*—The cross-entropy (CE) method is a popular stochastic method for optimization due to its simplicity and effectiveness. Designed for rare-event simulations where the probability of a target event occurring is relatively small, the CE-method relies on enough objective function calls to accurately estimate the optimal parameters of the underlying distribution. Certain objective functions may be computationally expensive to evaluate, and the CE-method could potentially get stuck in local minima. This is compounded with the need to have an initial covariance wide enough to cover the design space of interest. We introduce novel variants of the CE-method to address these concerns. To mitigate expensive function calls, during optimization we use every sample to build a surrogate model to approximate the objective function. The surrogate model augments the belief of the objective function with less expensive evaluations. We use a Gaussian process for our surrogate model to incorporate uncertainty in the predictions which is especially helpful when dealing with sparse data. To address local minima convergence, we use Gaussian mixture models to encourage exploration of the design space. We experiment with evaluation scheduling techniques to reallocate true objective function calls earlier in the optimization when the covariance is the largest. To test our approach, we created a parameterized test objective function with many local minima and a single global minimum. Our test function can be adjusted to control the spread and distinction of the minima. Experiments were run to stress the cross-entropy method variants and results indicate that the surrogate model-based approach reduces local minima convergence using the same number of function evaluations.**

I. Introduction

The cross-entropy (CE) method is a probabilistic optimization approach that attempts to iteratively fit a distribution to elite samples from an initial input distribution [1], [2]. The goal is to estimate a rare-event probability by minimizing the *cross-entropy* between the two distributions [3]. The CE-method has gained popularity in part due to its simplicity in implementation and straightforward derivation. The technique uses *importance sampling* which introduces a proposal distribution over the rare-events to sample from then re-weights the posterior likelihood by the *likelihood ratio* of the true distribution over the proposal distribution.

There are a few key assumptions that make the CE-method work effectively. Through random sampling, the CE-method assumes that there are enough objective function evaluations to accurately represent the objective. This may not be a problem for simple applications, but can be an issue for computationally expensive objective functions. Another assumption is that the initial parameters of the input distribution are wide enough to cover the design space of interest. For the case with a multivariate Gaussian distribution, this corresponds to an appropriate mean and wide covariance. In rare-event simulations with many local minima, the CE-method can fail to find a global minima especially with sparse objective function evaluations.

This work aims to address the key assumptions of the CE-method. We introduce variants of the CE-method that use surrogate modeling to approximate the objective function, thus updating the belief of the underlying objective through estimation. As part of this approach, we introduce evaluation scheduling techniques to reallocate true objective function calls earlier in the optimization when we know the covariance will be large. The evaluation schedules can be based on a distribution (e.g., the Geometric distribution) or can be prescribed manually depending on the problem. We also use a Gaussian mixture model representation of the prior distribution as a method to explore competing local optima. While the use of Gaussian mixture models in the CE-method is not novel, we connect the use of mixture models and surrogate modeling in the CE-method. This connection uses each elite sample as the mean of a component distribution in the mixture, optimized through a subroutine call to the standard CE-method using the learned surrogate model. To test our approach, we introduce a parameterized test objective function called *sierra*. The sierra function is built from a multivariate Gaussian mixture model with many local minima and a single global minimum. Parameters for the sierra function allow control over both the spread and distinction of the minima. Lastly, we provide an analysis of the weak areas of the CE-method compared to our proposed variants.

II. Related Work

The cross-entropy method is popular in the fields of operations research, machine learning, and optimization [4], [5]. The combination of the cross-entropy method, surrogate modeling, and mixture models has been explored in other work [6]. The work in [6] proposed an adaptive grid approach to accelerate Gaussian-process-based surrogate modeling using mixture models as the prior in the cross-entropy method. They showed that a mixture model performs better than a single Gaussian when the objective function is multimodal. Our work differs in that we augment the "elite" samples both by an approximate surrogate model and by a subroutine call to the CE-method using the learned surrogate model. Other related work use Gaussian processes and a modified cross-entropy

Code available at https://github.com/mossr/CrossEntropyVariants.jl method for receding-horizon trajectory optimization [7]. Their cross-entropy method variant also incorporates the notion of exploration in the context of path finding applications. An approach based on *relative entropy*, described in section III-A, proposed a model-based stochastic search that seeks to minimize the relative entropy [8]. They also explore the use of a simple quadratic surrogate model to approximate the objective function. Prior work that relate cross-entropy-based adaptive importance sampling with Gaussian mixture models show that a mixture model require less objective function calls than a naïve Monte Carlo or standard unimodal cross-entropy-based importance sampling method [9], [10].

III. Background

This section provides necessary background on techniques used in this work. We provide introductions to cross-entropy and the cross-entropy method, surrogate modeling using Gaussian processes, and multivariate Gaussian mixture models.

A. Cross-Entropy

Before understanding the cross-entropy method, we first must understand the notion of *cross-entropy*. Cross-entropy is a metric used to measure the distance between two probability distributions, where the distance may not be symmetric [3]. The distance used to define cross-entropy is called the *Kullback-Leibler (KL) distance* or *KL divergence*. The KL distance is also called the *relative entropy*, and we can use this to derive the cross-entropy. Formally, for a random variable $\mathbf{X} = (X_1, \dots, X_n)$ with a support of $\mathcal{X}$, the KL distance between two continuous probability density functions f and g is defined to be:

$$\begin{aligned}\mathcal{D}(f,g) &= \mathbb{E}_f\left[\log\frac{f(\mathbf{X})}{g(\mathbf{X})}\right]\\ &= \int\limits_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})\log f(\mathbf{x})d\mathbf{x} - \int\limits_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})\log g(\mathbf{x})d\mathbf{x}\end{aligned}$$

We denote the expectation of some function with respect to a distribution f as $\mathbb{E}_f$. Minimizing the KL distance $\mathcal{D}$ between our true distribution f and our proposal distribution g parameterized by θ, is equivalent to choosing θ that minimizes the following, called the *cross-entropy*:

$$\begin{aligned}H(f,g) &= H(f) + \mathcal{D}(f,g)\\ &= -\mathbb{E}_f[\log g(\mathbf{X})] \qquad \text{(using KL distance)}\\ &= -\int\limits_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})\log g(\mathbf{x}\mid\theta)d\mathbf{x}\end{aligned}$$

where $H(f)$ denotes the entropy of the distribution f (where we conflate entropy and continuous entropy for convenience). This assumes that f and g share the support $\mathcal{X}$ and are continuous with respect to $\mathbf{x}$. The minimization problem then becomes:

$$\underset{\theta}{\text{minimize}} \quad -\int\limits_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})\log g(\mathbf{x}\mid\theta)d\mathbf{x} \tag{1}$$

Efficiently finding this minimum is the goal of the cross-entropy method algorithm.

B. Cross-Entropy Method

Using the definition of cross-entropy, intuitively the *cross-entropy method* (CEM or CE-method) aims to minimize the cross-entropy between the unknown true distribution f and a proposal distribution g parameterized by θ. This technique reformulates the minimization problem as a probability estimation problem, and uses adaptive importance sampling to estimate the unknown expectation [3]. The cross-entropy method has been applied in the context of both discrete and continuous optimization problems [2], [11].

The initial goal is to estimate the probability

$$\ell = P_\theta(S(\mathbf{x}) \geq \gamma)$$

where S can the thought of as an objective function of $\mathbf{x}$, and $\mathbf{x}$ follows a distribution defined by $g(\mathbf{x}\mid\theta)$. We want to find events where our objective function S is above some threshold γ. We can express this unknown probability as the expectation

$$\ell = \mathbb{E}_\theta[\mathbb{1}_{(S(\mathbf{x})\geq\gamma)}] \tag{2}$$

where $\mathbb{1}$ denotes the indicator function. A straightforward way to estimate eq. (2) can be done through Monte Carlo sampling. But for rare-event simulations where the probability of a target event occurring is relatively small, this estimate becomes inadequate. The challenge of the minimization in eq. (1) then becomes choosing the density function for the true distribution $f(\mathbf{x})$. Importance sampling tells us that the optimal importance sampling density can be reduced to

$$f^*(\mathbf{x}) = \frac{\mathbb{1}_{(S(\mathbf{x})\geq\gamma)}g(\mathbf{x}\mid\theta)}{\ell}$$

thus resulting in the optimization problem:

$$\begin{aligned}\theta_g^* &= \arg\min_{\theta_g} -\int\limits_{\mathbf{x}\in\mathcal{X}} f^*(\mathbf{x})\log g(\mathbf{x}\mid\theta_g)d\mathbf{x}\\ &= \arg\min_{\theta_g} -\int\limits_{\mathbf{x}\in\mathcal{X}} \frac{\mathbb{1}_{(S(\mathbf{x})\geq\gamma)}g(\mathbf{x}\mid\theta)}{\ell}\log g(\mathbf{x}\mid\theta_g)d\mathbf{x}\end{aligned}$$

Note that since we assume f and g belong to the same family of distributions, we get that $f(\mathbf{x}) = g(\mathbf{x}\mid\theta_g)$. Now notice that ℓ is independent of θ_g, thus we can drop ℓ and get the final optimization problem of:

$$\begin{aligned}\theta_g^* &= \arg\min_{\theta_g} -\int\limits_{\mathbf{x}\in\mathcal{X}} \mathbb{1}_{(S(\mathbf{x})\geq\gamma)}g(\mathbf{x}\mid\theta)\log g(\mathbf{x}\mid\theta_g)d\mathbf{x} \qquad (3)\\ &= \arg\min_{\theta_g} -\mathbb{E}_\theta[\mathbb{1}_{(S(\mathbf{x})\geq\gamma)}\log g(\mathbf{x}\mid\theta_g)]\end{aligned}$$

The CE-method uses a multi-level algorithm to estimate θ_g^* iteratively. The parameter θ_k at iteration k is used to find new parameters $\theta_{k'}$ at the next iteration k'. The threshold γ_k becomes smaller that its initial value, thus artificially making events *less rare* under $\mathbf{X} \sim g(\mathbf{x}\mid\theta_k)$.

In practice, the CE-method algorithm requires the user to specify a number of *elite* samples m_{elite} which are used when fitting the new parameters for iteration k'. Conveniently, if our distribution g belongs to the *natural exponential family* then the optimal parameters can be found analytically [5]. For a multivariate Gaussian distribution parameterized by $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, the optimal parameters for the next iteration k' correspond to the maximum likelihood estimate (MLE):

$$\boldsymbol{\mu}_{k'} = \frac{1}{m_{\text{elite}}} \sum_{i=1}^{m_{\text{elite}}} \mathbf{x}_i$$
$$\boldsymbol{\Sigma}_{k'} = \frac{1}{m_{\text{elite}}} \sum_{i=1}^{m_{\text{elite}}} (\mathbf{x}_i - \boldsymbol{\mu}_{k'})(\mathbf{x}_i - \boldsymbol{\mu}_{k'})^\top$$

The cross-entropy method algorithm is shown in algorithm 1. For an objective function S and input distribution g, the CE-method algorithm will run for k_{max} iterations. At each iteration, m inputs are sampled from g and evaluated using the objective function S. The sampled inputs are denoted by $\mathbf{X}$ and the evaluated values are denoted by $\mathbf{Y}$. Next, the top m_{elite} samples are stored in the elite set $\mathbf{e}$, and the distribution g is fit to the elites. This process is repeated for k_{max} iterations and the resulting parameters $\boldsymbol{\theta}_{k_{\text{max}}}$ are returned. Note that a variety of input distributions for g are supported, but we focus on the multivariate Gaussian distribution and the Gaussian mixture model in this work.

Algorithm 1 Cross-entropy method.

```
function CrossEntropyMethod($S, g, m, m_{\text{elite}}, k_{\text{max}}$)
    for $k \in [1, \ldots, k_{\text{max}}]$ do
        $\mathbf{X} \sim g(\,\cdot \mid \boldsymbol{\theta}_k)$ where $\mathbf{X} \in \mathbb{R}^m$
        $\mathbf{Y} \leftarrow S(\mathbf{x})$ for $\mathbf{x} \in \mathbf{X}$
        $\mathbf{e} \leftarrow$ store top $m_{\text{elite}}$ from $\mathbf{Y}$
        $\boldsymbol{\theta}_{k'} \leftarrow \textsc{Fit}(g(\,\cdot \mid \boldsymbol{\theta}_k), \mathbf{e})$
    return $g(\,\cdot \mid \boldsymbol{\theta}_{k_{\text{max}}})$
```

C. Mixture Models

A standard Gaussian distribution is *unimodal* and can have trouble generalizing over data that is *multimodal*. A *mixture model* is a weighted mixture of component distributions used to represent continuous multimodal distributions [4]. Formally, a Gaussian mixture model (GMM) is defined by its parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ and associated weights $\mathbf{w}$ where $\sum_{i=1}^{n} w_i = 1$. We denote that a random variable $\mathbf{X}$ is distributed according to a mixture model as $\mathbf{X} \sim \text{Mixture}(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{w})$. The probability density of the GMM then becomes:

$$P(\mathbf{X} = \mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{w}) = \sum_{i=1}^{n} w_i \, \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

To fit the parameters of a Gaussian mixture model, it is well known that the *expectation-maximization (EM)* algorithm can be used [12], [13]. The EM algorithm seeks to find the maximum likelihood estimate of the hidden variable H using the observed data defined by E. Intuitively, the algorithm alternates between an expectation step (E-step) and a maximization step (M-step) to guarantee convergence to a local minima. A simplified EM algorithm is provide in algorithm 2 for reference and we refer to [12], [13] for further reading.

Algorithm 2 Expectation-maximization.

```
function ExpectationMaximization($H, E, \boldsymbol{\theta}$)
    for E-step do
        Compute $Q(h) = P(H = h \mid E = e, \boldsymbol{\theta})$ for each $h$
        Create weighted points: $(h, e)$ with weight $Q(h)$
    for M-step do
        Compute $\hat{\boldsymbol{\theta}}_{\text{MLE}}$
    Repeat until convergence.
    return $\hat{\boldsymbol{\theta}}_{\text{MLE}}$
```

D. Surrogate Models

In the context of optimization, a surrogate model $\hat{S}$ is used to estimate the true objective function and provide less expensive evaluations. Surrogate models are a popular approach and have been used to evaluate rare-event probabilities in computationally expensive systems [14], [15]. The simplest example of a surrogate model is linear regression. In this work, we focus on the *Gaussian process* surrogate model. A Gaussian process (GP) is a distribution over functions that predicts the underlying objective function S and captures the uncertainty of the prediction using a probability distribution [5]. This means a GP can be sampled to generate random functions, which can then be fit to our given data $\mathbf{X}$. A Gaussian process is parameterized by a mean function $\mathbf{m}(\mathbf{X})$ and kernel function $\mathbf{K}(\mathbf{X}, \mathbf{X})$, which captures the relationship between data points as covariance values. We denote a Gaussian process that produces estimates $\hat{\mathbf{y}}$ as:

$$\begin{aligned}\hat{\mathbf{y}} &\sim \mathcal{N}\left(\mathbf{m}(\mathbf{X}), \mathbf{K}(\mathbf{X}, \mathbf{X})\right) \\ &= \left[\hat{S}(\mathbf{x}_1), \ldots, \hat{S}(\mathbf{x}_n)\right]\end{aligned}$$

where

$$\mathbf{m}(\mathbf{X}) = \left[m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n)\right]$$
$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

We use the commonly used zero-mean function $m(\mathbf{x}_i) = \mathbf{0}$. For the kernel function $k(\mathbf{x}_i, \mathbf{x}_i)$, we use the squared exponential kernel with variance σ^2 and characteristic scale-length ℓ, where larger ℓ values increase the correlation between successive data points, thus smoothing out the generated functions. The squared exponential kernel is defined as:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{(\mathbf{x} - \mathbf{x}')^\top (\mathbf{x} - \mathbf{x}')}{2\ell^2}\right)$$

We refer to [5] for a detailed overview of Gaussian processes and different kernel functions.

IV. Algorithms

We can now describe the cross-entropy method variants introduced in this work. This section will first cover the main algorithm introduced, the cross-entropy surrogate method

(CE-surrogate). Then we introduce a modification to the CE-surrogate method, namely the cross-entropy mixture method (CE-mixture). Lastly, we describe various evaluation schedules for redistributing objective function calls over the iterations.

A. *Cross-Entropy Surrogate Method*

The main CE-method variant we introduce is the cross-entropy surrogate method (CE-surrogate). The CE-surrogate method is a superset of the CE-method, where the differences lie in the evaluation scheduling and modeling of the elite set using a surrogate model. The goal of the CE-surrogate algorithm is to address the shortcomings of the CE-method when the number of objective function calls is sparse and the underlying objective function S has multiple local minima.

The CE-surrogate algorithm is shown in algorithm 3. It takes as input the objective function S, the distribution $\mathbf{M}$ parameterized by $\boldsymbol{\theta}$, the number of samples m, the number of elite samples m_{elite}, and the maximum iterations $k_{\max}$. For each iteration k, the number of samples m are redistributed through a call to EVALUATIONSCHEDULE, where m controls the number of true objective function evaluations of S. Then, the algorithm samples from $\mathbf{M}$ parameterized by the current $\boldsymbol{\theta}_k$ given the adjusted number of samples m. For each sample in $\mathbf{X}$, the objective function S is evaluated and the results are stored in $\mathbf{Y}$. The top m_{elite} evaluations from $\mathbf{Y}$ are stored in $\mathbf{e}$. Using all of the current function evaluations $\mathbf{Y}$ from sampled inputs $\mathbf{X}$, a modeled elite set $\mathbf{E}$ is created to augment the sparse information provided by a low number of true objective function evaluations. Finally, the distribution $\mathbf{M}$ is fit to the elite set $\mathbf{E}$ and the distribution with the final parameters $\boldsymbol{\theta}_{k_{\max}}$ is returned.

Algorithm 3 Cross-entropy surrogate method.

```
function CE-SURROGATE($S$, $\mathbf{M}$, $m$, $m_{\text{elite}}$, $k_{\max}$)
    for $k \in [1, \ldots, k_{\max}]$ do
        $m, m_{\text{elite}} \leftarrow$ EVALUATIONSCHEDULE($k, k_{\max}$)
        $\mathbf{X} \sim \mathbf{M}(\cdot \mid \boldsymbol{\theta}_k)$ where $\mathbf{X} \in \mathbb{R}^m$
        $\mathbf{Y} \leftarrow S(\mathbf{x})$ for $\mathbf{x} \in \mathbf{X}$
        $\mathbf{e} \leftarrow$ store top $m_{\text{elite}}$ from $\mathbf{Y}$
        $\mathbf{E} \leftarrow$ MODELELITESET($\mathbf{X}, \mathbf{Y}, \mathbf{M}, \mathbf{e}, m, m_{\text{elite}}$)
        $\boldsymbol{\theta}_{k'} \leftarrow$ FIT($\mathbf{M}(\cdot \mid \boldsymbol{\theta}_k), \mathbf{E}$)
    return $\mathbf{M}(\cdot \mid \boldsymbol{\theta}_{k_{\max}})$
```

The main difference between the standard CE-method and the CE-surrogate variant lies in the call to MODELELITESET. The motivation is to use *all* of the already evaluated objective function values $\mathbf{Y}$ from a set of sampled inputs $\mathbf{X}$. This way the expensive function evaluations—otherwise discarded—can be used to build a surrogate model of the underlying objective function. First, a surrogate model $\hat{S}$ is constructed from the samples $\mathbf{X}$ and true objective function values $\mathbf{Y}$. We used a Gaussian process with a specified kernel and optimizer, but other surrogate modeling techniques such as regression with basis functions can be used. We chose a Gaussian process because it incorporates probabilistic uncertainty in the predictions, which may more accurately represent our objective function, or at least be sensitive to over-fitting to sparse data. Now we have an approximated objective function $\hat{S}$ that we can inexpensively call. We sample $10m$ values from the distribution $\mathbf{M}$ and evaluate them using the surrogate model. We then store the top $10m_{\text{elite}}$ values from the estimates $\hat{\mathbf{Y}}_{\text{m}}$. We call these estimated elite values $\mathbf{e}_{\text{model}}$ the *model-elites*. The surrogate model is then passed to SUBELITESET, which returns more estimates for elite values. Finally, the elite set $\mathbf{E}$ is built from the true-elites $\mathbf{e}$, the model-elites $\mathbf{e}_{\text{model}}$, and the subcomponent-elites $\mathbf{e}_{\text{sub}}$. The resulting concatenated elite set $\mathbf{E}$ is returned.

Algorithm 4 Modeling elite set.

```
function MODELELITESET($\mathbf{X}, \mathbf{Y}, \mathbf{M}, \mathbf{e}, m, m_{\text{elite}}$)
    $\hat{S} \leftarrow$ GAUSSIANPROCESS($\mathbf{X}, \mathbf{Y}$, kernel, optimizer)
    $\mathbf{X}_{\text{m}} \sim \mathbf{M}(\cdot \mid \boldsymbol{\theta}_k)$ where $\mathbf{X}_{\text{m}} \in \mathbb{R}^{10m}$
    $\hat{\mathbf{Y}}_{\text{m}} \leftarrow \hat{S}(\mathbf{x}_{\text{m}})$ for $\mathbf{x}_{\text{m}} \in \mathbf{X}_{\text{m}}$
    $\mathbf{e}_{\text{model}} \leftarrow$ store top $10m_{\text{elite}}$ from $\hat{\mathbf{Y}}_{\text{m}}$
    $\mathbf{e}_{\text{sub}} \leftarrow$ SUBELITESET($\hat{S}, \mathbf{M}, \mathbf{e}$)
    $\mathbf{E} \leftarrow \{\mathbf{e}\} \cup \{\mathbf{e}_{\text{model}}\} \cup \{\mathbf{e}_{\text{sub}}\}$        ▷ elite set
    return $\mathbf{E}$
```

To encourage exploration of promising areas of the design space, the algorithm SUBELITESET focuses on the already marked true-elites $\mathbf{e}$. Each elite $e_x \in \mathbf{e}$ is used as the mean of a new multivariate Gaussian distribution with covariance inherited from the distribution $\mathbf{M}$. The collection of subcomponent distributions is stored in $\mathbf{m}$. The idea is to use the information given to us by the true-elites to emphasize areas of the design space that look promising. For each distribution $\mathbf{m}_i \in \mathbf{m}$ we run a subroutine call to the standard CE-method to fit the distribution $\mathbf{m}_i$ using the surrogate model $\hat{S}$. Then the best objective function value is added to the subcomponent-elite set $\mathbf{e}_{\text{sub}}$, and after iterating the full set is returned. Note that we use θ_{CE} to denote the parameters for the CE-method algorithm. In our case, we recommend using a small $k_{\max}$ of around 2 so the subcomponent-elites do not over-fit to the surrogate model but have enough CE-method iterations to tend towards optimal.

Algorithm 5 Subcomponent elite set.

```
function SUBELITESET($\hat{S}, \mathbf{M}, \mathbf{e}$)
    $\mathbf{e}_{\text{sub}} \leftarrow \emptyset$
    $\mathbf{m} \leftarrow \{e_x \in \mathbf{e} \mid \mathcal{N}(e_x, \mathbf{M}.\Sigma)\}$
    for $\mathbf{m}_i \in \mathbf{m}$ do
        $\mathbf{m}_i \leftarrow$ CROSSENTROPYMETHOD($\hat{S}, \mathbf{m}_i \mid \theta_{\text{CE}}$)
        $\mathbf{e}_{\text{sub}} \leftarrow \{\mathbf{e}_{\text{sub}}\} \cup \{$BEST($\mathbf{m}_i$)$\}$
    return $\mathbf{e}_{\text{sub}}$
```

B. *Cross-Entropy Mixture Method*

We refer to the variant of our CE-surrogate method that takes an input *mixture model* $\mathbf{M}$ as the cross-entropy mixture method (CE-mixture). The CE-mixture algorithm is identical to the CE-surrogate algorithm, but calls a custom FIT function to fit a mixture model to the elite set $\mathbf{E}$. The input distribution

$\mathbf{M}$ is cast to a mixture model using the subcomponent distributions $\mathbf{m}$ as the components of the mixture. We use the default uniform weighting for each mixture component. The mixture model $\mathbf{M}$ is then fit using the expectation-maximization algorithm shown in algorithm 2, and the resulting distribution is returned. The idea is to use the distributions in $\mathbf{m}$ that are centered around each true-elite as the components of the casted mixture model. Therefore, we would expect better performance of the CE-mixture method when the objective function has many competing local minima. Results in section V-C aim to show this behavior.

Algorithm 6 Fitting mixture models (used by CE-mixture).

```
function Fit($\mathbf{M}, \mathbf{m}, \mathbf{E}$)
    $\mathbf{M} \leftarrow \text{Mixture}(\mathbf{m})$
    $\hat{\boldsymbol{\theta}} \leftarrow$ ExpectationMaximization($\mathbf{M}, \mathbf{E}$)
    return $\mathbf{M}(\,\cdot \mid \hat{\boldsymbol{\theta}})$
```

C. Evaluation Scheduling

Given the nature of the CE-method, we expect the covariance to shrink over time, thus resulting in a solution with higher confidence. Yet if each iteration is given the same number of objective function evaluations m, there is the potential for elite samples from early iterations dominating the convergence. Therefore, we would like to redistribute the objective function evaluations throughout the iterations to use more truth information early in the process. We call these heuristics *evaluation schedules*. One way to achieve this is to reallocate the evaluations according to a Geometric distribution. Evaluation schedules can also be ad-hoc and manually prescribed based on the current iteration.

We provide the evaluation schedule we use that follows a Geometric distribution with parameter p in algorithm 7. We denote $G \sim \text{Geo}(p)$ to be a random variable that follows a truncated Geometric distribution with the probability mass function $p_G(k) = p(1-p)^k$ for $k \in \{0, 1, 2, \ldots, k_{\text{max}}\}$. Note the use of the integer rounding function (e.g., $\lfloor x \rceil$), which we later have to compensate for towards the final iterations. Results in section V-C compare values of p that control the redistribution of evaluations.

Algorithm 7 Evaluation schedule using a Geometric distr.

```
function EvaluationSchedule($k, k_{\text{max}}$)
    $G \sim \text{Geo}(p)$
    $N_{\text{max}} \leftarrow k_{\text{max}} \cdot m$
    $m \leftarrow \lfloor N_{\text{max}} \cdot p_G(k) \rceil$
    if $k = k_{\text{max}}$ then
        $s \leftarrow \sum_{i=1}^{k_{\text{max}}-1} \lfloor N_{\text{max}} \cdot p_G(i) \rceil$
        $m \leftarrow \min(N_{\text{max}} - s, N_{\text{max}} - m)$
    $m_{\text{elite}} \leftarrow \min(m_{\text{elite}}, m)$
    return $(m, m_{\text{elite}})$
```

V. Experiments

In this section, we detail the experiments we ran to compare the CE-method variants and evaluation schedules. We first introduce a test objective function we created to stress the issue of converging to local minima. We then describe the experimental setup for each of our experiments and provide an analysis and results.

A. Test Objective Function Generation

To stress the cross-entropy method and its variants, we created a test objective function called *sierra* that is generated from a mixture model comprised of 49 multivariate Gaussian distributions. We chose this construction so that we can use the negative peeks of the component distributions as local minima and can force a global minimum centered at our desired $\tilde{\boldsymbol{\mu}}$. The construction of the sierra test function can be controlled by parameters that define the spread of the local minima. We first start with the center defined by a mean vector $\tilde{\boldsymbol{\mu}}$ and we use a common covariance $\tilde{\boldsymbol{\Sigma}}$:

$$\tilde{\boldsymbol{\mu}} = [\mu_1, \mu_2], \quad \tilde{\boldsymbol{\Sigma}} = \begin{bmatrix} \sigma & 0 \\ 0 & \sigma \end{bmatrix}$$

Next, we use the parameter δ that controls the clustered distance between symmetric points:

$$\mathbf{G} = \big\{[+\delta, +\delta], [+\delta, -\delta], [-\delta, +\delta], [-\delta, -\delta]\big\}$$

We chose points $\mathbf{P}$ to fan out the clustered minima relative to the center defined by $\tilde{\boldsymbol{\mu}}$:

$$\mathbf{P} = \big\{[0,0], [1,1], [2,0], [3,1], [0,2], [1,3]\big\}$$

The vector $\mathbf{s}$ is used to control the $\pm$ distance to create an 's' shape comprised of minima, using the standard deviation σ: $\mathbf{s} = \big[+\sigma, -\sigma\big]$. We set the following default parameters: standard deviation $\sigma = 3$, spread rate $\eta = 6$, and cluster distance $\delta = 2$. We can also control if the local minima clusters "decay", thus making those local minima less distinct (where decay $\in \{0, 1\}$). The parameters that define the sierra function are collected into $\boldsymbol{\theta} = \langle \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}, \mathbf{G}, \mathbf{P}, \mathbf{s} \rangle$. Using these parameters, we can define the mixture model used by the sierra function as:

$$\mathbf{M}_{\mathcal{S}} \sim \text{Mixture}\left(\left\{\boldsymbol{\theta} \;\middle|\; \mathcal{N}\left(\mathbf{g} + s\mathbf{p}_i + \tilde{\boldsymbol{\mu}},\; \tilde{\boldsymbol{\Sigma}} \cdot i^{\text{decay}}/\eta\right)\right\}\right)$$
$$\text{for } (\mathbf{g}, \mathbf{p}_i, s) \in (\mathbf{G}, \mathbf{P}, \mathbf{s})$$

We add a final component to be our global minimum centered at $\tilde{\boldsymbol{\mu}}$ and with a covariance scaled by $\sigma\eta$. Namely, the global minimum is $\mathbf{x}^* = \mathbb{E}[\mathcal{N}(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}/(\sigma\eta))] = \tilde{\boldsymbol{\mu}}$. We can now use this constant mixture model with 49 components and define the sierra objective function $\mathcal{S}(\mathbf{x})$ to be the negative probability density of the mixture at input $\mathbf{x}$ with uniform weights:

$$\mathcal{S}(\mathbf{x}) = -P(\mathbf{M}_{\mathcal{S}} = \mathbf{x}) = -\frac{1}{|\mathbf{M}_{\mathcal{S}}|}\sum_{j=1}^{n} \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

An example of six different objective functions generated using the sierra function are shown in fig. 1, sweeping over the spread rate η, with and without decay.

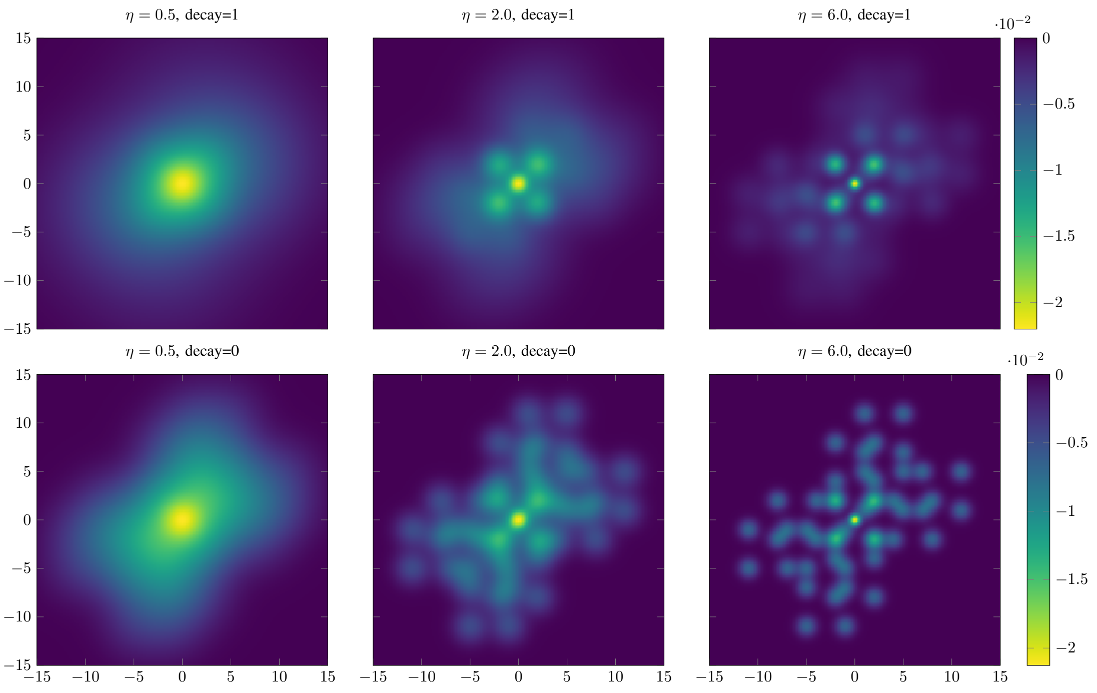


Fig. 1. Example test objective functions generated using the sierra function.

B. Experimental Setup

Experiments were run to stress a variety of behaviors of each CE-method variant. The experiments are split into two categories: algorithmic and scheduling. The algorithmic category aims to compare features of each CE-method variant while holding common parameters constant (for a better comparison). While the scheduling category experiments with evaluation scheduling heuristics.

Because the algorithms are stochastic, we run each experiment with 50 different random number generator seed values. To evaluate the performance of the algorithms in their respective experiments, we define three metrics. First, we define the average "optimal" value $\bar{b}_v$ to be the average of the best so-far objective function value (termed "optimal" in the context of each algorithm). Again, we emphasize that we average over the 50 seed values to gather meaningful statistics. Another metric we monitor is the average distance to the true global optimal $\bar{b}_d = \|\mathbf{b}_\mathbf{x} - \mathbf{x}^*\|$, where $\mathbf{b}_\mathbf{x}$ denotes the $\mathbf{x}$-value associated with the "optimal". We make the distinction between these metrics to show both "closeness" in *value* to the global minimum and "closeness" in the *design space* to the global minimum. Our final metric looks at the average runtime of each algorithm, noting that our goal is to off-load computationally expensive objective function calls to the surrogate model.

For all of the experiments, we use a common setting of the following parameters for the sierra test function (shown in the top-right plot in fig. 1):

$$(\tilde{\boldsymbol{\mu}} = [0,0],\ \sigma = 3,\ \delta = 2,\ \eta = 6,\ \text{decay} = 1)$$

1) Algorithmic Experiments: We run three separate algorithmic experiments, each to test a specific feature. For our first algorithmic experiment (1A), we want to test each algorithm when the user-defined mean is centered at the global minimum and the covariance is arbitrarily wide enough to cover the design space. Let $\mathbf{M}$ be a distribution parameterized by $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and for experiment (1A) we set the following:

$$\boldsymbol{\mu}^{(1A)} = [0,0] \qquad \boldsymbol{\Sigma}^{(1A)} = \begin{bmatrix} 200 & 0 \\ 0 & 200 \end{bmatrix}$$

For our second algorithmic experiment (1B), we test a mean that is far off-centered with a wider covariance:

$$\boldsymbol{\mu}^{(1B)} = [-50,-50] \qquad \boldsymbol{\Sigma}^{(1B)} = \begin{bmatrix} 2000 & 0 \\ 0 & 2000 \end{bmatrix}$$

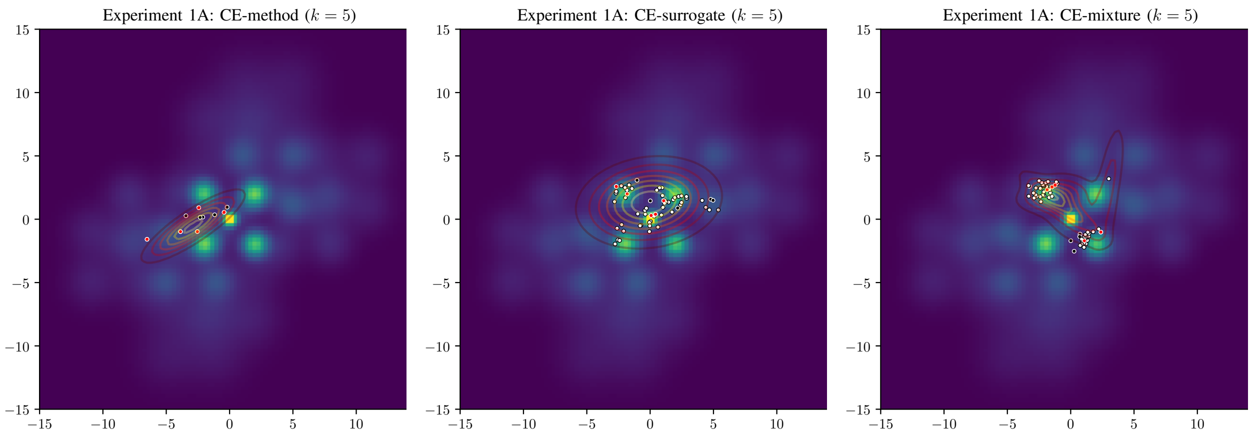
This experiment is used to test the "exploration" of the CE-method variants introduced in this work. In experiments (1A) and (1B), we set the following common parameters across each CE-method variant:

$$(k_{\text{max}} = 10,\ m = 10,\ m_{\text{elite}} = 5)^{(1A,1B)}$$

This results in $m \cdot k_{\text{max}} = 100$ objective function evaluations, which we define to be *relatively* low.

For our third algorithmic experiment (1C), we want to test how each variant responds to an extremely low number of function evaluations. This sparse experiment sets the common CE-method parameters to:

$$(k_{\text{max}} = 10,\ m = 5,\ m_{\text{elite}} = 3)^{(1C)}$$



(a) The cross-entropy method. (b) The cross-entropy surrogate method. (c) The cross-entropy mixture method.

Fig. 2. Iteration $k = 5$ illustrated for each algorithm. The covariance is shown by the contours.

This results in $m \cdot k_{\max} = 50$ objective function evaluations, which we defined to be *extremely* low. We use the same mean and covariance defined for experiment (1A):

$$\boldsymbol{\mu}^{(1C)} = [0, 0] \qquad \boldsymbol{\Sigma}^{(1C)} = \begin{bmatrix} 200 & 0 \\ 0 & 200 \end{bmatrix}$$

2) Scheduling Experiments: In our final experiment (2), we test the evaluation scheduling heuristics which are based on the Geometric distribution. We sweep over the parameter p that determines the Geometric distribution which controls the redistribution of objective function evaluations. In this experiment, we compare the CE-surrogate methods using the same setup as experiment (1B), namely the far off-centered mean. We chose this setup to analyze exploration schemes when given very little information about the true objective function.

C. Results and Analysis

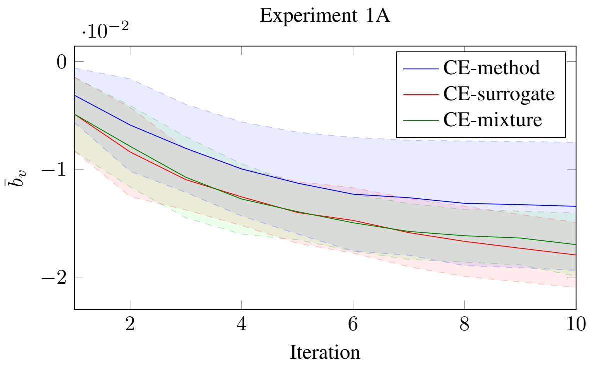


Fig. 3. Average optimal value for experiment (1A) when the initial mean is centered at the global minimum and the covariance sufficiently covers the design space.

Figure 3 shows the average value of the current optimal $\bar{b}_v$ for the three algorithms for experiment (1A). One standard deviation is plotted in the shaded region. Notice that the standard CE-method converges to a local minima before $k_{\max}$ is reached. Both CE-surrogate method and CE-mixture stay below the standard CE-method curve, highlighting the mitigation of convergence to local minima. Minor differences can be seen between CE-surrogate and CE-mixture, differing slightly towards the tail in favor of CE-surrogate. The average runtime of the algorithms along with the performance metrics are shown together for each experiment in table I.

TABLE I
EXPERIMENTAL RESULTS.

Exper.	Algorithm	Runtime	$\bar{b}_v$	$\bar{b}_d$
1A	CE-method	**0.029** s	-0.0134	23.48
	CE-surrogate	1.47 s	$\mathbf{-0.0179}$	**12.23**
	CE-mixture	9.17 s	-0.0169	16.87
1B	CE-method	**0.046** s	-0.0032	138.87
	CE-surrogate	11.82 s	$\mathbf{-0.0156}$	**18.24**
	CE-mixture	28.10 s	-0.0146	33.30
1C	CE-method	**0.052** s	-0.0065	43.14
	CE-surrogate	0.474 s	$\mathbf{-0.0156}$	**17.23**
	CE-mixture	2.57 s	-0.0146	22.17
2	CE-surrogate, Uniform	—	$\mathbf{-0.0193}$	**8.53**
	CE-surrogate, Geo(0.1)	—	-0.0115	25.35
	CE-surrogate, Geo(0.2)	—	-0.0099	27.59
	CE-surrogate, Geo(0.3)	—	-0.0089	30.88
			$-0.0220 \approx \mathbf{x}^*$	

An apparent benefit of the standard CE-method is in its simplicity and speed. As shown in table I, the CE-method is the fastest approach by about 2-3 orders of magnitude compared to CE-surrogate and CE-mixture. The CE-mixture method is notably the slowest approach. Although the runtime is also based on the objective function being tested, recall that we are using the same number of true objective function calls in each algorithm, and the metrics we are concerned with in optimization are to minimize $\bar{b}_v$ and $\bar{b}_d$. We can see that the CE-surrogate method consistently out performs the other methods. Surprisingly, a uniform evaluation schedule performs the best even in the sparse scenario where the initial mean is far away from the global optimal.

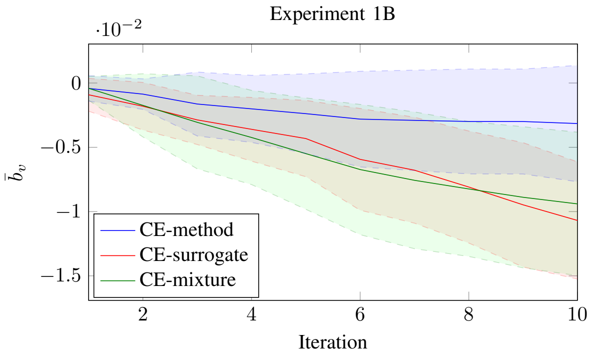


Fig. 4. Average optimal value for experiment (1B) when the initial mean is far from the global minimum with a wide covariance.

When the initial mean of the input distribution is placed far away from the global optimal, the CE-method tends to converge prematurely as shown in fig. 4. This scenario is illustrated in fig. 5. We can see that both CE-surrogate and CE-mixture perform well in this case.

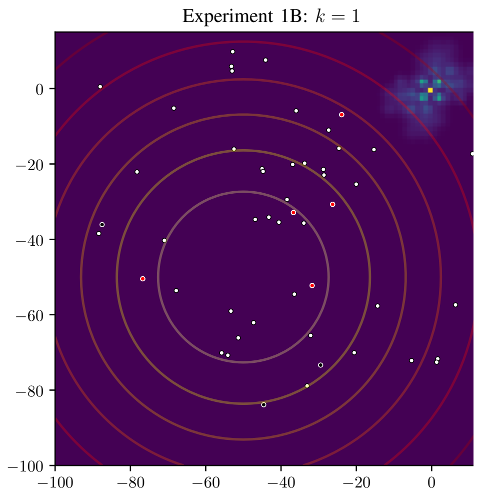


Fig. 5. First iteration of the scenario in experiment (1B) where the initial distribution is far away form the global optimal. The red dots indicate the true-elites, the black dots with white outlines indicate the "non-elites" evaluated from the true objective function, and the white dots with black outlines indicate the samples evaluated using the surrogate model.

Given the same centered mean as before, when we restrict the number of objective function calls even further to just 50 we see interesting behavior. Notice that the results of experiment (1C) shown in fig. 6 follow a curve closer to the far away mean from experiment (1B) than from the same setup as experiment (1A). Also notice that the CE-surrogate results cap out at iteration 9 due to the evaluation schedule front-loading the objective function calls, thus leaving none for the

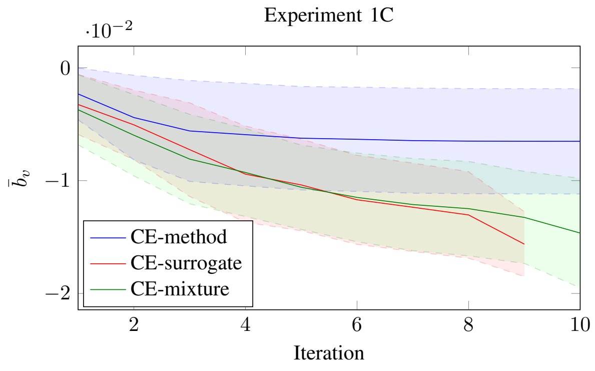


Fig. 6. Average optimal value for experiment (1C) when we restrict the number of objective function calls.

final iteration (while still maintaining the same total number of evaluations of 50).

VI. Conclusion

We presented variants of the popular cross-entropy method for optimization of objective functions with multiple local minima. Using a Gaussian processes-based surrogate model, we can use the same number of true objective function evaluations and achieve better performance than the standard CE-method on average. We also explored the use of a Gaussian mixture model to help find global minimum in multimodal objective functions. We introduce a parameterized test objective function with a controllable global minimum and spread of local minima. Using this test function, we showed that the CE-surrogate algorithm achieves the best performance relative to the standard CE-method, each using the same number of true objective function evaluations.

References

[1] R. Rubinstein and D. Kroese, *The cross-entropy method: A unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer, 2004.

[2] R. Rubinstein, "The cross-entropy method for combinatorial and continuous optimization," *Methodology and Computing in Applied Probability*, vol. 1, no. 2, pp. 127–190, 1999.

[3] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *ANNALS OF OPERATIONS RESEARCH*, vol. 134, no. 1, pp. 19–67, 2005.

[4] M. J. Kochenderfer, *Decision Making Under Uncertainty: Theory and Application*. MIT Press, 2015.

[5] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for optimization*. MIT Press, 2019.

[6] R. Bardenet and B. Kégl, "Surrogating the surrogate: Accelerating gaussian-process-based global optimization with a mixture cross-entropy algorithm," in *International Conference on Machine Learning (ICML)*, 2010, pp. 55–62.

[7] Y. T. Tan, A. Kunapareddy, and M. Kobilarov, "Gaussian process adaptive sampling using the cross-entropy method for environmental sensing and monitoring," in *International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6220–6227.

[8] A. Abdolmaleki, R. Lioutikov, J. R. Peters, N. Lau, L. Pualo Reis, and G. Neumann, "Model-based relative entropy stochastic search," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 3537–3545.

[9] N. Kurtz and J. Song, "Cross-entropy-based adaptive importance sampling using gaussian mixture," *Structural Safety*, vol. 42, pp. 35–44, 2013.

[10] Z. Wang and J. Song, “Cross-entropy-based adaptive importance sampling using von Mises-Fisher mixture for high dimensional reliability analysis,” *Structural Safety*, vol. 59, pp. 42–52, 2016.

[11] D. P. Kroese, S. Porotsky, and R. Y. Rubinstein, “The cross-entropy method for continuous multi-extremal optimization,” *Methodology and Computing in Applied Probability*, vol. 8, no. 3, pp. 383–407, 2006.

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[13] M. Aitkin and G. T. Wilson, “Mixture models, outliers, and the EM algorithm,” *Technometrics*, vol. 22, no. 3, pp. 325–331, 1980.

[14] J. Li and D. Xiu, “Evaluation of failure probability via surrogate models,” *Journal of Computational Physics*, vol. 229, no. 23, pp. 8966–8980, 2010.

[15] J. Li, J. Li, and D. Xiu, “An efficient surrogate-based method for computing rare failure probability,” *Journal of Computational Physics*, vol. 230, no. 24, pp. 8683–8697, 2011.