Journal of Quantitative Spectroscopy & Radiative Transfer 345 (2025) 109534

Contents lists available at ScienceDirect

Journal of Quantitative Spectroscopy and Radiative Transfer

journal homepage: www.elsevier.com/locate/jqsrt

An algorithm for solving the inverse problem in total internal reflection microscopy

Alexandru Doicu [a], Dmitry S. Efremenko [b,*], Christopher L. Wirth [c], Thomas Wriedt [d]

[a] Independent Researcher, 82110, Germering, Germany
[b] Institut für Fernerkundung (IMF), Deutsches Zentrum für Luft- und Raumfahrt (DLR), Oberpfaffenhofen, Germany
[c] Department of Chemical and Biomolecular Engineering, Case Western Reserve University, Cleveland, OH, USA
[d] Leibniz-Institut für Werkstofforientierte Technologien - IWT, University of Bremen, Bremen, Germany

ARTICLE INFO

Keywords:
Total internal reflection microscopy
Electromagnetic scattering
Evanescent waves
Inverse problems

ABSTRACT

The inverse problem encountered in total internal reflection microscopy is particularly challenging due to the complex relationship between the data and the state vector. To address the computational difficulties and compute all possible solutions to this inverse problem, we propose an algorithm for solving problems of the form $\mathbf{F}(\mathbf{x}) = \mathbf{y}$, where $\mathbf{F}$ is a nonlinear and continuous forward model, $\mathbf{x}$ is the state vector, and $\mathbf{y}$ is the data vector. The forward model is computationally expensive, lacks derivatives, and is treated as a black-box function, with no prior information available about the statistical properties of $\mathbf{x}$, aside from a box constraint. Additionally, the data vector $\mathbf{y}$ is affected by uncertainties, making the retrieval of accurate solutions more challenging. To address these issues, the algorithm incorporates a neural network-based surrogate model to reduce computational costs and manage the absence of derivatives. A whitening transformation is applied to handle isotropic noise, and the residual is minimized within simple bounds. The method is further enhanced by a multistart solver that generates starting points through a hitting set problem, improving the exploration of the solution space and increasing the likelihood of finding all solutions. The solver iterates over the absolute function tolerance to identify both global and local minima, minimizes a least-squares function while adhering to simple bounds on variables during the local search phase, and includes a double box stopping rule. Additionally, the algorithm employs various sampling techniques and optimization methods, with the covariance matrix estimated using either a standard analytical approach or a Monte Carlo analysis.

1. Introduction

Colloidal particles, which range from the nanometer to micrometer scale, interact with nearby surfaces–including adjacent particles and boundaries–through weak interactions at the kT scale [1,2]. These interactions play a significant role in processes such as particle deposition and the rheological properties of colloidal suspensions. Total Internal Reflection Microscopy (TIRM) was developed to explore the potential energy landscape and weak surface forces acting on particles near a boundary, allowing for the measurement of both conservative (e.g., electrostatic) and non-conservative (e.g., hydrodynamic) forces [3,4]. TIRM operates by analyzing the light scattered from a spherical particle situated close to a plane substrate illuminated by an evanescent wave, with the intensity of the scattered light varying exponentially with the distance between the particle and the surface.

Recent developments in Scattering Morphology Resolved TIRM (SMR-TIRM) enhance the technique's applicability to non-spherical particles by examining light scattering patterns to ascertain particle position and orientation [5–7]. It is important to note that while the separation distance alone can provide information about the potential energy profile for spherical particles, both separation distance and particle orientation are crucial for accurately tracking dynamics and calculating the potential energy landscape of non-spherical particles.

In the TIRM setup illustrated in Fig. 1, a combination of optical and fluidic components is utilized to investigate colloidal interactions. The microfluidic cell contains a sample suspended in an aqueous solution, positioned on a hemispherical glass prism using refractive index matching oil. A thick microscope slide forms the bottom of the cell, while a thinner cover slip is placed on top. A specialized coating is applied to the bottom slide to investigate specific material effects on surface interactions. The microscope objective captures the light scattered by particles on the glass surface, which is then analyzed by the CCD camera.

* Corresponding author.
E-mail address: dmitry.efremenko@dlr.de (D.S. Efremenko).

https://doi.org/10.1016/j.jqsrt.2025.109534
Received 19 November 2024; Received in revised form 27 February 2025; Accepted 22 May 2025
Available online 7 June 2025

In a TIRM experiment, we are faced with solving both a direct and an inverse problem (Fig. 2).

1. The direct problem has the following formulation: For a spheroidal particle with known (i) optical and geometrical parameters and (ii) Euler orientation angles (α, β), undergoing a TIRM experiment with evanescent wave illumination, determine the integrated signal I_M, the morphology orientation angle α_M, and the aspect ratio ε_M observed in an image, as functions of the separation distance d and the particle orientation angles (α, β).
2. The inverse problem has the following formulation: Given the measured morphology quantities $(I_M, \alpha_M, \varepsilon_M)$, solve the nonlinear equations $I_M = I_M(d, \alpha, \beta)$, $\alpha_M = \alpha_M(d, \alpha, \beta)$, and $\varepsilon_M = \varepsilon_M(d, \alpha, \beta)$ for the parameters (d, α, β) that are necessary for tracking dynamics.

The direct problem was analyzed in Ref. [8]. This research introduced a comprehensive model for light scattering that focuses on two primary elements: (i) the scattering of an axisymmetric particle oriented arbitrarily within a stratified medium, and (ii) the imaging of the resulting scattered light. Essential calculations within the model included (i) the responses of the layered medium to both incoming and scattered light, (ii) the construction of the interaction matrix, and (iii) the transition matrix for an isolated particle. The responses of the layer system were derived using plane wave expansions, which were subsequently transformed into regular spherical wave expansions. The transition matrix was obtained via the null-field method. To accommodate the arbitrary orientations of particles, we applied the addition theorem for vector spherical wave functions with appropriate coordinate rotations. The imaging of the scattered light was accomplished by computing (i) the scattered field on the Gaussian reference sphere of the collector lens, (ii) the transmitted field on the Gaussian reference sphere of the detector lens, and (iii) the focused field through the Debye diffraction integral and fast Fourier transform. Additionally, an image processing technique that involved contour extraction and ellipse fitting was utilized to enhance the accuracy of particle orientation reconstruction. Fig. 3 shows intensity distributions and fitted ellipses in some scattered field images.

This study examines the inverse problem in a TIRM experiment, which is part of a broader category of inverse problems requiring efficient solutions of a system of nonlinear equations.

2. The inversion algorithm

Consider an inverse problem that requires effective solutions to the equation

$$\mathbf{F}(\mathbf{x}) = \mathbf{y}, \tag{1}$$

where $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^m$ is a forward model representing a physical process, $\mathbf{x} \in \mathbb{R}^n$ is the state vector to be retrieved, and $\mathbf{y} \in \mathbb{R}^m$ is the data vector. The inverse problem has the following characteristics:

1. $\mathbf{F}$ is nonlinear and continuous.
2. $\mathbf{F}$ is computationally expensive to evaluate.
3. The derivatives of $\mathbf{F}$ are not available, meaning that $\mathbf{F}$ is a black-box function.
4. There is no prior information about the smoothness or statistical properties of the state vector $\mathbf{x}$, except that it belongs to the box $X = [a_1, b_1] \times [a_2, b_2] \ldots \times [a_n, b_n]$.
5. The data vector $\mathbf{y}$ is affected by uncertainties.

Such inverse problems arise in various scientific and engineering fields, often suffering from ill-posedness and non-uniqueness. The nonlinearity of the forward model, combined with the limited and uncertain nature of the data, can lead to multiple model configurations producing indistinguishable observations. For instance, in electrical impedance tomography, different conductivity distributions may yield identical boundary voltage measurements due to the ill-posed nature of the problem, compounded by measurement noise and electrode limitations [9]. Seismic inversion faces similar ambiguities, as different subsurface velocity structures can generate comparable seismic signals, particularly given the limited resolution and sparse data coverage [10]. In atmospheric remote sensing, the radiative transfer equation allows multiple atmospheric compositions to produce nearly identical radiance spectra,

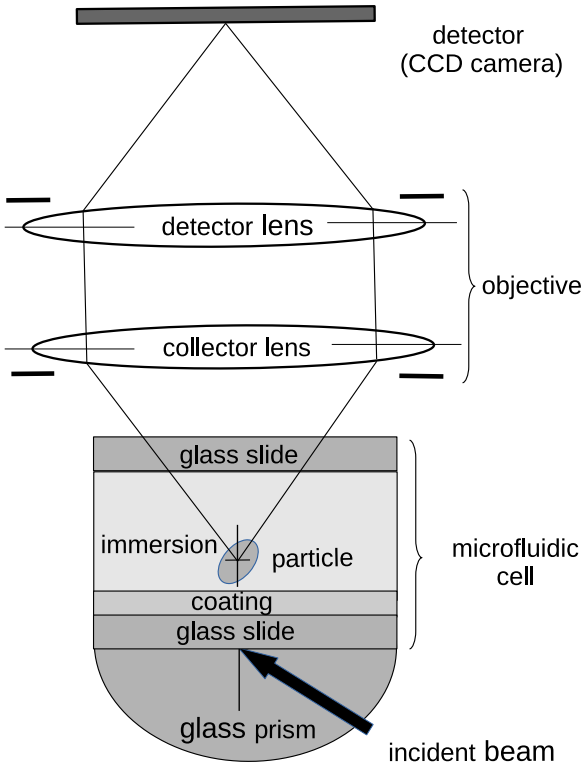

Fig. 1. Simplified TIRM setup.
Source: Adapted from Ref. [8]

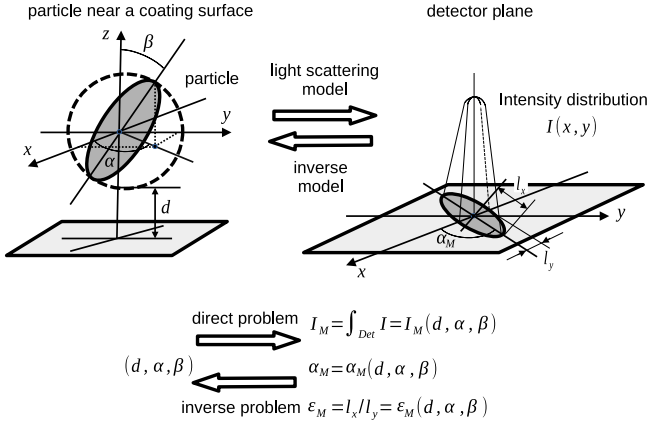

Fig. 2. Illustration of direct and inverse problems in a TIRM experiment. Direct problem: For a spheroidal particle with known orientation angles (α, β), determine the integrated signal I_M, the morphology orientation angle α_M, and the aspect ratio ε_M observed in an image, as functions of the separation distance d and the particle orientation angles (α, β). Inverse problem: Given the measured morphology quantities $(I_M, \alpha_M, \varepsilon_M)$, solve the nonlinear equations $I_M = I_M(d, \alpha, \beta)$, $\alpha_M = \alpha_M(d, \alpha, \beta)$, and $\varepsilon_M = \varepsilon_M(d, \alpha, \beta)$ for the parameters (d, α, β).

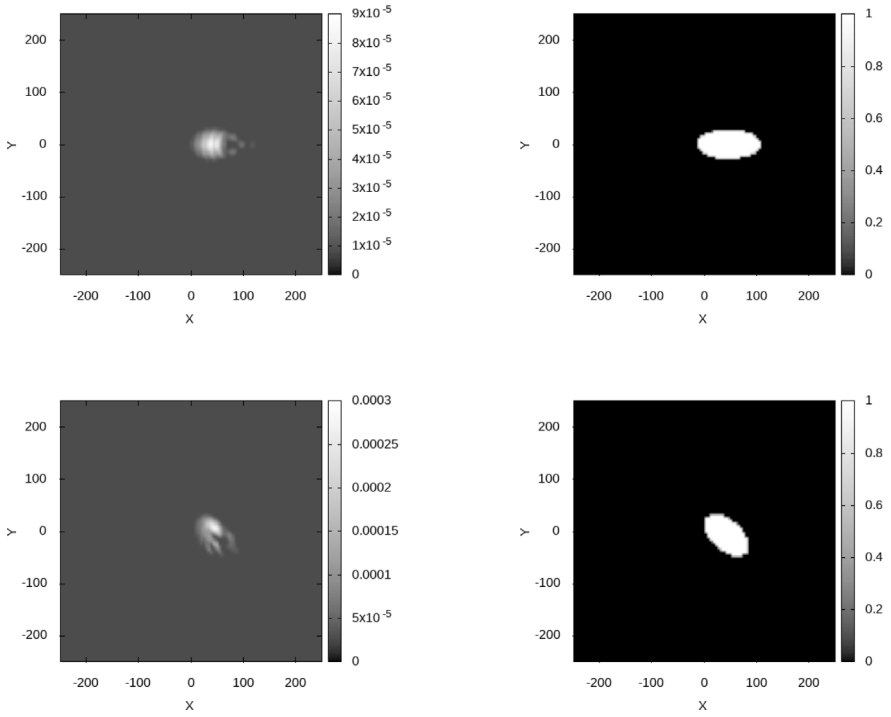

Fig. 3. Intensity distributions (left panels) and fitted ellipses (right panels). The results correspond to the azimuthal orientation angles $\alpha = 0°$ (upper panels), and 45° (lower panels).
Source: Adapted from Ref. [8].

creating uncertainties in the retrieval of gas concentrations, as well as cloud and aerosol properties [11]. X-ray computed tomography with limited-angle data suffers from structural ambiguities, where missing projections conceal critical features, leading to multiple plausible reconstructions [12]. Electromagnetic scattering inverse problems exhibit related challenges, as different combinations of particle size and refractive index can result in identical scattering patterns. For example, in the case of mononuclear cells modeled as coated spheres, multiple sets of cell diameter, nucleus-to-cell diameter ratio, and refractive indices of the cytoplasm and nucleus can produce indistinguishable angular scattering patterns [13]. Similarly, retrieving the size and refractive index of homogeneous spheres from the power Fourier spectrum of their scattering patterns, as measured by a scanning flow cytometer, leads to comparable ambiguities [14]. To address these challenges, we propose an algorithm that consists of the following components:

1. A neural network-based surrogate model that approximates the forward model, significantly reducing computational cost while accommodating the lack of derivative information.
2. A whitening transformation to recast the inverse problem in a form where noise can be assumed isotropic, facilitating the solution by minimizing the residual subject to simple variable bounds.
3. An advanced multistart solver designed to systematically explore the solution space and identify all solutions of the inverse problem, accounting for its potential non-uniqueness.

2.1. Neural network-based surrogate model

When approximating a multivariate function $\mathbf{F} : X \subset \mathbb{R}^n \to \mathbb{R}^m$ based on a dataset $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$, where $\mathbf{y}_i = \mathbf{F}(\mathbf{x}_i)$ and N is the number of samples, several surrogate models can be applied. Each model offers a way to estimate $\mathbf{F}$ based on data values $\mathbf{y}_i$ at sample points $\mathbf{x}_i$. Traditional approaches include orthonormal polynomials [15], radial basis functions [16], radial basis functions with centers [17], and kriging (Gaussian process regression) [18]. Recently, however, neural networks have emerged as the most widely used and effective approximation method. Unlike polynomial-based models, which struggle with high-dimensional problems due to the curse of dimensionality, or kriging, which becomes computationally expensive for large datasets, neural networks excel at capturing complex, nonlinear relationships with high accuracy [19–23]. Their ability to learn intricate patterns from data, generalize well across different domains, and efficiently scale with increasing data dimensions makes them superior to traditional surrogate models. An additional advantage is that the derivatives of a neural network model with respect to its inputs can be computed analytically. As a result, a neural network algorithm can not only provide an approximation of the quantities of interest but also offer an estimate of their derivatives with respect to the model inputs. Given these advantages, we focus our discussion on neural network-based approximation methods.

In machine learning, the goal is to approximate $\mathbf{F}(\mathbf{x})$ by a neural network model $\mathbf{f}(\mathbf{x}, \boldsymbol{\omega})$ characterized by a set of parameters $\boldsymbol{\omega}$ using the training set $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$. If $\mathbf{y}_i = \mathbf{F}(\mathbf{x}_i)$, the neural network model is said to be trained with exact data or noise-free output. A standard neural network consists of multiple layers of interconnected computational units. These layers include an input layer that receives the data, an output layer that generates predictions, and a set of hidden layers that process the information. Suppose the network has $L + 1$ layers, where the number of units in layer l is denoted as N_l, for $l = 0, \ldots, L$. The input layer corresponds to $l = 0$, while the output layer is indexed as $l = L$, meaning that the input and output dimensions satisfy $n = N_0$ and $m = N_L$, respectively. In a feed-forward architecture, signals from layer $l - 1$ are propagated to layer l through weighted connections. Specifically, each unit i in layer $l - 1$ transmits an activation $y_{i,l-1}$, which is scaled by a weight $w_{ji,l}$ and summed over all incoming connections before adding a bias term $b_{j,l}$. This results in a pre-activation signal $u_{j,l} = \sum_{i=1}^{N_{l-1}} w_{ji,l} y_{i,l-1} + b_{j,l}$. Applying an activation function g_l to this pre-activation signal produces the final activation output for unit j in layer l, $y_{j,l} = g_l(u_{j,l})$. To express these computations

in matrix notation, let $\mathsf{W}_l \in \mathbb{R}^{N_l \times N_{l-1}}$ be the weight matrix and $\mathbf{b}_l \in \mathbb{R}^{N_l}$ the bias vector, where their elements are given by $[\mathsf{W}_l]_{ji} = w_{ji,l}$ and $[\mathbf{b}_l]_j = b_{j,l}$, respectively. Defining the set of all network parameters as $\boldsymbol{\omega} = \{\mathsf{W}_l, \mathbf{b}_l\}_{l=1}^{L}$, the feed-forward process can be written compactly as

$$\mathbf{y}_0 = \mathbf{x}, \tag{2a}$$

$$\mathbf{u}_l = \mathsf{W}_l \mathbf{y}_{l-1} + \mathbf{b}_l, \tag{2b}$$

$$\mathbf{y}_l = g_l(\mathbf{u}_l), \quad l = 1, \ldots, L, \tag{2c}$$

$$\mathbf{f}(\mathbf{x}, \boldsymbol{\omega}) = \mathbf{y}_L, \tag{2d}$$

where $[\mathbf{y}_l]_i = y_{i,l}$ and $[\mathbf{u}_l]_j = u_{j,l}$. Training a neural network involves determining the optimal parameter set $\boldsymbol{\omega}$ that best fits the given dataset $\mathcal{D}$. This process, known as deep learning, is typically performed by minimizing a chosen loss function using backpropagation [24]. Appendix provides a statistical perspective on the core principles guiding the development of retrieval algorithms based on neural networks. The discussion encompasses (i) techniques for deriving point estimates, (ii) various forms of uncertainty, and (iii) the utilization of Bayesian regularization for addressing the inverse problem.

2.2. Formulation and solution of the inverse problem with isotropic noise

In this section, we identify the sources of uncertainty in the data vector, formulate an inverse problem with isotropic noise by applying a whitening transformation, and define the objective of the analysis as solving the ill-posed inverse problem by minimizing the residual subject to simple variable bounds.

2.2.1. Sources of uncertainty

We assume that the total uncertainty in the data vector originates from the following sources:

1. Measurement noise $\boldsymbol{\delta}_\mathrm{N}$: This uncertainty stems from errors in the observed data, which vary with each experiment. The measurement noise $\boldsymbol{\delta}_\mathrm{N}$ is assumed to be additive, following a Gaussian distribution with zero mean and a known covariance matrix $\mathsf{C}_{\delta\mathrm{N}}$, i.e., $\boldsymbol{\delta}_\mathrm{N} \sim \mathcal{N}(\mathbf{0}, \mathsf{C}_{\delta\mathrm{N}})$.
2. Surrogate model approximation error $\boldsymbol{\delta}_\mathrm{A}$: This uncertainty arises from approximating $\mathbf{F}$ with a surrogate function $\mathbf{f}$, leading to the approximation error $\boldsymbol{\delta}_\mathrm{A} = \mathbf{F} - \mathbf{f}$. We model $\boldsymbol{\delta}_\mathrm{A}$ as Gaussian random vector with zero mean and covariance matrix $\mathsf{C}_{\delta\mathrm{A}}$, i.e., $\boldsymbol{\delta}_\mathrm{A} \sim \mathcal{N}(\mathbf{0}, \mathsf{C}_{\delta\mathrm{A}})$. To estimate the approximation error, we consider a test set $\mathcal{D}_\mathrm{T} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N_\mathrm{T}}$, where $\mathbf{y}_i = \mathbf{F}(\mathbf{x}_i)$ and N_T is the number of test points. A common estimate for the empirical covariance matrix of the approximation error is the uncentered conditional average covariance matrix over the test set $\mathcal{D}_\mathrm{T}$, given by $\mathsf{C}_{\delta\mathrm{A}} \approx \mathbb{E}(\mathsf{C}_{\delta\mathrm{A}}|\mathcal{D}_\mathrm{T}) = (1/N_\mathrm{T}) \sum_{i=1}^{N_\mathrm{T}} \mathbf{e}_i \mathbf{e}_i^T$, where $\mathbf{e}_i = \mathbf{y}_i - \mathbf{f}(\mathbf{x}_i)$ is the error for the ith test point. This approach assumes that the error vectors $\mathbf{e}_i$ have zero mean. However, the assumption that $\boldsymbol{\delta}_\mathrm{A}$ follows a Gaussian distribution with zero mean holds in expectation, but real samples have a nonzero sample mean. To account for this, we use the maximum likelihood estimate for the conditional average covariance matrix, which centers the data by subtracting the sample mean of the errors $\bar{\mathbf{e}} = (1/N_\mathrm{T}) \sum_{i=1}^{N_\mathrm{T}} \mathbf{e}_i$. Thus, the covariance matrix is estimated as $\mathsf{C}_{\delta\mathrm{A}} \approx \mathbb{E}(\mathsf{C}_{\delta\mathrm{A}}|\mathcal{D}_\mathrm{T}) = (1/N_\mathrm{T}) \sum_{i=1}^{N_\mathrm{T}} (\mathbf{e}_i - \bar{\mathbf{e}})(\mathbf{e}_i - \bar{\mathbf{e}})^T$. This ensures that the covariance estimates reflect the true variance of the approximation error.

Since these error sources are assumed to be independent, we define the noisy data vector through the successive transformations

$$\mathbf{y}^\delta = \mathbf{F}(\mathbf{x}) + \boldsymbol{\delta}_\mathrm{N} = \mathbf{f}(\mathbf{x}) + \boldsymbol{\delta}_\mathrm{A} + \boldsymbol{\delta}_\mathrm{N} = \mathbf{f}(\mathbf{x}) + \boldsymbol{\delta}, \tag{3}$$

where $\boldsymbol{\delta} = \boldsymbol{\delta}_\mathrm{A} + \boldsymbol{\delta}_\mathrm{N}$ is the total uncertainty in the data vector. It is evident that $\boldsymbol{\delta}$ follows a Gaussian distribution $\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \mathsf{C}_\delta)$ with the total covariance matrix $\mathsf{C}_\delta = \mathsf{C}_{\delta\mathrm{A}} + \mathsf{C}_{\delta\mathrm{N}}$. In this regard, given the noisy data vector (3), we aim to solve the inverse problem

$$\mathbf{f}(\mathbf{x}) = \mathbf{y}^\delta. \tag{4}$$

Rather than using the full covariance matrix, we can simplify our analysis by assuming that each type of error is isotropic with an average variance $\sigma_{\delta\mathrm{X}}^2 = \frac{1}{m} \sum_{k=1}^{m} [\mathsf{C}_{\delta\mathrm{X}}]_{kk}$, where X stands for N (measurement noise) and A (surrogate model approximation error). Consequently, we obtain $\mathsf{C}_\delta = \sigma_\delta^2 \mathrm{I}_m$, where $\sigma_\delta^2 = \sigma_{\delta\mathrm{N}}^2 + \sigma_{\delta\mathrm{A}}^2$.

In addition to the above uncertainties, the forward model parameter error $\boldsymbol{\delta}_\mathrm{B}$, which are not trivial to compute, can also be taken into account. This uncertainty arises when the forward model depends not only on the state vector $\mathbf{x}$ but also on a set of parameters $\mathbf{b}$, which are not known precisely. These parameters are not retrieved directly but still influence the retrieval process. Assuming that $\mathbf{b}$ follows a Gaussian distribution with mean $\bar{\mathbf{b}}$ and covariance matrix C_b, i.e., $\mathbf{b} \sim \mathcal{N}(\bar{\mathbf{b}}, \mathsf{C}_\mathrm{b})$, the forward model parameter error is defined as $\boldsymbol{\delta}_\mathrm{B} = \mathbf{F}(\mathbf{x}, \mathbf{b}) - \mathbf{F}(\mathbf{x}, \bar{\mathbf{b}})$. By linearizing $\mathbf{F}(\mathbf{x}, \mathbf{b})$ around $(\mathbf{x}, \bar{\mathbf{b}})$, we obtain $\boldsymbol{\delta}_\mathrm{B} = \mathsf{K}_\mathrm{b}(\mathbf{b} - \bar{\mathbf{b}})$, where K_b is the Jacobian of $\mathbf{F}$ with respect to $\mathbf{b}$. Consequently, the error $\boldsymbol{\delta}_\mathrm{B}$ follows a Gaussian distribution $\boldsymbol{\delta}_\mathrm{B} \sim \mathcal{N}(\mathbf{0}, \mathsf{C}_{\delta\mathrm{B}})$, where the covariance matrix is given by $\mathsf{C}_{\delta\mathrm{B}} = \mathsf{K}_\mathrm{b} \mathsf{C}_\mathrm{b} \mathsf{K}_\mathrm{b}^T$. However, since K_b is not available (as the derivatives of $\mathbf{F}$ are computationally expensive to evaluate) the statistics of $\boldsymbol{\delta}_\mathrm{B}$ can be approximating by performing a Monte Carlo analysis, where samples $\mathbf{b}$ are drawn from the Gaussian distribution $\mathcal{N}(\bar{\mathbf{b}}, \mathsf{C}_\mathrm{b})$. When the forward model parameter error is also taken into account, the noisy data vector is defined as

$$\mathbf{y}^\delta = \mathbf{F}(\mathbf{x}, \mathbf{b}) + \boldsymbol{\delta}_\mathrm{N} = \mathbf{F}(\mathbf{x}, \bar{\mathbf{b}}) + \boldsymbol{\delta}_\mathrm{B} + \boldsymbol{\delta}_\mathrm{N} = \mathbf{f}(\mathbf{x}) + \boldsymbol{\delta}_\mathrm{A} + \boldsymbol{\delta}_\mathrm{B} + \boldsymbol{\delta}_\mathrm{N} = \mathbf{f}(\mathbf{x}) + \boldsymbol{\delta}, \tag{5}$$

where the total uncertainty in the data vector $\boldsymbol{\delta} = \boldsymbol{\delta}_\mathrm{A} + \boldsymbol{\delta}_\mathrm{B} + \boldsymbol{\delta}_\mathrm{N}$ follows a Gaussian distribution, i.e., $\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \mathsf{C}_\delta)$, with the total covariance matrix $\mathsf{C}_\delta = \mathsf{C}_{\delta\mathrm{A}} + \mathsf{C}_{\delta\mathrm{B}} + \mathsf{C}_{\delta\mathrm{N}}$.

2.2.2. Whitening transformation

In the next step, we transform the representation of the noisy data vector (see Eq. (3)) into a form in which the noise is isotropic. For this purpose, we define the average variance of the total error as

$$\sigma_\delta^2 = \frac{1}{m} \sum_{k=1}^{m} [\mathsf{C}_\delta]_{kk} \tag{6}$$

and introduce the normalized covariance matrix

$$\widehat{\mathsf{C}}_\delta = \frac{1}{\sigma_\delta^2} \mathsf{C}_\delta. \tag{7}$$

To decorrelate the noise, we apply a whitening transformation using a matrix W, which satisfies $\mathsf{W} \widehat{\mathsf{C}}_\delta \mathsf{W}^T = \mathrm{I}_m$. A common choice for W is $\widehat{\mathsf{C}}_\delta^{-1/2}$. One way to compute $\widehat{\mathsf{C}}_\delta^{-1/2}$ is through the Cholesky factorization $\widehat{\mathsf{C}}_\delta = \mathsf{L}\mathsf{L}^T$, where L is the lower triangular Cholesky factor, in which case, $\mathsf{W} = \mathsf{L}^{-1}$. Alternatively, if the eigenvalue decomposition of $\widehat{\mathsf{C}}_\delta$ is available, i.e., $\widehat{\mathsf{C}}_\delta = \mathsf{U}\Lambda\mathsf{U}^T$, then $\mathsf{W} = \Lambda^{-1/2}\mathsf{U}^T$. Applying W to both sides of the representation of the noisy data vector (3) transforms it into $\mathsf{W}\mathbf{y}^\delta = \mathsf{W}\mathbf{f}(\mathbf{x}) + \mathsf{W}\boldsymbol{\delta}$. Since the transformed noise $\widehat{\boldsymbol{\delta}} = \mathsf{W}\boldsymbol{\delta}$ now follows an isotropic Gaussian distribution $\widehat{\boldsymbol{\delta}} \sim \mathcal{N}(\mathbf{0}, \sigma_\delta^2 \mathrm{I}_m)$, the problem is reformulated as

$$\widehat{\mathbf{y}}^\delta = \widehat{\mathbf{f}}(\mathbf{x}) + \widehat{\boldsymbol{\delta}}, \tag{8}$$

where $\widehat{\mathbf{y}}^\delta = \mathsf{W}\mathbf{y}^\delta$ and $\widehat{\mathbf{f}}(\mathbf{x}) = \mathsf{W}\mathbf{f}(\mathbf{x})$. Thus, we focus on solving the transformed inverse problem

$$\widehat{\mathbf{f}}(\mathbf{x}) = \widehat{\mathbf{y}}^\delta. \tag{9}$$

Clearly, if all data errors are isotropic, we have $\widehat{\mathsf{C}}_\delta = \mathrm{I}_m$ and $\mathsf{W} = \mathrm{I}_m$, and the transformed inverse problem becomes identical to the original one, with $\widehat{\mathbf{f}}(\mathbf{x})$ and $\widehat{\mathbf{y}}^\delta$ directly corresponding to $\mathbf{f}(\mathbf{x})$ and $\mathbf{y}^\delta$, respectively.

2.2.3. Solving the ill-posed inverse problem by minimizing residuals with simple variable bounds

According to Hadamard [25], the inverse problem (9) is said to be well-posed if the solution satisfies three conditions: existence, uniqueness, and stability. If any of these conditions are not met, the problem is classified as ill-posed. In our case, these conditions manifest as follows:

1. Non-existence occurs when $\hat{\mathbf{y}}^\delta \notin \mathcal{R}(\hat{\mathbf{f}})$, meaning that the given data lies outside the range of $\hat{\mathbf{f}}$.
2. Non-uniqueness arises because $\mathbf{F}$ (and consequently, $\hat{\mathbf{f}}$, assuming that the surrogate approximation and the whitening transformation preserve this property) is not injective.
3. Instability is assessed by analyzing the Jacobian matrix of $\hat{\mathbf{f}}$. Specifically:
   (a) A high condition number indicates that small changes in the output can lead to large variations in the input, making the solution highly sensitive to noise.
   (b) Eigenvalues close to zero suggest that the function is nearly flat along certain directions, implying potential ill-conditioning and sensitivity to perturbations.

Ill-posed problems are often addressed using regularization techniques, with Tikhonov regularization being a prominent example [26]. In this context, the solution of the inverse problem is the minimizer of the regularized objective function

$$\mathcal{F}_{\delta,\lambda}(\mathbf{x}) = \frac{1}{2}(\|\hat{\mathbf{f}}(\mathbf{x}) - \hat{\mathbf{y}}^\delta\|^2 + \lambda\|\mathsf{L}(\mathbf{x} - \mathbf{x}_\mathrm{a})\|^2), \tag{10}$$

where λ is the regularization parameter, L the regularization matrix, and $\mathbf{x}_\mathrm{a}$ the prior estimate of $\mathbf{x}$. A few comments can be made here.

1. The regularization matrix impose specific features on the solution. If the magnitude of the solution should not be extremely large, L can be chosen as either the identity matrix or a diagonal matrix. If the solution should be smooth, L can be selected as discrete approximations to the first- and second-order derivative operators. If statistical information about the solution is available, such as the prior covariance matrix C_x, L can be chosen as the Cholesky factor of $\mathsf{C}_\mathrm{x}^{-1}$. In this case, the resulting method is known as Bayesian regularization (or optimal estimation) [10,11], where the regularization (penalty) term takes the form $(\mathbf{x} - \mathbf{x}_\mathrm{a})^T\mathsf{C}_\mathrm{x}^{-1}(\mathbf{x} - \mathbf{x}_\mathrm{a})$, and the minimizer of the regularized objective function represents the maximum a posteriori estimate. The fundamentals of Bayesian regularization for solving inverse problems using neural networks is outlined in Appendix.
2. Selecting an appropriate regularization parameter λ requires careful consideration to achieve an optimal balance between residual error, which accounts for noise, and smoothing error, which governs the smoothness of the solution. A small value of λ tends to emphasize the minimization of the residual error, which measures the discrepancy between the model predictions and the noisy observations. Conversely, a large value of λ places greater weight on the regularization term, which enforces smoothness or other prior information about the solution. For nonlinear problems, the regularization parameter can be chosen using two main approaches: (i) a priori methods, with the expected error estimation method as a prominent example, and (ii) selection criteria with a variable regularization parameter, which include error-free parameter choice methods such as generalized cross-validation, maximum likelihood estimation, and the L-curve method. Unfortunately, no regularization method is both efficient and infallible. For instance, the expected error estimation method is extremely computationally expensive, as it involves randomly exploring the domain where the solution is assumed to lie and computing the optimal regularization parameter for error estimation for each realization of the state vector. On the other hand, the generalized cross-validation function, and sometimes the maximum likelihood function, may exhibit a very flat minimum, making it difficult to determine precisely, while the L-curve may lose its characteristic shape, obscuring the corner location [27].

When specific information about the solution, such as its smoothness or covariance structure, is lacking (as in the case considered in our analysis), it is appropriate to minimize the objective function

$$\mathcal{F}_\delta(\mathbf{x}) = \frac{1}{2}\|\hat{\mathbf{f}}(\mathbf{x}) - \hat{\mathbf{y}}^\delta\|^2 \tag{11}$$

while imposing simple bounds on the variables. By constraining the solution within a defined box $X = [a_1, b_1] \times [a_2, b_2] \ldots \times [a_n, b_n]$, we can effectively control the solution's magnitude and mitigate potential instabilities arising from data uncertainty. Imposing simple bounds on the variables, where each component of $\mathbf{x}$ is constrained within its respective range, can also be interpreted as a form of regularization. This approach limits the solution space, preventing the solution from becoming excessively large or unbounded, similar to a regularization term with the identity matrix. While this method does not replace formal regularization techniques, it serves as a practical alternative when specific prior information about the solution is unavailable. In both cases, the bounds act as a "soft" constraint, helping to prevent overfitting to the noisy data by controlling the solution's magnitude without imposing complex structure. Thus, minimizing residuals with simple bounds on the variables can be viewed as a weak form of regularization.

2.3. Advanced multistart solver

Consider the solution to the inverse problem (cf. Eq. (9)) $\hat{\mathbf{f}}(\mathbf{x}) = \hat{\mathbf{y}}^\delta$ within the box $X = [a_1, b_1] \times [a_2, b_2] \ldots \times [a_n, b_n]$ by minimizing the least-squares function (cf. Eq. (11)) $\mathcal{F}_\delta(\mathbf{x}) = (1/2)\|\hat{\mathbf{f}}(\mathbf{x}) - \hat{\mathbf{y}}^\delta\|^2$ subject to simple bounds on the variables. Assuming that the surrogate function $\mathbf{f}$ inherits the nonlinearity of the forward model $\mathbf{F}$, and the whitening transformation preserves the nonlinearity of $\mathbf{f}$, we deduce that $\hat{\mathbf{f}}$ is also nonlinear. This implies that for $\hat{\mathbf{y}}^\delta \in \mathcal{R}(\hat{\mathbf{f}})$, the objective function $\mathcal{F}_\delta$ has several zero global minima, while for $\hat{\mathbf{y}}^\delta \notin \mathcal{R}(\hat{\mathbf{f}})$, it has several local minima.

Deterministic global optimization methods provide rigorous frameworks for identifying all minima of an objective function. These methods systematically explore the search space, ensuring completeness and convergence guarantees. Among these approaches, αBB (alpha-Branch-and-Bound) extends classical branch-and-bound techniques by constructing convex underestimators for nonconvex functions, enabling rigorous global optimization [28]. Lipschitz optimization methods utilize known Lipschitz constants to bound function variations, refining search regions efficiently [29,30]. Interval branch-and-bound methods use interval arithmetic to systematically enclose solutions and eliminate regions that cannot contain global minima, ensuring completeness [31]. Global homotopy methods with branch-switching track solution paths by continuously deforming a simpler problem into the target problem. By detecting bifurcations, branch-switching strategies enhance standard homotopy methods, making them particularly useful for root-finding but less suited for optimization [32]. Finally, exhaustive grid search systematically samples the domain, ensuring all minima are identified within the resolution of the discretization. While reliable in low-dimensional settings, its exponential complexity growth limits its scalability to high-dimensional problems [33]. Another class of methods addressing global optimization involves clustering-based techniques integrated with multistart algorithms [34–37]. These techniques generate a diverse set of starting points, followed by clustering to group similar points. The goal is to ensure thorough exploration of the search space while significantly reducing the number of local search applications. In this context, the MinFinder algorithm, developed by Tsoulos and Lagaris [38], is particularly noteworthy. This clustering-based technique, inspired by the Topographical Multilevel

Single Linkage approach by Ali and Storey [39], combines a multistart strategy with efficient validation criteria for starting points to avoid redundant searches. These methods serve as robust frameworks for global optimization but are not specifically suited for nonlinear least-squares problems.

Although the inverse problem involves the noisy data vector $\widehat{\mathbf{y}}^{\delta} = \widehat{\mathbf{f}}(\mathbf{x}) + \widehat{\boldsymbol{\delta}}$, we begin by examining the solvable equation $\widehat{\mathbf{f}}(\mathbf{x}) = \widehat{\mathbf{y}}$, where $\widehat{\mathbf{y}} \in \mathcal{R}(\widehat{\mathbf{f}})$ is the exact data vector corresponding to $\widehat{\mathbf{f}}$. A multistart algorithm for solving the equation $\widehat{\mathbf{f}}(\mathbf{x}) = \widehat{\mathbf{y}}$ employs, at each iteration, a new set of starting points $S_0 = \{\mathbf{x}_{0k}\}_{k=1}^{N_0}$. For each starting point, a local search (denoted by L) is applied to minimize the least-squares function $\mathcal{F}(\mathbf{x}) = (1/2)\|\widehat{\mathbf{f}}(\mathbf{x}) - \widehat{\mathbf{y}}\|^2$. With the assumption $\widehat{\mathbf{y}} \in \mathcal{R}(\widehat{\mathbf{f}})$, the algorithm selects solutions $\mathbf{x}_{Sk}$ that meet the absolute function convergence criterion $\mathcal{F}(\mathbf{x}_{Sk}) \leq \varepsilon_{\mathrm{F}}$, with ε_{F} as the absolute function tolerance (near to zero). Such solutions are denoted by $\mathbf{x}_{Sk} = L(\widehat{\mathbf{y}} \mid \mathbf{x}_{0k})$. At its core, the main steps of a multistart-like method are outlined in Algorithm 1. Multistart algorithms differ in how they generate and validate starting points, the local search method they employ, and the termination conditions they use. For example, in the MinFinder algorithm, the double box stopping rule serves as the termination criterion, while a point is considered a valid starting point if it is not too close to an already identified minimum or another sample (the closeness to a local minimum or another sample is determined by the so-called typical distance and the gradient criterion).

Algorithm 1 Main steps of a multistart algorithm.

1. *Input*: Data vector $\widehat{\mathbf{y}} \in \mathcal{R}(\widehat{\mathbf{f}})$.
2. *Initialization*: Set $X_{\mathrm{S}} = \emptyset$ to store all solutions.
3. *Generate Starting Points*: Randomly generate a configuration of sample points representing a set of N_0 starting points $S_0 = \{\mathbf{x}_{0k}\}_{k=1}^{N_0}$.
4. *Iterate Through Starting Points*:
For each starting points $\mathbf{x}_{0k}$, $k = 1, \ldots, N_0$ do:
   - *Validation of Starting Points*: Check if $\mathbf{x}_{0k}$ is a valid starting point.
   - *Solution Search*: If valid, use $\mathbf{x}_{0k}$ as the initial point for a local optimization search. If the local search is successful and the solution $\mathbf{x}_{\mathrm{S}} = L(\widehat{\mathbf{y}} \mid \mathbf{x}_{0k}) \notin X_{\mathrm{S}}$, update X_{S} to include this new solution.
5. *Termination Check*: If a termination criterion is satisfied, then return the set of solutions X_{S} as the output; else, go to Step 3.

In this section we present a multistart algorithm that integrates the double box stopping rule from the MinFinder algorithm, while introducing an enhanced validation criterion for the starting points. This criterion is based on the observation that, since we are solving the inverse problem $\widehat{\mathbf{f}}(\mathbf{x}) = \widehat{\mathbf{y}}^{\delta}$ for multiple data vectors rather than a general global optimization problem, an appropriate set of starting points should reflect the characteristics of $\widehat{\mathbf{f}}$, particularly its injectivity domains. To explore the relationship between the selection of starting points and the injectivity domains of $\widehat{\mathbf{f}}$, we assume that the domain X can be partitioned into N_{IN} injectivity domains X_k, where $k = 1, \ldots, N_{\mathrm{IN}}$. A subset $X_k \subset X$ is considered an injectivity domain of $\widehat{\mathbf{f}}$, if for each $\widehat{\mathbf{y}} \in \mathcal{R}(\widehat{\mathbf{f}}\,|_{X_k})$, the equation $\widehat{\mathbf{f}}(\mathbf{x}) = \widehat{\mathbf{y}}$ has a unique solution, with $\widehat{\mathbf{f}}\,|_{X_k}$ representing the restriction of $\widehat{\mathbf{f}}$ to the domain X_k. Furthermore, we make the following assumptions regarding the local search:

A1. A solution within an injectivity domain can only be generated by a starting point from the same injectivity domain.
A2. A starting point within one injectivity domain always produces a solution in the same injectivity domain.

Assumption A1 implies that the injectivity domains X_k are disjoint, meaning no point belongs to more than one injectivity domain. Assumption A2 implies that the boundaries of each injectivity domain are preserved, preventing leakage of solutions between domains. In other words, the mapping from a starting point to its corresponding injectivity domain is one-to-one. Therefore, under the above assumptions, a set of starting points must contain at least one point from each domain of injectivity. The optimal set of starting points will contain exactly N_{IN} starting points, each corresponding to a distinct injectivity domain.

Computing the injectivity domains of $\widehat{\mathbf{f}}$ is a challenging task, especially for high-dimensional problems. Although a few studies, such as those in Refs. [40,41], focus on testing injectivity over domains, to the best of our knowledge, no work has been conducted on the direct computation of these domains. Since the computation of injectivity domains is a highly complex and difficult task, we propose estimating these domains roughly to determine an appropriate set of starting points. More precisely, and referring to the notations in Algorithm 1, we aim to determine a set of starting points $\widehat{S}_0$ such that, for any $\widehat{\mathbf{y}} \in \mathcal{R}(\widehat{\mathbf{f}})$, the multistart algorithm using the starting points in $\widehat{S}_0$ will yield a set of solutions that approximates the set of all solutions X_{S} as closely as possible. The approach will involve an exploration of a solution set and a starting point set to establish a mapping between each solution and a set of starting points that lead to it. Since a starting point may generate several solutions within the same injectivity domain, a hitting set problem will be solved to determine a minimal set of starting points that produce all considered solutions. This set, which captures the nonlinearity of the function $\widehat{\mathbf{f}}$ and is independent of the data vector $\widehat{\mathbf{y}}$, will be used in the first configuration step of a multistart algorithm (meaning the first step will not involve random generation of starting points). If, for any new data vector, which implies the solution of a new global optimization problem, the set of starting points captures almost all solutions at the first iteration, we expect that fewer random starting point configurations—and consequently, fewer local search calls—will be required to find the remaining solutions in subsequent iterations. This will improve the efficiency of the multistart algorithm.

2.3.1. Starting point generation algorithm

The starting point generation algorithm is based on the observation that each solution can be reached from multiple starting points and that a minimal set of starting points covering all solutions can be obtained by solving a hitting set problem. To implement this idea, we generate a large set of starting points $S_0 = \{\mathbf{x}_{0k}\}_{k=1}^{N_0}$ and a large set of solutions $X_{\mathrm{S}} = \{\mathbf{x}_{Sn}\}_{n=1}^{N_{\mathrm{S}}}$. For each solution $\mathbf{x}_{Sn}$, we compute the data vector $\widehat{\mathbf{y}}_{Sn} = \widehat{\mathbf{f}}(\mathbf{x}_{Sn})$, and identify the set of starting points $S_{0n} = \{\mathbf{x}_{0k_n}\}_{k_n \in I_0}$ with $I_0 = \{k\}_{k=1}^{N_0}$, yielding approximately the solution $\mathbf{x}_{Sn}$ within a specified tolerance. By this procedure, we generate the map

$$\mathbf{x}_{Sn} \longmapsto S_{0n} = \{\mathbf{x}_{0k_n}\}_{k_n \in I_0} \text{ for all } n = 1, \ldots, N_{\mathrm{S}},$$

and note that a starting point may belong to multiple sets S_{0n}. Next, we formulate a hitting set problem with the universe $S_0 = \{\mathbf{x}_{0k}\}_{k=1}^{N_0}$ and the collection of sets $C = \{S_{0n} \mid n = 1, \ldots, N_{\mathrm{S}}\}$. The goal is to determine a collection of N_{H} sets $S = \{S_{0h} \mid h = 1, \ldots, N_{\mathrm{H}}\}$ from the universe S_0, such that all sets in C are hit by any S_{0h}, i.e., for any S_{0h} with $h = 1, \ldots, N_{\mathrm{H}}$, we have that $S_{0h} \cap S_{0n} \neq \emptyset$ for all $n = 1, \ldots, N_{\mathrm{S}}$. If the set S_{0h} cannot be made smaller without losing this property, it is said to be a minimal hitting set. In other words, any set $S_{0h} = \{\mathbf{x}_{0k_h}\}_{k_h \in I_0}$ that solves the hitting set problem contains a small number of elements and produces all solutions in X_{S}. The hitting set problem can be solved for the sets $S_{0n} = \{\mathbf{x}_{0k_n}\}_{k_n \in I_0}$, or more efficiently, for the set of starting-point indices $\mathcal{I}_{0n} = \{k_n\}_{k_n \in I_0}$ associated with S_{0n}.

The solution to the hitting set problem is a crucial aspect of our algorithm. There are many algorithms for enumerating minimal hitting sets with publicly-available software implementations [42]. Some examples are the MTMiner algorithm [43], the DL algorithm [44], the greedy algorithm (e.g., [45]), and the integer programming algorithm [46]. The classical greedy algorithm can find hitting sets, but it does not necessarily find minimal ones. To find minimal sets, we propose the following modified version:

1. *Basic step*: In every iteration step of the algorithm, an element from the universe S_0 is selected that most often hits the unhit sets in C. If there are several elements with this property, one of them is randomly selected. The element is then added to the hitting set S_{0h}, and all newly hit sets are removed from C. The algorithm proceeds until C runs empty.
2. *Minimality step*: Iteratively remove elements from a current hitting set S_{0h}, and if the resulting set still hits all the sets in C, replace the current set with this new, smaller set.

A comparison between the proposed greedy algorithm and the DL algorithm has shown that the greedy algorithm finds some minimal hitting sets but not all of them. However, since finding all minimal hitting sets is not essential for our purposes, we will use this algorithm in our numerical analysis, as it is clearly superior to the MTMiner and DL algorithms in terms of efficiency and memory requirements.

In principle, after solving the hitting set problem, we can identify $\widehat{S}_0$ with any set S_{0h}. For example, we may select the set with the fewest elements for efficiency reasons in Steps 9 and 10 of Algorithm 3, which will be discussed in Section 2.3.2, or the set with the most elements for completeness reasons in the multistart algorithm, to be covered in Section 2.3.3. Another option is to compute all solutions of the equation $\widehat{\mathbf{f}}(\mathbf{x}) = \widehat{\mathbf{y}}$ for a given set of data vectors using the initial set of starting points S_0 and any S_{0h}, $h = 1, \ldots, N_{\mathrm{H}}$, identifying $\widehat{S}_0$ as the set S_{0h_0} that produces a total number of solutions closest to that obtained with S_0. The starting point generation procedure is implemented in Algorithm 2, which solves the hitting set problem for the set of starting-point indices $\mathcal{I}_{0n} = \{k_n\}_{k_n \in I_0}$.

Algorithm 2 Starting point generation algorithm.

1. *Inputs*: Number of starting points N_0, number of solutions N_{S}, solution tolerance ε_{X}, and function tolerance ε_{F} for the local search.
2. *Generate Initial Sets*: Randomly generate a (large) set of N_0 starting points $S_0 = \{\mathbf{x}_{0k}\}_{k=1}^{N_0}$, and a set of N_{S} solutions $X_{\mathrm{S}} = \{\mathbf{x}_{\mathrm{S}n}\}_{n=1}^{N_{\mathrm{S}}} \subset X$. Let $I_0 = \{k\}_{k=1}^{N_0}$ be the set of starting-point indices associated to S_0.
3. *Generate Data Set*: Compute the data set $Y_{\mathrm{S}} = \{\widehat{\mathbf{y}}_{\mathrm{S}n} = \widehat{\mathbf{f}}(\mathbf{x}_{\mathrm{S}n})\}_{n=1}^{N_{\mathrm{S}}}$.
4. *Initialization*: Set $S_{0n} = \emptyset$, $n = 1, \ldots, N_{\mathrm{S}}$ to store all starting points that yield the solution $\mathbf{x}_{\mathrm{S}n}$.
5. *Identify Starting Points Yielding Solutions*:
For each solution pair $(\mathbf{x}_{\mathrm{S}n}, \widehat{\mathbf{y}}_{\mathrm{S}n})$, $n = 1, \ldots, N_{\mathrm{S}}$ do:
   - For each starting point $\mathbf{x}_{0k}$, $k = 1, \ldots, N_0$ do:
     - Run a local search with the starting point $\mathbf{x}_{0k}$. If a starting point $\mathbf{x}_{0k_n}$, $k_n \in I_0$ yields (approximately) the solution $\mathbf{x}_{\mathrm{S}n}$, i.e., $\left\|\mathbf{x}_{\mathrm{S}n} - L(\widehat{\mathbf{y}}_{\mathrm{S}n} \mid \mathbf{x}_{0k_n})\right\| \leq \varepsilon_{\mathrm{X}} \left\|\mathbf{x}_{\mathrm{S}n}\right\|$, insert $\mathbf{x}_{0k_n}$ in the set S_{0n}.
6. *Solve Hitting Set Problem*: Define $\mathcal{I}_{0n} = \{k_n\}_{k_n \in I_0}$ as the set of starting-point indices associated to $S_{0n} = \{\mathbf{x}_{0k_n}\}_{k_n \in I_0}$. Solve a hitting set problem for the collection of sets of starting-point indices $C = \{\mathcal{I}_{0n} \mid n = 1, \ldots, N_{\mathrm{S}}\}$ and universe I_0. For the solution comprising a collection of N_{H} hitting sets $H = \{I_{0h} \mid h = 1, \ldots, N_{\mathrm{H}}\}$, where $I_{0h} = \{k_h\}_{k_h \in I_0}$, consider the associated collection of starting point sets $S = \{S_{0h} \mid h = 1, \ldots, N_{\mathrm{H}}\}$, where $S_{0h} = \{\mathbf{x}_{0k_h}\}_{k_h \in I_0}$.
7. *Select Starting Set*: From the collection S, select a starting set $\widehat{S}_0 = \{\widehat{\mathbf{x}}_{0k}\}_{k=1}^{\widehat{N}_0}$.
8. *Output*: Return the starting set $\widehat{S}_0 = \{\widehat{\mathbf{x}}_{0k}\}_{k=1}^{\widehat{N}_0}$.

In conclusion, Algorithm 2 provides a set of starting points $\widehat{S}_0 = \{\widehat{\mathbf{x}}_{0k}\}_{k=1}^{\widehat{N}_0}$ that presumably yield a set of solutions close to the set of all solutions of the equation $\widehat{\mathbf{f}}(\mathbf{x}) = \widehat{\mathbf{y}}$.

2.3.2. Rough estimation of injectivity domains

In principle, the starting points generated by Algorithm 2 can be associated with the injectivity domains of $\widehat{\mathbf{f}}$. To illustrate this, we augment Algorithm 2 with two additional steps, as detailed in Algorithm 3. In Step 9, after determining the minimal hitting set $\widehat{S}_0 = \{\widehat{\mathbf{x}}_{0k}\}_{k=1}^{\widehat{N}_0}$, further local searches are conducted from these selected starting points to identify regions corresponding to presumed injectivity domains. Specifically, each starting point $\widehat{\mathbf{x}}_{0k}$ is associated to a possible injectivity domain $X_{\mathrm{S}k} = \{\mathbf{x}_{\mathrm{S}n_k}\}_{n_k}$, where $n_k \in \{1, \ldots, N_{\mathrm{S}}\}$. In an ideal scenario, if the sets S_0 and X_{S} are sufficiently large and the local search algorithm performs effectively (in the sense that Assumptions A1 and A2 are satisfied), the mapping $\widehat{\mathbf{x}}_{0k} \longmapsto X_{\mathrm{S}k} = \{\mathbf{x}_{\mathrm{S}n_k}\}_{n_k}$ is one-to-one. However, if this is not the case, two potential situations may arise:

1. Assumption A1 is not satisfied: A solution within an injectivity domain may be generated by a starting point from a different injectivity domain, resulting in overlapping injectivity sets. Step 10 of Algorithm 3 addresses this issue. It generates disjoint injectivity sets by: (i) identifying all starting points $\widehat{S}_{0n}$ that produce a given solution $\mathbf{x}_{\mathrm{S}n}$, (ii) selecting the starting point $\widehat{\mathbf{x}}_{0l}$ from $\widehat{S}_{0n}$ that is closest to $\mathbf{x}_{\mathrm{S}n}$ in Euclidean distance, and (iii) retaining $\mathbf{x}_{\mathrm{S}n}$ in $X_{\mathrm{S}l}$, while removing it from all other sets $X_{\mathrm{S}k}$ associated with the starting points in $\widehat{S}_{0n}$.
2. Assumption A2 is not satisfied: A starting point within one injectivity domain may produce a solution in a different injectivity domain. This issue arises when starting points near the boundary of an injectivity domain "drift" into adjacent domains due to the gradient descent behavior of the algorithm, particularly when a Newton-type method is used to solve the least-squares problem. Consequently, the boundaries of the injectivity domain are not preserved, leading to a leak of solutions into adjacent domains and a breakdown in the structure of the solution process.

Algorithm 3 Injectivity set generation algorithm.

8. *Initialization*: Set $X_{\mathrm{S}k} = \emptyset$, $k = 1, \ldots, \widehat{N}_0$ to store all solutions corresponding to $\widehat{\mathbf{x}}_{0k}$.
9. *Generate Injectivity Sets*:
For each solution pair $(\mathbf{x}_{\mathrm{S}n}, \widehat{\mathbf{y}}_{\mathrm{S}n} = \widehat{\mathbf{f}}(\mathbf{x}_{\mathrm{S}n}))$, $n = 1, \ldots, N_{\mathrm{S}}$ do:
   - For each starting point $\widehat{\mathbf{x}}_{0k}$, $k = 1, \ldots, \widehat{N}_0$ do:
     - Run a local search with the starting point $\widehat{\mathbf{x}}_{0k}$. If a starting point $\widehat{\mathbf{x}}_{0k_n}$, $k_n \in \widehat{I}_0 = \{k\}_{k=1}^{\widehat{N}_0}$ yields (approximately) the solution $\mathbf{x}_{\mathrm{S}n}$, i.e., $\left\|\mathbf{x}_{\mathrm{S}n} - L(\widehat{\mathbf{y}}_{\mathrm{S}n} \mid \widehat{\mathbf{x}}_{0k_n})\right\| \leq \varepsilon_{\mathrm{X}} \left\|\mathbf{x}_{\mathrm{S}n}\right\|$, insert $\mathbf{x}_{\mathrm{S}n}$ in the set $X_{\mathrm{S}k_n}$.
10. *Generate Disjoint Injectivity Sets*:
For each solution $\mathbf{x}_{\mathrm{S}n}$, $n = 1, \ldots, N_{\mathrm{S}}$ do:
   - Initialize the set of all starting points producing the solution $\mathbf{x}_{\mathrm{S}n}$, $\widehat{S}_{0n} = \emptyset$.
   - For each starting point $\widehat{\mathbf{x}}_{0k}$, $k = 1, \ldots, \widehat{N}_0$ do:
     - If for a starting point $\widehat{\mathbf{x}}_{0k_n}$, $k_n \in \widehat{I}_0$, the associated injectivity set $X_{\mathrm{S}k_n}$ contains the solution $\mathbf{x}_{\mathrm{S}n}$, insert $\widehat{\mathbf{x}}_{0k_n}$ in the set $\widehat{S}_{0n}$.
   - From the set $\widehat{S}_{0n}$, choose the starting point $\widehat{\mathbf{x}}_{0l}$ that is closest to the solution $\mathbf{x}_{\mathrm{S}n}$ with respect to the Euclidean distance. Keep $\mathbf{x}_{\mathrm{S}n}$ in the set $X_{\mathrm{S}l}$, and remove $\mathbf{x}_{\mathrm{S}n}$ from all the other sets $X_{\mathrm{S}k}$ associated to the starting points in $\widehat{S}_{0n}$.
11. *Outputs*: Return the starting set $\widehat{S}_0 = \{\widehat{\mathbf{x}}_{0k}\}_{k=1}^{\widehat{N}_0}$ and the associated injectivity sets $X_{\mathrm{S}k}$, $k = 1, \ldots, \widehat{N}_0$.

In fact, Step 10 of Algorithm 3 does not provide a precise rule for constructing disjoint injectivity sets (the use of Euclidean distance is

just one possible approach). Therefore, Steps 9 and 10 should be considered optional. They offer only a rough estimation of the injectivity domains and do not guarantee the exact boundaries of these domains. For this reason, the set of starting points generated by Algorithm 2 will be used in the first configuration step of a multistart algorithm, with new configurations of samples introduced in subsequent iterations to capture all solutions.

2.3.3. Multistart algorithm with enhanced starting point generation

The multistart algorithm that utilizes the set of starting points $\widehat{S}_0 = \{\widehat{\mathbf{x}}_{0k}\}_{k=1}^{\widehat{N}_0}$ in the first configuration step is referred to as the multistart algorithm with enhanced starting point generation. Algorithm 4 outlines the main steps of this approach, which consists of an outer loop over the absolute function tolerance and an inner loop over the starting point configurations.

Absolute function tolerance loop. While the algorithm is primarily designed to handle noisy data, it can also be applied to solve the equation $\widehat{\mathbf{f}}(\mathbf{x}) = \widehat{\mathbf{y}}$, where $\widehat{\mathbf{y}} \in \mathcal{R}(\widehat{\mathbf{f}})$ is the exact data vector corresponding to $\widehat{\mathbf{f}}$. In this case, a single absolute function tolerance is sufficient, eliminating the need for a loop. This approach involves minimizing the noise-free objective function $\mathcal{F}(\mathbf{x}) = (1/2)\|\widehat{\mathbf{f}}(\mathbf{x}) - \widehat{\mathbf{y}}\|^2$ within a partitioned domain X comprising injectivity domains X_k, where a local search algorithm can ideally identify solutions if the starting point lies within an injectivity domain.

However, with noisy data $\widehat{\mathbf{y}}^\delta = \widehat{\mathbf{f}}(\mathbf{x}) + \widehat{\boldsymbol{\delta}}$, the scenario becomes more complex.

1. If $\widehat{\mathbf{y}}^\delta \in \mathcal{R}(\widehat{\mathbf{f}})$, noisy data can lead to global minima (near to zero) of the noise-affected objective function $\mathcal{F}_\delta(\mathbf{x}) = (1/2)\|\widehat{\mathbf{f}}(\mathbf{x}) - \widehat{\mathbf{y}}^\delta\|^2$.
2. If $\widehat{\mathbf{y}}^\delta \notin \mathcal{R}(\widehat{\mathbf{f}})$, the presence of noise can create an optimization landscape with valleys and peaks, leading to local minima.

In summary, noisy data can give rise to both global and local minima. Local minima typically occur when the multistart algorithm encounters parameter values that yield lower objective function values, even though these solutions may not correspond to injectivity domains.

A multistart algorithm can effectively capture multiple global and local minima in the presence of noisy data. However, when employing an absolute function convergence criterion for a local search, selecting the appropriate tolerance is essential to ensure that the algorithm accurately identifies minima without terminating prematurely. Algorithm 4 starts with a small absolute function tolerance ε_{F} and increasing it if no solutions are found. This approach prioritizes computational resources for finding global minima first. If no global minima are found, the search criteria are relaxed to capture local minima. Specifically, the absolute function tolerance ε_{F} follows a geometric sequence with ratio c_ε, ranging between $\varepsilon_{\mathrm{Fmin}}$ and $\varepsilon_{\mathrm{F\,max}}$. The initial tolerance $\varepsilon_{\mathrm{Fmin}}$ is set to a very low value, say, 10^{-16}, while the final tolerance $\varepsilon_{\mathrm{F\,max}}$ is set to $\Delta/2$, where $\Delta = \sqrt{m}\sigma_\delta$ represents the noise level and σ_δ^2 is the average variance of the noise given by Eq. (6). Notably, this choice aligns with the discrepancy principle, which states that iteration should be stopped when the residual is of the order of the noise level, or more precisely, when $\|\widehat{\mathbf{f}}(\mathbf{x}_{\mathrm{S}}^\delta) - \widehat{\mathbf{y}}^\delta\| \leq \eta\Delta$ with $\eta > 1$ being a control parameter [47].

Starting point configuration loop. At the first configuration step, the set of starting points $\widehat{S}_0 = \{\widehat{\mathbf{x}}_{0k}\}_{k=1}^{\widehat{N}_0}$, which was originally computed using Algorithm 2 for exact data corresponding to $\widehat{\mathbf{f}}$, is also employed in the case of noisy data. In subsequent iterations, starting points are sampled from a larger box X_2, which contains X, and is constructed in a way such that $\mu(X_2) = 2\mu(X)$, where $\mu(X)$ is the measure of X (the volume of the box X). This sampling approach follows the double box stopping rule used in the MinFinder algorithm. Actually this stopping rule enhances the algorithm's efficiency when the set of starting points $\widehat{S}_0$ provided by the starting point generation algorithm is used. To demonstrate this, we will examine the stopping rule in detail. The quantity $\delta_j = jN_{0j}/N_{0\mathrm{P}}$, where N_{0j} is the number of sample points at the current iteration j and $N_{0\mathrm{P}}$ is the total number of sample points after j iterations, has (i) the expectation value $\langle\delta\rangle_j = (1/j)\sum_{i=1}^{j}\delta_i$ that tends to $\mu(X)/\mu(X_2) = 1/2$ as $j \to \infty$, and (ii) the variance $\sigma_j^2(\delta) = \langle\delta^2\rangle_j - \langle\delta\rangle_j^2$ that tends to 0 as $j \to \infty$. The variance, being smoother than the expectation value, is chosen for the stopping rule. Accordingly, the algorithm terminates when the variance $\sigma_j^2(\delta)$ drops below the variance tolerance a. The tolerance a is decreased during the iteration, i.e., if a new solution is found, a is set to a fraction $p < 1$ of the current variance $\sigma_j^2(\delta)$. If the convergence parameter p is small ($p \to 0$), the algorithm performs a thorough search of the domain, whereas larger values of p ($p \to 1$) lead to earlier termination, which might be premature. In fact, the efficiency of the algorithm depends on how quickly it finds all the solutions (the more solutions are found in the late steps of the algorithm, the more the variance tolerance a becomes smaller, and the more iterations are needed for the algorithm to terminate). A favorable situation is when at the first iteration the algorithm finds most of the solutions. The probability of finding the remaining solutions within a few iteration steps (configurations) is high, meaning the variance tolerance a will not be significantly reduced. Therefore, for the set of starting points $\widehat{S}_0$, we expect the algorithm to potentially complete more quickly.

Some other distinctive features of the multistart algorithm, including the starting point validation, generation of sampling points, local optimization methods, and the estimation of the posterior covariance matrix are outlined below.

1. *Starting point validation*. The validation criterion used at configuration steps $j \geq 2$ is somewhat conservative and may not be very efficient. Typically, validation is based on the function values at the initial points rather than those obtained after several iterations of the local search. However, the reason for using this criterion is that we want to discard starting points only when we are absolutely sure that the space in their immediate vicinity is less promising. The function tolerance parameter λ_{FV} allows us to control the number of valid starting points.
2. *Generation of sampling points*. In the literature, several sampling methods have been described, including pseudo-random number generators [48,49], chaotic methods [50–52], low discrepancy methods (Halton, Sobol, and Faure sequences), Latin hypercube sampling [53], quasi-oppositional differential evolution [54,55], and centroidal Voronoi tessellation [56] are used as sampling methods. For our implementation, we focused on pseudo-random number generators, low discrepancy methods, Latin hypercube sampling, and centroidal Voronoi tessellation. To select an appropriate sampling method, we use the $L_2^\star$ star discrepancy, which is a measure of the uniformity of point distributions in a domain.
3. *Local optimization methods*. The algorithm leverages local optimization techniques from the PORT library [57,58], as well as BFGS [59], conjugate gradient [60,61], scaled conjugate gradient [62], and DQED [63]. These methods can be applied either individually or sequentially; if one method does not achieve the desired reduction in the objective function within a specified tolerance, the next method in the sequence is used.
4. *Posterior covariance matrix*. The posterior covariance matrix at the solution $\mathbf{x}_{\mathrm{S}k}^\delta$ is computed as the inverse of the Hessian matrix of the objective function $\mathcal{F}_\delta$. Under the Gauss–Newton approximation to the Hessian, the computational formula for the covariance matrix is $\widehat{\mathrm{C}}_{\mathrm{x}} = \sigma_\delta^2(\mathrm{K}^T\mathrm{K})^{-1}$, where K is the Jacobian of $\widehat{\mathbf{f}}$ evaluated at $\mathbf{x}_{\mathrm{S}k}^\delta$. The key assumption underlying this estimate is that approximating $\mathcal{F}_\delta$ by a second-order Taylor expansion introduces negligible error in the covariance matrix estimation. This assumption holds if the function $\widehat{\mathbf{f}}$ is not overly nonlinear. If the function is highly nonlinear, Monte Carlo techniques may be required to compute a more realistic covariance matrix. In such cases, given a data vector $\widehat{\mathbf{y}}^\delta$, we compute a set of solutions

$\{\mathbf{x}_{\mathrm{S}k}^{\delta}\}_{k=1}^{N_{\mathrm{S}}}$ using the multistart algorithm, and form N_{S} clusters centered at each solution $\mathbf{x}_{\mathrm{S}k}^{\delta}$. Noisy data vectors are then generated with noise $\widehat{\boldsymbol{\delta}} \sim \mathcal{N}(\mathbf{0}, \sigma_{\delta}^2 \mathrm{I}_m)$ centered around $\hat{\mathbf{y}}^{\delta}$. The multistart algorithm is applied to each noisy data vector, and each solution corresponding to a noisy data vector is assigned to the cluster whose center is closest, using Euclidean distance as the distance metric. The solutions within each cluster centered at $\mathbf{x}_{\mathrm{S}k}^{\delta}$ are then used to compute the estimated solution $\mathbb{E}\{\mathbf{x}_{\mathrm{S}k}^{\delta}\}$ and the covariance matrix $\widehat{\mathsf{C}}_{\mathbf{x}}(\mathbf{x}_{\mathrm{S}k}^{\delta})$.

In summary, the proposed inversion algorithm involves the following steps:

1. *Surrogate Model Approximation*: Replace the forward model $\mathbf{F}(\mathbf{x})$ with a surrogate function $\mathbf{f}(\mathbf{x})$, trained on an exact dataset $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$, where $\mathbf{y}_i = \mathbf{F}(\mathbf{x}_i)$ and N is the number of samples. This approximation is achieved using a neural network approach.
2. *Estimation of Surrogate Approximation Error*: Assuming that the surrogate approximation error follows a normal distribution $\boldsymbol{\delta}_{\mathrm{A}} \sim \mathcal{N}(\mathbf{0}, \mathsf{C}_{\delta\mathrm{A}})$, estimate the covariance matrix $\mathsf{C}_{\delta\mathrm{A}}$ by computing the conditional average covariance matrix over a test set $D_{\mathrm{T}} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N_{\mathrm{T}}}$, i.e., $\mathsf{C}_{\delta\mathrm{A}} \approx \mathbb{E}(\mathsf{C}_{\delta\mathrm{A}}|D_{\mathrm{T}}) = (1/N_{\mathrm{T}}) \sum_{i=1}^{N_{\mathrm{T}}} (\mathbf{e}_i - \bar{\mathbf{e}})(\mathbf{e}_i - \bar{\mathbf{e}})^T$, where $\mathbf{e}_i = \mathbf{y}_i - \mathbf{f}(\mathbf{x}_i)$ with $\mathbf{y}_i = \mathbf{F}(\mathbf{x}_i)$ is the error for the ith test point, $\bar{\mathbf{e}} = (1/N) \sum_{i=1}^{N} \mathbf{e}_i$ is the sample mean, and N_{T} is the number of test points.
3. *Total Data Uncertainty Model*: Under the assumption that the measurement noise follows a normal distribution $\boldsymbol{\delta}_{\mathrm{N}} \sim \mathcal{N}(\mathbf{0}, \mathsf{C}_{\delta\mathrm{N}})$, compute the covariance matrix C_{δ} of the total uncertainty in the data vector, given by $\boldsymbol{\delta} = \boldsymbol{\delta}_{\mathrm{N}} + \boldsymbol{\delta}_{\mathrm{A}} \sim \mathcal{N}(\mathbf{0}, \mathsf{C}_{\delta})$, as $\mathsf{C}_{\delta} = \mathsf{C}_{\delta\mathrm{N}} + \mathsf{C}_{\delta\mathrm{A}}$. This leads to the inverse problem $\mathbf{f}(\mathbf{x}) = \mathbf{y}^{\delta}$, where the noisy data is modeled as $\mathbf{y}^{\delta} = \mathbf{f}(\mathbf{x}) + \boldsymbol{\delta}$.
4. *Whitening Transformation*: Compute the average variance of the total error $\sigma_{\delta}^2 = (1/m) \sum_{k=1}^{m} [\mathsf{C}_{\delta}]_{kk}$, and the normalized covariance matrix $\widehat{\mathsf{C}}_{\delta} = (1/\sigma_{\delta}^2)\mathsf{C}_{\delta}$. Apply a whitening transformation using a matrix W, defined in terms of the normalized covariance matrix $\widehat{\mathsf{C}}_{\delta}$, leading to the transformed inverse problem $\widehat{\mathbf{f}}(\mathbf{x}) = \hat{\mathbf{y}}^{\delta}$, where $\widehat{\mathbf{f}}(\mathbf{x}) = \mathsf{W}\mathbf{f}(\mathbf{x})$ and $\hat{\mathbf{y}}^{\delta} = \mathsf{W}\mathbf{y}^{\delta} = \widehat{\mathbf{f}}(\mathbf{x}) + \widehat{\boldsymbol{\delta}}$ with $\widehat{\boldsymbol{\delta}} = \mathsf{W}\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \sigma_{\delta}^2 \mathrm{I}_m)$.
5. *Generation of Starting Points*: Use Algorithm 2 to compute a set of starting points $\widehat{S}_0 = \{\hat{\mathbf{x}}_{0k}\}_{k=1}^{\widehat{N}_0}$ that yield a set of solutions close to the set of all solutions of the solvable equation $\widehat{\mathbf{f}}(\mathbf{x}) = \hat{\mathbf{y}}$, where $\hat{\mathbf{y}} \in \mathcal{R}(\widehat{\mathbf{f}})$ is the exact data vector corresponding to $\widehat{\mathbf{f}}$.
6. *Solution of the Inverse Problem*: Determine all solutions to the inverse problem with noisy data $\widehat{\mathbf{f}}(\mathbf{x}) = \hat{\mathbf{y}}^{\delta}$ using the multistart algorithm with enhanced starting point generation 4.
7. *Uncertainty Quantification*: Compute the posterior covariance matrix at the solutions using (i) an analytical method based on a second-order Taylor expansion of the objective function combined with a Gauss–Newton approximation to the Hessian, or (ii) a Monte Carlo approach.

3. Application to TIRM

In a TIRM experiment, we consider a prolate spheroid with semi-major and semi-minor axes of 2.0 μm and 1.0 μm, respectively, and for the state vector $\mathbf{x} = [d, \alpha, \beta]^T$ compute the forward model $\mathbf{F} = [I_M, \alpha_M, \varepsilon_M]^T$ using the light scattering methodology described in Ref. [8]. The box X is chosen as $X = [0, 0.1] \times [0°, 180°] \times [60°, 120°]$, where d is in micrometers, and α and β are in degrees.

Algorithm 4 Multistart algorithm with enhanced starting point generation.

1. *Inputs:* Initial and final absolute function tolerances $\varepsilon_{\mathrm{F}\min}$ and $\varepsilon_{\mathrm{F}\max}$, respectively, the ratio c_{ε} defining the geometric sequence of absolute function tolerances, whitened data vector $\hat{\mathbf{y}}^{\delta} = \mathsf{W}\mathbf{y}^{\delta}$, starting points $\widehat{S}_0$ delivered by the starting points generation algorithm, number of starting points N_0 to be used when the set $\widehat{S}_0$ is not utilized, maximum configurations of sample points N_{C}, parameter $p < 1$ that controls the variance tolerance for the stopping criterion, and function tolerance parameter λ_{FV} for the validation step.
2. *Initialization of Absolute Function Tolerance Loop*: Set $\varepsilon_{\mathrm{F}} = \varepsilon_{\mathrm{F}\min}$.
3. *Absolute Function Tolerance Loop:*
While $\varepsilon_{\mathrm{F}} \leq \varepsilon_{\mathrm{F}\max}$ do:

- *Initialization of Starting Point Configuration Loop*: Set $a = 0$, $N_{0\mathrm{P}} = 0$, and $X_{\mathrm{S}}^{\delta} = \emptyset$, where a represents the variance tolerance, $N_{0\mathrm{P}}$ is the total number of sample points at the current iteration, and X_{S}^{δ} is the set of solutions.
- *Starting Point Configuration Loop*:
For each configurations j, $j = 1, \ldots, N_{\mathrm{C}}$ do:
  - *Select Starting Points*: If $j = 1$ choose the set of starting points $S_0 = \{\mathbf{x}_{0k}\}_{k=1}^{N_0}$ as $S_0 = \widehat{S}_0$, and set $N_{0\mathrm{P}j} = \widehat{N}_0$ and $N_{0j} = \widehat{N}_0$; else, choose $S_0 = \{\mathbf{x}_{0k}\}_{k=1}^{N_0}$ by sampling $N_{0\mathrm{P}j}$ points from X_2, where $\mu(X_2) = 2\mu(X)$, until N_0 points fall in X, and set $N_{0j} = N_0$.
  - *Compute Sample Statistics*: Update the total number of sample points after j iterations, i.e., $N_{0\mathrm{P}} = N_{0\mathrm{P}} + N_{0\mathrm{P}j}$, and compute the quantity $\delta_j = jN_{0j}/N_{0\mathrm{P}}$, its expectation value $\langle \delta \rangle_j = (1/j) \sum_{i=1}^{j} \delta_i$, and its variance $\sigma_j^2(\delta) = \langle \delta^2 \rangle_j - \langle \delta \rangle_j^2$.
  - *Validation of Starting Points*: If $j = 1$, all starting points are considered to be valid; else, use the following validation criterion: Perform an incomplete local search of 1 or 2 steps for each starting point $\mathbf{x}_{0k}$ in the set S_0, and compute $\mathcal{F}_{\delta\max} = \max_k \mathcal{F}_{\delta k}$ and $\mathcal{F}_{\delta\min} = \min_k \mathcal{F}_{\delta k}$, where $\mathcal{F}_{\delta k}$ is the value of the noise-affected objective function at the end of the local search, corresponding to $\mathbf{x}_{0k}$. If $\mathcal{F}_{\delta k} \leq \mathcal{F}_{\delta\min} + \lambda_{\mathrm{FV}}(\mathcal{F}_{\delta\max} - \mathcal{F}_{\delta\min})$, then $\mathbf{x}_{0k}$ is considered valid; otherwise, it is not considered valid.
  - *Solution Search*:
    - Initialize $NewSolution$ = false.
    - For each starting points $\mathbf{x}_{0k} \in S_0$, $k = 1, \ldots, N_0$ do:
      - If $\mathbf{x}_{0k}$ is a valid starting point, perform a local search with $\mathbf{x}_{0k}$ as the initial point. If the local search is successful (i.e., the absolute function convergence criterion is met with tolerance ε_{F}) and the solution $L(\hat{\mathbf{y}}^{\delta} \mid \mathbf{x}_{0k}) \notin X_{\mathrm{S}}^{\delta}$, then update X_{S}^{δ} to include this new solution, i.e., $X_{\mathrm{S}}^{\delta} = X_{\mathrm{S}}^{\delta} \cup \{L(\hat{\mathbf{y}}^{\delta} \mid \mathbf{x}_{0k})\}$, and set $NewSolution$ = true.
  - *Adjust Tolerance*: If $NewSolution$ = true, set $a = p\sigma_j^2(\delta)$.
  - *Convergence Check*: If $\sigma_j^2(\delta) < a$, then exit from the configuration loop (the j loop) and return the set of solutions X_{S}^{δ} as the output.
- *Check Solutions:* If at least one minimum is detected in X_{S}^{δ}, i.e., if $N_{\mathrm{S}}^{\delta} > 1$, terminate the tolerance loop; else, update the tolerance as $\varepsilon_{\mathrm{F}} = c_{\varepsilon}\varepsilon_{\mathrm{F}}$.

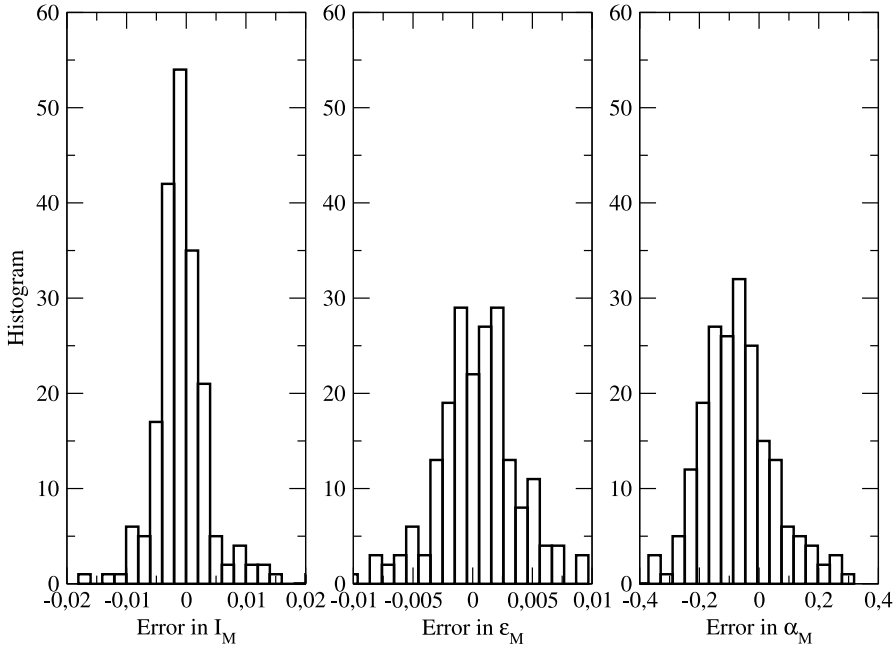

Fig. 4. Histograms of the errors in the integrated intensity I_M, the morphology aspect ratio ε_M, and the morphology orientation angle α_M.

3.1. Neural network-based surrogate model

The forward model is approximated by a surrogate function using a neural network approach. For this purpose, we use $N = 50000$ data points randomly generated using a Sobol sequence, with 45,000 samples for training and the remaining 5000 for validation. We employ a feed-forward multilayer perceptron architecture with four hidden layers containing 32, 64, 64, and 32 neurons, respectively. The Exponential Linear Unit (ELU) activation function is used in the hidden layers, while a linear activation function is used in the output layer. Optimization is performed using mini-batch gradient descent with adaptive moment estimation (ADAM) and a mini-batch size of 250 samples.

To estimate the approximation error, we consider a test set $\mathcal{D}_\mathrm{T} = \{(\mathbf{x}_i, \mathbf{y}_i = \mathbf{F}(\mathbf{x}_i))\}_{i=1}^{N_\mathrm{T}}$ with $N_\mathrm{T} = 200$ data points. The histograms of each component of the error vector $\mathbf{e}_i = \mathbf{y}_i - \mathbf{f}(\mathbf{x}_i)$, $i = 1, \ldots, N_\mathrm{T}$ are shown in Fig. 4. The covariance matrix of the approximation error is estimated as the conditional average covariance matrix over the test set $\mathcal{D}_\mathrm{T}$, yielding

$$\mathsf{C}_{\delta\mathrm{A}} \approx (1/N_\mathrm{T}) \sum_{i=1}^{N_\mathrm{T}} (\mathbf{e}_i - \bar{\mathbf{e}})(\mathbf{e}_i - \bar{\mathbf{e}})^T = \begin{bmatrix} 1.997\times 10^{-5} & 1.988\times 10^{-6} & -8.162\times 10^{-7} \\ 1.988\times 10^{-6} & 1.104\times 10^{-5} & -1.803\times 10^{-7} \\ -8.162\times 10^{-7} & -1.803\times 10^{-7} & 4.535\times 10^{-6} \end{bmatrix},$$

where

$$\bar{\mathbf{e}} = (1/N_\mathrm{T}) \sum_{i=1}^{N_\mathrm{T}} \mathbf{e}_i = [-7.339\times 10^{-4};\ 4.507\times 10^{-4};\ -1.319\times 10^{-3}]^T$$

is the sample mean of the errors. Furthermore, the average variance of the approximation error over the test set is $\sigma_{\delta\mathrm{A}}^2 = (1/m)\sum_{k=1}^{m} [\mathsf{C}_{\delta\mathrm{A}}]_{kk} = 1.185\times 10^{-5}$ and the average relative error of the approximation error is

$$\varepsilon_{\delta\mathrm{A}} = \frac{1}{N_\mathrm{T}} \sum_{i=1}^{N_\mathrm{T}} \frac{\|\mathbf{e}_i\|}{\|\mathbf{y}_i\|} = 2.462\times 10^{-3}.$$

In Fig. 5 we illustrate the morphology quantities computed with the neural network model. The results show that (i) the morphology orientation angle α_M depends strongly on the azimuthal particle orientation angle α and weakly on the polar particle orientation angle β, (ii) the morphology aspect ratio ε_M depends on the particle orientation angles (α, β), and (iii) the integrated intensity I_M depends on both the separation distance d and the particle orientation angles (α, β).

3.2. Measurement noise and whitening transformation

The measurement noise in each component of the output $\mathbf{y} = \mathbf{F} = [I_M, \alpha_M, \varepsilon_M]^T$, where I_M, α_M, and ε_M are the integrated signal, the morphology orientation angle, and the aspect ratio observed in an image, respectively, is estimated as follows.

1. The measurement noise in the integrated signal is assumed to be Gaussian with zero mean. For a state vector $\mathbf{x}_i$ in the test set $\mathcal{D}_\mathrm{T}$, the noise variance of the integrated signal $I_{M,i} = [\mathbf{F}(\mathbf{x}_i)]_1$ is estimated from the signal-to-noise ratio (SNR), defined as the ratio of the signal amplitude to the standard deviation (and not as the ratio of their power):

   $$\mathrm{SNR} = \frac{I_{M,i}}{\sigma_{\delta\mathrm{N1},i}}$$

   By choosing SNR = 100, which is a typical value for a TIRM experiment, we find that the average variance of the integrated signal over the test set is $\sigma_{\delta\mathrm{N1}}^2 = (1/N_\mathrm{T})\sum_{i=1}^{N_\mathrm{T}} \sigma_{\delta\mathrm{N1},i}^2 = 2.970\times 10^{-5}$.
2. Accurate distance and angle measurements using a CCD camera are influenced by instrumental noise, which originates from multiple sources, including readout noise, photon shot noise, and optical aberrations. In many practical applications, particularly when working with well-calibrated imaging systems under stable lighting conditions, the noise affecting the detected pixel positions can be approximated as Gaussian and isotropic. This assumption is valid when the positional errors in both x and y coordinates follow independent normal distributions with the same variance, i.e., $\epsilon_\mathrm{x}, \epsilon_\mathrm{y} \sim \mathcal{N}(0, \sigma_\mathrm{CCD}^2)$ for $x^\delta = x + \epsilon_\mathrm{x}$ and $y^\delta = y + \epsilon_\mathrm{y}$. Such conditions are typically met when the sensor resolution is high, the point spread function is approximately symmetric, and systematic distortions are either negligible or well-corrected through calibration. Under these circumstances,

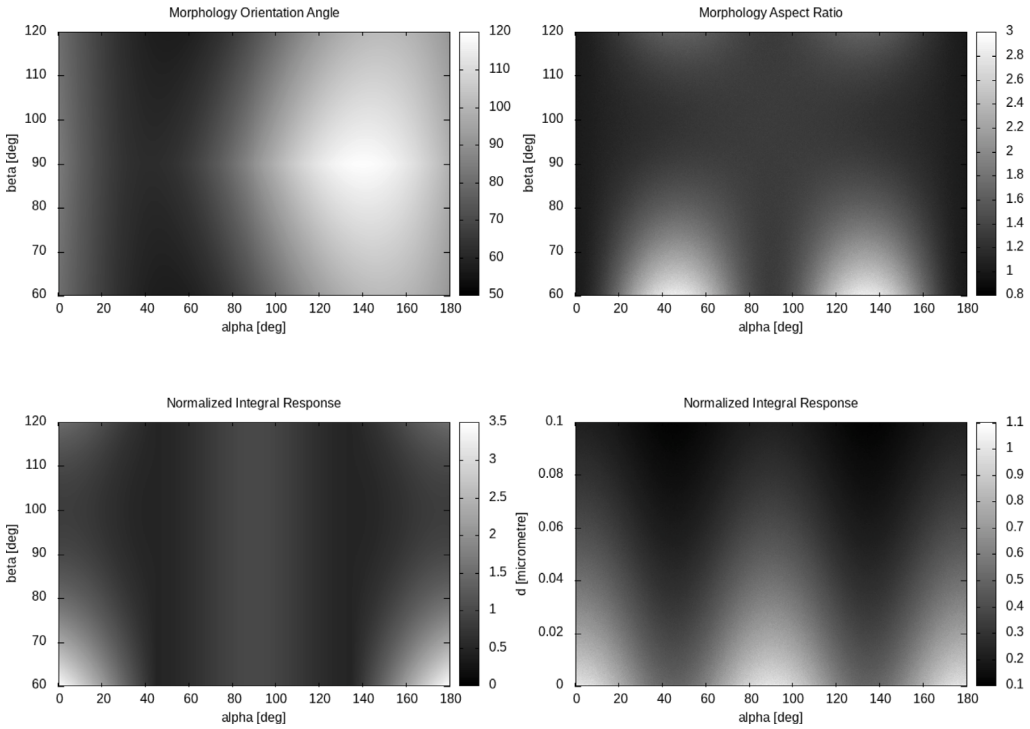

Fig. 5. Upper panels: The morphology orientation angle $\alpha_M(\alpha,\beta)$ and aspect ratio $\varepsilon_M(\alpha,\beta)$. Lower panels: The normalized integrated intensity $\overline{I}_M(d=0,\alpha,\beta)$ (left) and $\overline{I}_M(d,\alpha,\beta=0)$ (right), where $\overline{I}_M(d,\alpha,\beta)=I_M(d,\alpha,\beta)/I_M(d=0,\alpha=0,\beta=0)$.

the statistical properties of derived geometric quantities, such as the angle between two points with respect to a reference and the ratio of two distances can be analyzed using standard error propagation techniques. The variance of the angle $\alpha_M = \arctan(y_2-y_1)/(x_2-x_1)$ between the two points (x_1,y_1) and (x_2,y_2) with respect to a reference and the variance of the aspect ratio $\varepsilon_M = d_1/d_2$ of two measured distances d_1 and d_2, are given by

$$\sigma^2_{\delta\mathrm{N2}} = \mathrm{Var}(\alpha_M) = \frac{2\sigma^2_{\mathrm{CCD}}}{d^2},$$

and

$$\sigma^2_{\delta\mathrm{N3}} = \mathrm{Var}(\varepsilon_M) = \frac{2\sigma^2_{\mathrm{CCD}}}{d_2^2}(1+\varepsilon_M^2),$$

respectively, where $d = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$ represents the characteristic length scale of the measurement. For a conventional CCD camera with a typical positional noise of $\sigma_{\mathrm{CCD}} \approx 0.2$ pixels, the variances of α_M and ε_M can be estimated for practical scenarios. Assuming distances of approximately 10 to 30 pixels and aspect ratios in the range $1 \le \varepsilon_M \le 3$, the variance of α_M is found to be in the range $\mathrm{Var}(\alpha_M) \approx 8.9\times10^{-5}$ to 8×10^{-4}, while the variance of ε_M ranges between $\mathrm{Var}(\varepsilon_M) \approx 8\times10^{-4}$ to 8×10^{-3}.

Choosing $\sigma^2_{\delta\mathrm{N2}} = \sigma^2_{\delta\mathrm{N3}} = 8\times10^{-4}$, we deduce that the measurement noise $\boldsymbol{\delta}_{\mathrm{N}}$ follows a Gaussian distribution with zero mean and covariance matrix $\mathsf{C}_{\delta\mathrm{N}} = \sigma^2_{\delta\mathrm{N}}\mathrm{I}_m$, i.e., $\boldsymbol{\delta}_{\mathrm{N}} \sim \mathcal{N}(\mathbf{0},\mathsf{C}_{\delta\mathrm{N}})$, where the measurement noise variance is given by $\sigma^2_{\delta\mathrm{N}} = \sigma^2_{\delta\mathrm{N1}}+\sigma^2_{\delta\mathrm{N2}}+\sigma^2_{\delta\mathrm{N3}} = 1.629\times10^{-3}$. Comparing the variances of the measurement noise and the approximation error, we observe that the measurement noise is the dominant source of uncertainty, with its variance being an order of magnitude larger than that of the approximation error. Consequently, omitting the forward model parameter error $\boldsymbol{\delta}_{\mathrm{B}}$, we find that (i) the total covariance matrix

$$\mathsf{C}_\delta = \begin{bmatrix} 1.649\times10^{-3} & 1.988\times10^{-6} & -8.162\times10^{-7} \\ 1.988\times10^{-6} & 1.640\times10^{-3} & -1.803\times10^{-7} \\ -8.162\times10^{-7} & -1.803\times10^{-7} & 1.634\times10^{-3} \end{bmatrix}$$

is diagonally dominant, (ii) the average variance of the total error $\sigma^2_\delta = (1/m)\sum_{k=1}^m [\mathsf{C}_\delta]_{kk} = 1.641\times10^{-3}$ is close to the measurement noise variance, and (iii) the normalized total covariance matrix

$$\widehat{\mathsf{C}}_\delta = \begin{bmatrix} 1.005 & 1.211\times10^{-3} & -4.972\times10^{-4} \\ 1.211\times10^{-3} & 9.995\times10^{-1} & -1.098\times10^{-4} \\ -4.972\times10^{-4} & -1.098\times10^{-4} & 9.955\times10^{-1} \end{bmatrix}$$

as well as the whitening matrix, obtained through a Cholesky factorization of $\widehat{\mathsf{C}}_\delta$,

$$\mathsf{W} = \begin{bmatrix} 9.950\times10^{-1} & 0 & 0 \\ -1.205\times10^{-3} & 1.001 & 0 \\ 4.968\times10^{-4} & 1.098\times10^{-4} & 1.004 \end{bmatrix}$$

are both close to the identity matrix.

Since $\mathsf{W} \approx \mathrm{I}_m$, we can practically assume that transformed inverse problem $\widehat{\mathbf{f}}(\mathbf{x}) = \widehat{\mathbf{y}}^\delta$ is nearly identical to the original one $\mathbf{f}(\mathbf{x}) = \mathbf{y}^\delta$, where $\mathbf{y}^\delta = \mathbf{f}(\mathbf{x}) + \boldsymbol{\delta}$ with $\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \mathsf{C}_\delta = \sigma^2_\delta \mathrm{I}_m)$.

3.3. Advanced multistart solver

When applying the advanced multistart solver, the samples are randomly generated using a Sobol sequence, as this method was found to have the lowest $L_2^\star$ star discrepancy among the methods considered. The local search is conducted using the least-squares routine DRN2 GB from the PORT library (operating in reverse communication) with simple bounds on the variables, as it has been found to effectively capture the exact solutions by imposing small function and solution tolerances, ε_{F} and ε_{X}, respectively. The set of starting points is selected as the hitting set with the smallest number of elements. The simulations are performed using the following input parameters:

1. In the starting point generation algorithm (Algorithm 2), we set $N_0 = 200$ for the number of starting points and $N_{\mathrm{S}} = 800$ for the number of solutions. The tolerances ε_{F} and ε_{X} are set to 10^{-20} and 10^{-10}, respectively. The hitting set problem is solved using the greedy algorithm.

2. In the multistart algorithm with enhanced starting point generation (Algorithm 4), the initial and final absolute function tolerances are set to $\epsilon_{\text{Fmin}} = 10^{-16}$ and $\epsilon_{\text{F max}} = 3.5 \times 10^{-2} (\approx \sqrt{m}\sigma_\delta/2)$, respectively, and the factor for increasing the absolute function tolerance to $c_\epsilon = 10$. Additionally, the following input parameters are used: (i) $N_0 = \min(\widehat{N}_0, 10)$, the number of starting points to use if the set generated by the starting point generation algorithm is not utilized, (ii) $N_{\text{C}} = 1000$, the maximum number of sample point configurations, (iii) $p = 0.2$, the variance tolerance parameter for the stopping criterion, and (iv) $\lambda_{\text{FV}} = 0.5$, the function tolerance parameter for the validation step.

In the first stage, the starting point generation algorithm identifies $\widehat{N}_0 = 9$ points (from the set of $N_0 = 200$ starting points) to be used in the multistart algorithm with enhanced starting point generation. It should be pointed out that the choice $N_0 = \min(\widehat{N}_0, 10)$, which specifies the number of starting points to be used when the points delivered by the starting point generation algorithm are not utilized, is intended to reduce the computational time when $\widehat{N}_0 > 10$ (as a result, the acceleration factor is larger than that corresponding to the choice $N_0 = \widehat{N}_0$). In Fig. 6, we illustrate the injectivity domains projected onto the $\alpha\beta$-plane. With the exception of a few points from one injectivity domain that lie within another (a situation that may occur, as mentioned above), the projected domains are disjoint. Since two sets of points in three-dimensional space are disjoint if at least one of their projections onto the Cartesian coordinate planes is disjoint, we can conclude that the injectivity domains are disjoint in three-dimensional space.

In the second stage, for $N_{\text{Y}} = 200$ data vectors organized in the set $\mathcal{D}_{\text{Y}} = \{(\mathbf{x}_i, \mathbf{y}_i = \mathbf{f}(\mathbf{x}_i))\}_{i=1}^{N_{\text{Y}}}$ we apply Algorithm 4 to solve the equation $\mathbf{f}(\mathbf{x}) = \mathbf{y}$ for the exact data vector $\mathbf{y} \in \mathcal{R}(\mathbf{f})$. Additionally, for the associated noisy data set $\mathcal{D}_{\text{Y}}^\delta = \{(\mathbf{x}_i, \mathbf{y}_i^\delta = \mathbf{y}_i + \boldsymbol{\delta})\}_{i=1}^{N_{\text{Y}}}$, where $\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \mathsf{C}_\delta = \sigma_\delta^2 \mathrm{I}_m)$, we solve the inverse problem $\mathbf{f}(\mathbf{x}) = \mathbf{y}^\delta$ using the same algorithm. For the exact data, the algorithm identifies one solutions 73 times, two solutions 89 times, three solution 24 times, and four solutions 14 times out of the total 200 runs. We represent this set of solutions as $\{73 \times 1;\ 89 \times 2;\ 24 \times 3;\ 14 \times 4\}$. For the noisy data, the set of solutions is $\{70 \times 1;\ 88 \times 2;\ 24 \times 3;\ 14 \times 4;\ 1 \times 5;\ 1 \times 6;\ 1 \times 7;\ 1 \times 8\}$.

In Fig. 7 we illustrate the number of noisy solutions N_{S}^δ compared to the number of exact solutions N_{S}. The results indicate that for a pair $(\mathbf{y}_i, \mathbf{y}_i^\delta)$, where $i = 1, 2, \ldots, N_{\text{Y}}$, the numbers of solutions can differ. Specifically, the situation $N_{\text{S}}^\delta = N_{\text{S}}$ occurs in 138 cases, the situation $N_{\text{S}}^\delta < N_{\text{S}}$ occurs in 28 cases, and the situation $N_{\text{S}}^\delta > N_{\text{S}}$ occurs in 34 cases. As an example, Fig. 8 illustrate the distributions of the exact and noisy solutions in the cases $N_{\text{S}}^\delta = N_{\text{S}}$, $N_{\text{S}}^\delta < N_{\text{S}}$, and $N_{\text{S}}^\delta > N_{\text{S}}$. In fact, the final absolute function tolerance serves as an indication of the type of minima found. Global minima correspond to $\epsilon_{\text{F}} = 10^{-16}$, while local minima correspond to significantly larger values of ϵ_{F}. For example, in the case of exact data, all solutions were found with $\epsilon_{\text{F}} = 10^{-16}$, while in the case of noisy data, local minima were found with tolerances $\epsilon_{\text{F}} = 10^{-7}$ once, $\epsilon_{\text{F}} = 10^{-5}$ thirteen times, $\epsilon_{\text{F}} = 10^{-4}$ eighteen times, and $\epsilon_{\text{F}} = 10^{-3}$ nine times; the remaining 159 solutions were global minima, all found with $\epsilon_{\text{F}} = 10^{-16}$.

Because the function $\mathbf{f}$ is nonlinear, a Monte-Carlo analysis is required to compute a realistic posterior covariance matrix. The example illustrated in Fig. 9 corresponds to a noisy data vector $\mathbf{y}^{bold\,symbol\delta}$ with $N_{\text{S}}^\delta = 2$ solutions, denoted as $\{\mathbf{x}_{\text{S}k}^\delta\}_{k=1}^{N_{\text{S}}^\delta}$. We generate 200 noisy data vectors with noise $\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \mathsf{C}_\delta = \sigma_\delta^2 \mathrm{I}_m)$ centered around $\widehat{\mathbf{y}}^\delta$, define N_{S}^δ clusters centered at $\mathbf{x}_{\text{S}k}^\delta$, apply the multistart algorithm for each of the 200 noisy data vectors, assign each solution to the cluster with the nearest center, and use all solutions within a cluster to compute the estimated solutions $\mathbb{E}\{\mathbf{x}_{\text{S}k}^\delta\}$ and the covariance matrices $\widehat{\mathsf{C}}_{\text{x}}(\mathbf{x}_{\text{S}k}^\delta)$ for $k = 1, 2$. The relative errors in the expected solutions $\mathbb{E}\{\mathbf{x}_{\text{S}k}^\delta\}$ compared to the noisy solutions $\mathbf{x}_{\text{S}k}^\delta$ are 2.5×10^{-3} for the first solution and 7.8×10^{-4} for the second solutions. The standard deviations in the solution components (d, α, β) are $(1.8 \times 10^{-3}, 2.6°, 1.2°)$ and $(5.8 \times 10^{-3}, 1.5°, 1.0°)$, respectively. On the other hand, the standard deviations computed with the analytical formula $\widehat{\mathsf{C}}_{\text{x}} = \sigma_\delta^2 (\mathsf{K}^T \mathsf{K})^{-1}$, considering a second-order Taylor expansion of the objective functions, are $(2.6 \times 10^{-5}, 19.3°, 9.4°)$ and $(8.2 \times 10^{-5}, 11.5°, 8.7°)$, respectively, which, as shown in Fig. 9, seem unrealistic.

4. Conclusions

We developed an algorithm for solving inverse problems that involve finding all solutions to the equation $\mathbf{F}(\mathbf{x}) = \mathbf{y}$, where $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^m$ represents a nonlinear and continuous forward model, and $\mathbf{y} \in \mathbb{R}^m$ is the data vector. The function $\mathbf{F}$ is computationally expensive, lacks available derivatives, and is treated as a black-box function. Furthermore, there is no prior information about the smoothness or statistical properties of the state vector $\mathbf{x} \in \mathbb{R}^n$ except for the box constraint $X = [a_1, b_1] \times [a_2, b_2] \ldots \times [a_n, b_n]$, while the problem is further influenced by uncertainties in $\mathbf{y}$. Specifically, the algorithm includes:

1. A neural network-based surrogate model to reduce computational cost and handle the lack of derivatives.
2. Formulating an inverse problem with isotropic noise by applying a whitening transformation and solving it by minimizing the residual with simple bounds.
3. An advanced multistart solver to find all solutions of the inverse problem, addressing the nonlinearity of the forward model.

Regarding the first item, we note that Appendix covers the mathematical foundation for the development of retrieval algorithms based on neural networks from a statistical perspective.

The advanced multistart solver has several notable features:

1. The proposed methodology consists of two key components:
   (a) Starting Point Generation Algorithm: This algorithm recognizes that a solution can be reached from multiple starting points. It generates a set of starting points by solving a hitting set problem, which improves the performance of the multistart algorithm.
   (b) Multistart Algorithm: This algorithm uses the generated starting points as initial guesses to explore the solution space, increasing the likelihood of finding all possible solutions.
2. The multistart algorithm encompasses the following characteristics:
   (a) The algorithm features an outer loop that iterates over the absolute function tolerance, guaranteeing the identification of multiple global and local minima without premature termination. Global minima, which emerge at smaller absolute function tolerances, occur when the noisy data falls within an injectivity domain of the forward model. In contrast, local minima, detected at larger absolute function tolerances, arise when the noisy data lies outside the range of the forward model.
   (b) During the local search phase, a least-squares function is minimized while adhering to simple bounds on variables.
   (c) A double box stopping rule is employed, along with a validation criterion for starting points that relies on function values obtained after a few iterations of the local search. Additionally, the approach incorporates various sampling techniques and optimization methods.
   (d) To estimate the covariance matrix at the solution, the algorithm either uses a standard analytical approach or conducts a Monte Carlo analysis.

The algorithm was applied to a TIRM experiment to address the following inverse problem: Given the measured morphology quantities

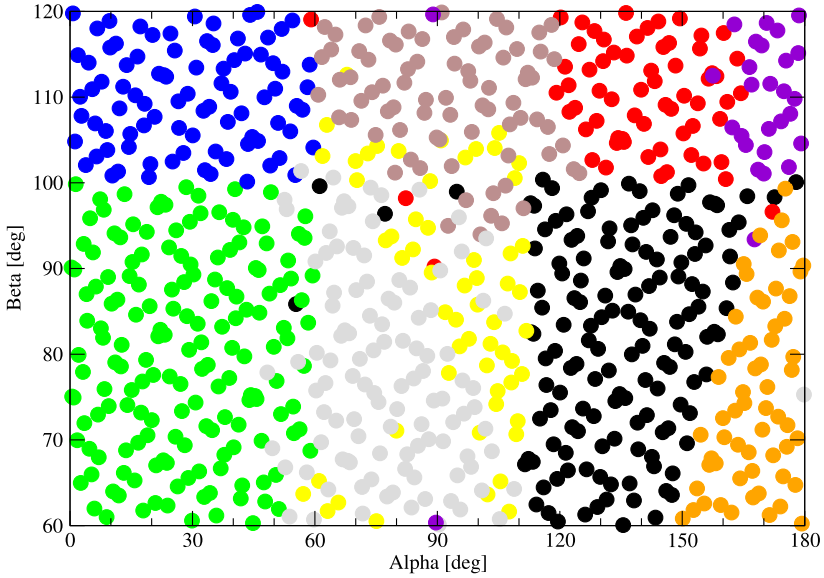

Fig. 6. Injectivity domains projected onto the $\alpha\beta$-plane.

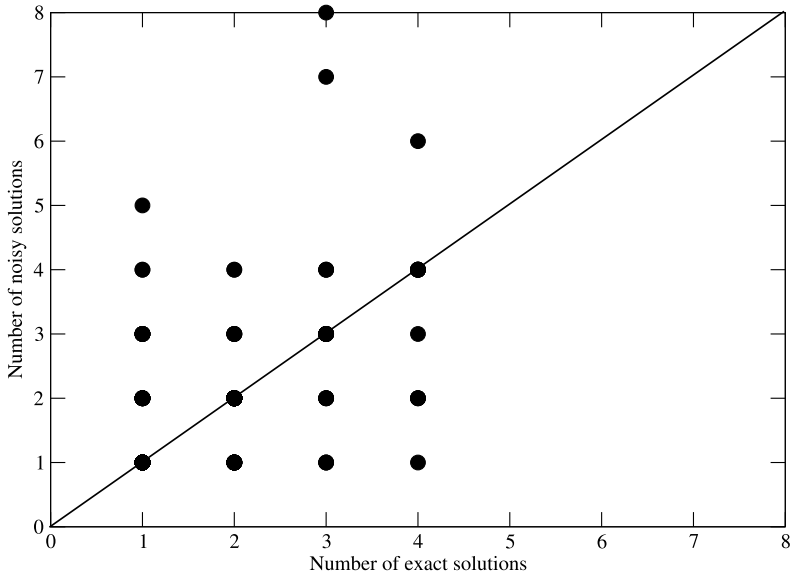

Fig. 7. Number of noisy solutions (corresponding to $\mathbf{y}_i^\delta$) compared to the number of exact solutions (corresponding to $\mathbf{y}_i$) for all $i = 1, 2, \ldots, N_Y$.

$(I_M, \alpha_M, \varepsilon_M)$ representing the integrated signal, the morphology orientation angle, and the aspect ratio ε_M observed in an image, respectively, solve the nonlinear equations $I_M = I_M(d, \alpha, \beta)$, $\alpha_M = \alpha_M(d, \alpha, \beta)$, and $\varepsilon_M = \varepsilon_M(d, \alpha, \beta)$ for the parameters (d, α, β) representing the separation distance and the particle orientation angles, respectively, that are necessary for tracking dynamics. The following results were obtained:

1. The neural-network surrogate model demonstrates exceptional approximation capabilities, with an average relative error of the approximation error as low as 2.5×10^{-3}.
2. For isotropic measurement noise with zero mean and a noise variance derived from a signal-to-noise ratio of 100, the measurement noise significantly outweighs the surrogate approximation error. Consequently, the total covariance matrix is diagonally dominant, and its average variance closely matches that of the measurement noise. Furthermore, both the normalized total covariance matrix and the whitening matrix approach the identity matrix, indicating that the original inverse problem can be solved directly without requiring a whitening transformation.
3. The number of starting points produced by the starting points generation algorithm is 4.5% of the total number of sample points generated by a Sobol sequence.
4. For exact data, the number of solutions ranges between 1 and 4, while for noisy data, it ranges between 1 and 8. In 70% of cases, the numbers of exact and noisy solutions are the same, while in the remaining cases, they differ.
5. Global minima were found with a small absolute function tolerance of 10^{-16}, whereas local minima were identified with a significantly larger tolerance of up to 10^{-3}.
6. The Monte-Carlo method provided a realistic posterior covariance matrix.

We conclude with some final comments:

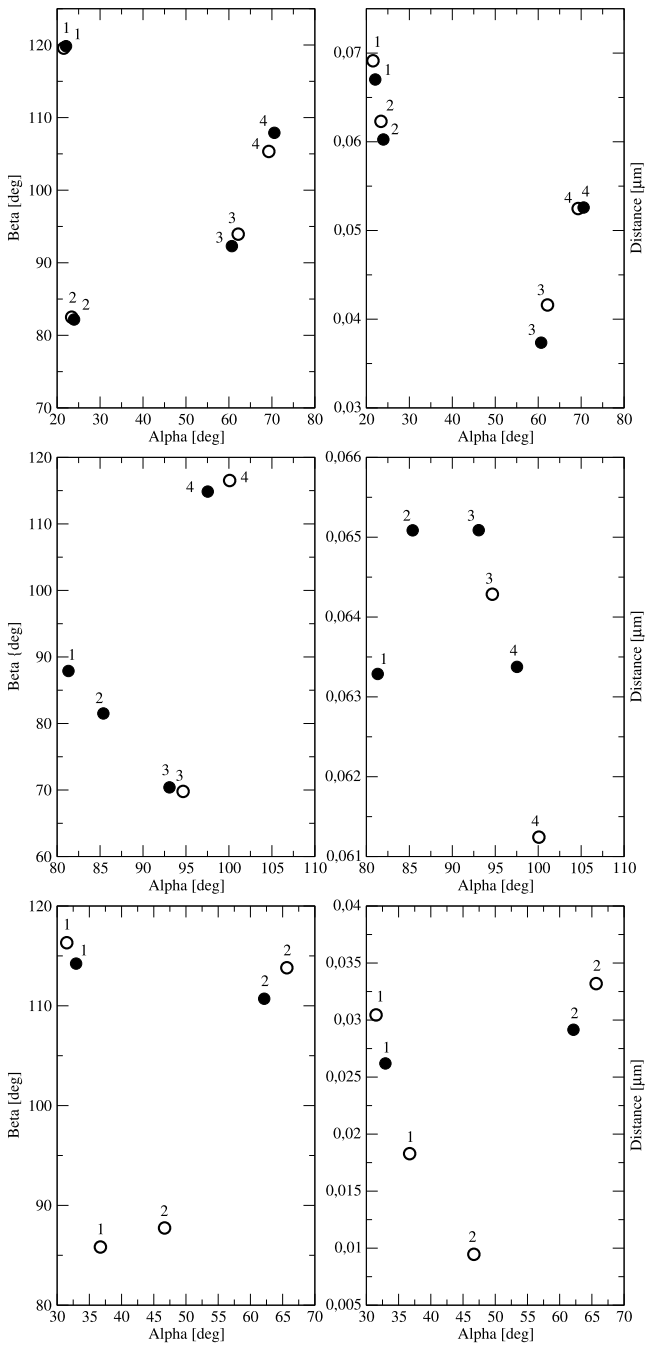

Fig. 8. Exact solutions (filled circles) and noisy solutions (unfilled circles). The upper panel corresponds to $N_S = N_S^\delta = 4$, the middle panel to $N_S = 4$ and $N_S^\delta = 2$, and the lower panel to $N_S = 2$ and $N_S^\delta = 4$. The noisy solutions are connected to the nearest exact solutions (although there are multiple options in their association).

1. It would be interesting to investigate in the future whether information about the orientation angles α and β can be extracted by analyzing the two-dimensional Fourier transform of the CCD signal, specifically the dominant frequency components and their distribution. The direction of the major frequency components corresponds to the azimuthal angle α, reflecting the particle's orientation in the imaging plane. Meanwhile, the relative strength of the frequencies and their distribution along specific axes provide insights into the zenith angle β, as they encode the elongation and projection effects of the prolate spheroid.
2. The algorithm is efficient for low-dimensional physical inverse problems, particularly when the dimension n is small, such as for $n \leq 10$. This is because the most computationally intensive part of the algorithm is solving the hitting set problem, which is NP-complete. The computational complexity of the hitting set problem grows exponentially with the size of the set collection and the universe of elements. As the number of sets increases, the number of potential combinations to evaluate becomes prohibitively large, leading to high computational costs. While the standard greedy algorithm can handle a reasonable number of sets, the modified version we developed (which ensures minimal hitting sets) becomes increasingly challenging to apply to collections containing more than a few thousand dense sets. Consequently, the algorithm remains efficient and practical for low-dimensional problems, especially for $n \leq 10$. For example, in an n-dimensional space, a Sobol sequence requires approximately 2^n sample points for reasonable coverage. This means that for $n = 10$, 1024 sample points are sufficient for sampling the solution space X_S^δ (in our simulations, we considered $N_S = 800$ samples, meaning that the number of sets in the collection is 800). However, for $n = 20$, the required number of sample points increases to 1,048,576, making the hitting set algorithm impractical without significantly higher computational resources.
3. The designed algorithm has multiple applications.

   (a) For instance, it can be applied in atmospheric remote sensing for retrieving cloud or aerosol optical thickness and top height.

      i. For cloud retrieval, incorporating the mean droplet radius into the retrieval process, alongside optical thickness and cloud top height, offers a more comprehensive understanding of cloud properties. Rather than relying on a fixed assumption for the mean radius, such as the common 10 μm value, this approach helps avoid introducing potential errors and biases into the retrieval process. It allows the algorithm to adapt to varying cloud conditions where the droplet size may differ. Since optical thickness and mean radius are often strongly correlated, with multiple combinations of these parameters potentially resulting in the same spectral response, an advanced multistart algorithm can identify all possible solutions arising from this coupling. The next challenge is to identify and select the most likely, physically meaningful solution, which can be achieved through the use of additional data.

      ii. For aerosol retrieval, incorporating the aerosol size distribution alongside optical depth and altitude improves the accuracy of aerosol property estimation. Aerosols exhibit a bimodal size distribution, with fine and coarse modes, requiring the retrieval of additional parameters: the effective radius of each mode. This approach avoids biases from oversimplifying aerosol properties. As with cloud retrieval, optical depth and aerosol size distribution are strongly coupled, and multiple combinations of these parameters can lead to the same spectral response, with an advanced multistart algorithm providing all possible solutions.

   (b) Other potential applications of the proposed algorithm, which will be explored in a future paper, include electromagnetic scattering inverse problems where different particle characteristics can produce identical scattering patterns. For example, in models of mononuclear cells as coated spheres or homogeneous spheres analyzed via

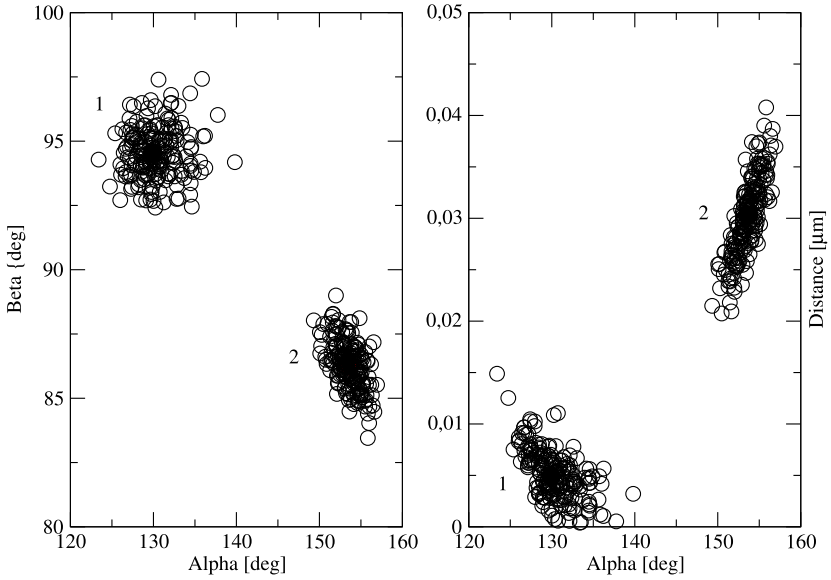

Fig. 9. Monte-Carlo analysis for a noisy data $\mathbf{y}^\delta$, with $N_S^\delta = 2$ identified solutions.

scanning flow cytometry, retrieving both the refractive index and size parameter can result in multiple parameter sets yielding indistinguishable scattering behavior [13, 14].

CRediT authorship contribution statement

Alexandru Doicu: Writing – original draft, Investigation. **Dmitry S. Efremenko:** Writing – original draft, Software. **Christopher L. Wirth:** Writing – review & editing, Conceptualization. **Thomas Wriedt:** Writing – review & editing, Validation.

Declaration of competing interest

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

Acknowledgments

This work was supported by the National Science Foundation CAREER Award, NSF No. 2023525 (CLW)

Appendix

In this appendix, we review the fundamental theory underlying the design of retrieval algorithms using neural networks, following the analysis presented in Ref. [64]. First, we discuss the case of a neural network model with output noise, and then we specialize the obtained results to the case of a neural network model with exact data. Finally, we consider a neural network model that accounts for both output noise and forward model parameter error.

A.1. Neural network model with output noise

The analysis of a neural network model with output noise is conducted within a stochastic framework, focusing on (i) the methodology for computing point estimates, (ii) different types of uncertainty, and (iii) the application of Bayesian regularization to solve the inverse problem.

Point estimates

Consider a data representation that incorporates measurement noise, expressed as

$$\mathbf{y}^\delta = \mathbf{f}(\mathbf{x}, \boldsymbol{\omega}) + \boldsymbol{\delta}_\mathrm{N}, \quad \boldsymbol{\delta}_\mathrm{N} \sim \mathcal{N}(\mathbf{0}, \mathsf{C}_{\delta\mathrm{N}}), \tag{A.1}$$

and a training dataset defined as $\mathcal{D}^\delta = \{(\mathbf{x}_i, \mathbf{y}_i^\delta)\}_{i=1}^N$, where $\mathbf{y}_i^\delta = \mathbf{F}(\mathbf{x}_i) + \boldsymbol{\delta}_\mathrm{N}$ and N is the number of samples. In this context, adding random noise to the output values in the training set, a process known as jittering output data, can help account for measurement uncertainty. In a probabilistic setting, a neural network can be interpreted as a model that defines the probability distribution $p(\mathbf{y}^\delta|\mathbf{x}, \boldsymbol{\omega})$. Given an input $\mathbf{x}$ and a parameter set $\boldsymbol{\omega}$, the network assigns a likelihood to each potential output $\mathbf{y}^\delta$. Based on the assumption in Eq. (A.1), the prior confidence in the predictive power of the model is given by

$$p(\mathbf{y}^\delta|\mathbf{x}, \boldsymbol{\omega}) = \mathcal{N}(\mathbf{f}(\mathbf{x}, \boldsymbol{\omega}), \mathsf{C}_{\delta\mathrm{N}}). \tag{A.2}$$

Learning from the dataset $\mathcal{D}^\delta$ is characterized by the posterior distribution $p(\boldsymbol{\omega}|\mathcal{D}^\delta) = p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}^\delta)$, where $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ and $\mathbf{Y}^\delta = \{\mathbf{y}_i^\delta\}_{i=1}^N$. This posterior captures the plausibility for the parameters $\boldsymbol{\omega}$ given the data $\mathcal{D}^\delta$, and can be estimated using the Bayesian theorem

$$p(\boldsymbol{\omega}|\mathcal{D}^\delta) = \frac{p(\mathcal{D}^\delta|\boldsymbol{\omega})p(\boldsymbol{\omega})}{p(\mathcal{D}^\delta)} \propto p(\mathcal{D}^\delta|\boldsymbol{\omega})p(\boldsymbol{\omega}) \propto \exp[-\mathcal{E}(\boldsymbol{\omega})]. \tag{A.3}$$

In Eq. (A.3), $p(\mathcal{D}^\delta|\boldsymbol{\omega})$ is the likelihood or the probability of the data, $p(\boldsymbol{\omega})$ is the prior over the network parameters, $p(\mathcal{D}^\delta) = \int p(\mathcal{D}^\delta|\boldsymbol{\omega})p(\boldsymbol{\omega})\,\mathrm{d}\boldsymbol{\omega}$ the evidence, and

$$\mathcal{E}(\boldsymbol{\omega}) = \mathcal{E}_\mathrm{D}(\boldsymbol{\omega}) + \mathcal{E}_\mathrm{R}(\boldsymbol{\omega}) \tag{A.4}$$

the loss function. The first term $\mathcal{E}_\mathrm{D}(\boldsymbol{\omega})$ in the expression of the loss function $\mathcal{E}(\boldsymbol{\omega})$ arises from the likelihood $p(\mathcal{D}^\delta|\boldsymbol{\omega})$, which factorizes as (cf. Eq. (A.2)):

$$p(\mathcal{D}^\delta|\boldsymbol{\omega}) = p(\mathbf{Y}^\delta|\mathbf{X}, \boldsymbol{\omega}) = \prod_{i=1}^N p(\mathbf{y}_i^\delta|\mathbf{x}_i, \boldsymbol{\omega}) \propto \exp[-\mathcal{E}_\mathrm{D}(\boldsymbol{\omega})], \tag{A.5}$$

$$\mathcal{E}_\mathrm{D}(\boldsymbol{\omega}) = \frac{1}{2}\sum_{i=1}^N [\mathbf{y}_i^\delta - \mathbf{f}(\mathbf{x}_i, \boldsymbol{\omega})]^T \mathsf{C}_{\delta\mathrm{N}}^{-1} [\mathbf{y}_i^\delta - \mathbf{f}(\mathbf{x}_i, \boldsymbol{\omega})]. \tag{A.6}$$

The second term $\mathcal{E}_\mathrm{R}(\boldsymbol{\omega})$ corresponds to the contribution from the prior $p(\boldsymbol{\omega})$, and can, for example, be chosen as a Gaussian distribution

$$p(\boldsymbol{\omega}) = \mathcal{N}(\mathbf{0}, \mathsf{C}_\omega) \propto \exp[-\mathcal{E}_\mathrm{R}(\boldsymbol{\omega})], \tag{A.7}$$

$$\mathcal{E}_\mathrm{R}(\boldsymbol{\omega}) = \frac{1}{2}\boldsymbol{\omega}^T \mathsf{C}_\omega^{-1} \boldsymbol{\omega}. \tag{A.8}$$

Point estimates incorporating regularization are obtained by maximizing the posterior probability $p(\boldsymbol{\omega}|D^\delta)$,

$$\widehat{\boldsymbol{\omega}} = \boldsymbol{\omega}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{\omega}} \log p(\boldsymbol{\omega}|D^\delta) = \arg\min_{\boldsymbol{\omega}} \mathcal{E}(\boldsymbol{\omega}). \tag{A.9}$$

When regularization is not applied, point estimates are derived by maximizing the likelihood function $p(D^\delta|\boldsymbol{\omega})$,

$$\widehat{\boldsymbol{\omega}} = \boldsymbol{\omega}_{\mathrm{MLE}} = \arg\max_{\boldsymbol{\omega}} \log \mathrm{p}(D^\delta|\boldsymbol{\omega}) = \arg\min_{\boldsymbol{\omega}} \mathcal{E}_{\mathrm{D}}(\boldsymbol{\omega}). \tag{A.10}$$

In summary, from a probabilistic standpoint, the loss function represents the negative log-likelihood of the data, often augmented with a regularization term to restrict parameter values. Statistically, this corresponds to maximum likelihood estimation (MLE) in the absence of regularization and to maximum a posteriori (MAP) estimation when regularization is introduced.

A comment can be made here. For $\mathsf{C}_{\delta\mathrm{N}} = \sigma_{\delta\mathrm{N}}^2 \mathrm{I}_m$, we have

$$\mathcal{E}_{\mathrm{D}}(\boldsymbol{\omega}) = \sum_{i=1}^{N} \frac{1}{2\sigma_{\delta\mathrm{N}}^2} \|\mathbf{y}_i^\delta - \mathbf{f}(\mathbf{x}_i, \boldsymbol{\omega})\|^2, \tag{A.11}$$

or more precisely,

$$\mathcal{E}_{\mathrm{D}}(\boldsymbol{\omega}) = \sum_{i=1}^{N} \Big[\frac{1}{2\sigma_{\delta\mathrm{N}}^2} \|\mathbf{y}_i^\delta - \mathbf{f}(\mathbf{x}_i, \boldsymbol{\omega})\|^2 + \frac{m}{2} \log(2\pi\sigma_{\delta\mathrm{N}}^2) \Big]. \tag{A.12}$$

Thus, the maximum likelihood estimate is given by

$$\widehat{\boldsymbol{\omega}} = \boldsymbol{\omega}_{\mathrm{MLE}} = \arg\min_{\boldsymbol{\omega}} \Big[\frac{1}{2} \sum_{i=1}^{N} \|\mathbf{y}_i^\delta - \mathbf{f}(\mathbf{x}_i, \boldsymbol{\omega})\|^2 \Big]. \tag{A.13}$$

Since the normalization constant $(1/\sigma_{\delta\mathrm{N}}^2)$ in the Gaussian likelihood does not depend on $\boldsymbol{\omega}$, it does not affect the optimization when taking the maximum likelihood estimate. Moreover, assuming $\mathsf{C}_\omega = \sigma_\omega^2 \mathrm{I}$ and using Eqs. (A.4), (A.9), and (A.11) we obtain the maximum a posteriori estimate

$$\widehat{\boldsymbol{\omega}} = \boldsymbol{\omega}_{\mathrm{MAP}} = \arg\min_{\boldsymbol{\omega}} \Big[\frac{1}{2} \sum_{i=1}^{N} \|\mathbf{y}_i^\delta - \mathbf{f}(\mathbf{x}_i, \boldsymbol{\omega})\|^2 + \lambda \|\boldsymbol{\omega}\|^2 \Big] \tag{A.14}$$

where $\lambda = \sigma_{\delta\mathrm{N}}^2/(2\sigma_\omega^2)$ is the regularization parameter.

Uncertainties

We consider the estimation of uncertainty associated with the underlying processes. The most precise measure of model uncertainty is the predictive distribution of an unknown output $\mathbf{y}^\delta$ given an input $\mathbf{x}$, which is expressed as

$$p(\mathbf{y}^\delta|\mathbf{x}, D^\delta) = \int p(\mathbf{y}^\delta|\mathbf{x}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|D^\delta)\, \mathrm{d}\boldsymbol{\omega}. \tag{A.15}$$

Knowing $p(\mathbf{y}^\delta|\mathbf{x}, D^\delta)$ allows us to compute the first two moments of the output $\mathbf{y}^\delta$ as follows:

$$\mathbb{E}(\mathbf{y}^\delta) = \int \mathbf{y}^\delta p(\mathbf{y}^\delta|\mathbf{x}, D^\delta)\, \mathrm{d}\mathbf{y}^\delta, \tag{A.16}$$

$$\mathbb{E}(\mathbf{y}^\delta \mathbf{y}^{\delta T}) = \int \mathbf{y}^\delta \mathbf{y}^{\delta T} p(\mathbf{y}^\delta|\mathbf{x}, D^\delta)\, \mathrm{d}\mathbf{y}^\delta. \tag{A.17}$$

From these, the covariance matrix is obtained as

$$\mathrm{Cov}(\mathbf{y}^\delta) = \mathbb{E}(\mathbf{y}^\delta \mathbf{y}^{\delta T}) - \mathbb{E}(\mathbf{y}^\delta)\mathbb{E}(\mathbf{y}^\delta)^T. \tag{A.18}$$

Eq. (A.15) reveals that the predictive distribution $p(\mathbf{y}^\delta|\mathbf{x}, D^\delta)$ depends on the Bayesian posterior $p(\boldsymbol{\omega}|D^\delta)$. However, directly computing $p(\boldsymbol{\omega}|D^\delta)$ using Eq. (A.3) is often infeasible due to the complexity of evaluating the evidence $p(D^\delta) = \int p(D^\delta|\boldsymbol{\omega}) p(\boldsymbol{\omega})\, \mathrm{d}\boldsymbol{\omega}$. To approximate this intractable distribution, two common approaches are:

1. Laplace Approximation, which provides an approximate posterior distribution $p(\boldsymbol{\omega}|D^\delta)$.
2. Variational Inference, which estimates a variational distribution $q_\theta(\boldsymbol{\omega})$ that serves as a surrogate for the true posterior $p(\boldsymbol{\omega}|D^\delta)$.

The choice of method influences the nature of the neural network employed. With the Laplace approximation, a single deterministic estimate of the parameters $\boldsymbol{\omega}$ is learned, leading to conventional (point-estimate) neural networks. In contrast, variational inference captures uncertainty by learning a probability distribution over $\boldsymbol{\omega}$, resulting in stochastic neural networks. A Bayesian neural network can be regarded as a stochastic neural network trained by using Bayesian inference. This type of neural network provides a natural approach to quantify uncertainty in deep learning and allows to distinguish between different types of uncertainty. A mathematical description of the most relevant algorithm used for Bayesian inference, i.e., Bayes-by-backprop, as well as, of two Bayesian approximation methods (dropout and batch normalization) can be found in Ref. [64]. Here, we discuss the Laplace approximation, which is theoretically significant as it offers explicit representations for various types of uncertainty. In this approach, the loss function

$$\mathcal{E}(\boldsymbol{\omega}) = \frac{1}{2} \sum_{i=1}^{N} [\mathbf{y}_i^\delta - \mathbf{f}(\mathbf{x}_i, \boldsymbol{\omega})]^T \mathsf{C}_{\delta\mathrm{N}}^{-1} [\mathbf{y}_i^\delta - \mathbf{f}(\mathbf{x}_i, \boldsymbol{\omega})] + \frac{1}{2} \boldsymbol{\omega}^T \mathsf{C}_\omega^{-1} \boldsymbol{\omega} \tag{A.19}$$

is expanded around the point estimate $\widehat{\boldsymbol{\omega}} = \boldsymbol{\omega}_{\mathrm{MAP}}$ and the optimality condition $\nabla \mathcal{E}(\widehat{\boldsymbol{\omega}}) = \mathbf{0}$ is used. The result is

$$\mathcal{E}(\boldsymbol{\omega}) = \mathcal{E}(\widehat{\boldsymbol{\omega}}) + \frac{1}{2} (\boldsymbol{\omega} - \widehat{\boldsymbol{\omega}})^T \mathsf{H}(\widehat{\boldsymbol{\omega}}) (\boldsymbol{\omega} - \widehat{\boldsymbol{\omega}}), \tag{A.20}$$

where $\mathsf{H}(\widehat{\boldsymbol{\omega}})$ is the Hessian of $E(\boldsymbol{\omega})$, i.e.,

$$[\mathsf{H}(\widehat{\boldsymbol{\omega}})]_{ij} = \frac{\partial^2 \mathcal{E}}{\partial [\boldsymbol{\omega}]_i \partial [\boldsymbol{\omega}]_j} (\widehat{\boldsymbol{\omega}}).$$

From Eq. (A.3) we then obtain

$$p(\boldsymbol{\omega}|D^\delta) \propto \exp[-\mathcal{E}(\boldsymbol{\omega})] \propto \exp\Big[-\frac{1}{2} (\boldsymbol{\omega} - \widehat{\boldsymbol{\omega}})^T \mathsf{H}(\widehat{\boldsymbol{\omega}}) (\boldsymbol{\omega} - \widehat{\boldsymbol{\omega}}) \Big], \tag{A.21}$$

while substituting Eqs. (A.2) and (A.21) in Eq. (A.15), we find

$$\begin{aligned} p(\mathbf{y}^\delta|\mathbf{x}, D^\delta) &= \int p(\mathbf{y}^\delta|\mathbf{x}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|D^\delta)\, \mathrm{d}\boldsymbol{\omega} \\ &\propto \int \exp\Big\{ -\frac{1}{2} [\mathbf{y}^\delta - \mathbf{f}(\mathbf{x}, \boldsymbol{\omega})]^T \mathsf{C}_{\delta\mathrm{N}}^{-1} [\mathbf{y}^\delta - \mathbf{f}(\mathbf{x}, \boldsymbol{\omega})] \Big\} \\ &\quad \times \exp\Big[-\frac{1}{2} (\boldsymbol{\omega} - \widehat{\boldsymbol{\omega}})^T \mathsf{H}(\widehat{\boldsymbol{\omega}}) (\boldsymbol{\omega} - \widehat{\boldsymbol{\omega}}) \Big]\, \mathrm{d}\boldsymbol{\omega}. \end{aligned} \tag{A.22}$$

Approximating the surrogate function $\mathbf{f}(\mathbf{x}, \boldsymbol{\omega})$ by a linear Taylor expansion around $\widehat{\boldsymbol{\omega}}$, i.e.,

$$\mathbf{f}(\mathbf{x}, \boldsymbol{\omega}) \approx \mathbf{f}(\mathbf{x}, \widehat{\boldsymbol{\omega}}) + \mathsf{K}_\omega(\mathbf{x}, \widehat{\boldsymbol{\omega}}) (\boldsymbol{\omega} - \widehat{\boldsymbol{\omega}}), \tag{A.23}$$

where K_ω is the Jacobian of $\mathbf{f}$ with respect to $\boldsymbol{\omega}$, substituting Eq. (A.23) into Eq. (A.22), and computing the integral over $\boldsymbol{\omega}$ by using the basic integral

$$\int \exp(\mathbf{a}^T \mathbf{x}) \exp\Big(-\frac{1}{2} \mathbf{x}^T \mathsf{A} \mathbf{x} \Big)\, \mathrm{d}\mathbf{x} = (2\pi)^{\dim(\mathbf{x})/2} |\mathsf{A}|^{-1/2} \exp\Big(\frac{1}{2} \mathbf{a}^T \mathsf{A}^{-1} \mathbf{a} \Big), \tag{A.24}$$

we obtain

$$p(\mathbf{y}^\delta|\mathbf{x}, D^\delta) = \exp\Big\{ -\frac{1}{2} [\mathbf{y}^\delta - \mathbf{f}(\mathbf{x}, \widehat{\boldsymbol{\omega}})]^T \mathsf{C}_\delta^{-1} [\mathbf{y}^\delta - \mathbf{f}(\mathbf{x}, \widehat{\boldsymbol{\omega}})] \Big\}, \tag{A.25}$$

where

$$\mathsf{C}_\delta = \mathsf{C}_{\delta\mathrm{N}} + \mathsf{C}_{\delta\mathrm{A}}, \tag{A.26}$$

$$\mathsf{C}_{\delta\mathrm{A}} = \mathsf{K}_\omega(\mathbf{x}, \widehat{\boldsymbol{\omega}}) \mathsf{H}^{-1}(\widehat{\boldsymbol{\omega}}) \mathsf{K}_\omega(\mathbf{x}, \widehat{\boldsymbol{\omega}})^T. \tag{A.27}$$

From Eq. (A.26), we see that the covariance matrix C_δ has two components.

1. The first term, $\mathsf{C}_{\delta\mathrm{N}}$, represents the covariance matrix associated with measurement noise. In the context of neural networks, it quantifies aleatoric heteroscedastic uncertainty, which is captured by the likelihood function $p(\mathbf{y}^\delta|\mathbf{x}, \boldsymbol{\omega})$.

2. The second term, $\mathsf{C}_{\delta\mathrm{A}}$, corresponds to the covariance matrix of the approximation error. From a neural network perspective, this component accounts for epistemic (or model) uncertainty, which arises from uncertainty in the network's weights $\boldsymbol{\omega}$. This type of uncertainty reflects the lack of knowledge about the optimal model that fits the dataset and is characterized by the posterior distribution $p(\boldsymbol{\omega}|D^{\delta})$. Contributing factors include: (i) suboptimal hyperparameters, such as the number of hidden layers, units per layer, or activation functions; (ii) non-ideal training settings, including early stopping criteria, learning rate schedules, and mini-batch sizes; and (iii) the choice of optimization algorithm, such as ADAGRAD, ADADELTA, ADAM, ADAMAX, or NADAM.

Comments:

1. The calculation of the covariance matrix $\mathsf{C}_{\delta\mathrm{A}}$ from Eq. (A.27) requires access to the Hessian matrix H. In most cases, this presents a significant challenge, as the Hessian matrix is large and computationally expensive to obtain. However, this issue can be circumvented by adopting a diagonal approximation for the Hessian, given by

$$[\mathsf{H}(\widehat{\boldsymbol{\omega}})]_{ij} = \delta_{ij}\frac{\partial^2\mathcal{E}}{\partial[\boldsymbol{\omega}]_i^2}(\widehat{\boldsymbol{\omega}}), \tag{A.28}$$

where δ_{ij} is the Kronecker delta. The diagonal elements of the matrix can be efficiently computed using a technique akin to the back-propagation algorithm used for calculating first derivatives.
2. The covariance matrix C_{δ} can be approximated by the conditional average covariance matrix $\mathbb{E}(\mathsf{C}_{\delta}|D_{\mathrm{T}}^{\delta})$ over the test set $\mathcal{D}_{\mathrm{T}}^{\delta}$, given by

$$\mathsf{C}_{\delta} \approx \mathbb{E}(\mathsf{C}_{\delta}|D_{\mathrm{T}}^{\delta}) = \frac{1}{N_{\mathrm{T}}}\sum_{i=1}^{N_{\mathrm{T}}}(\mathbf{e}_i-\bar{\mathbf{e}})(\mathbf{e}_i-\bar{\mathbf{e}})^T, \tag{A.29}$$

where $\mathbf{e}_i = \mathbf{y}_i^{\delta} - \mathbf{f}(\mathbf{x}_i,\widehat{\boldsymbol{\omega}})$ is the error for the ith test point, $\bar{\mathbf{e}} = (1/N_{\mathrm{T}})\sum_{i=1}^{N_{\mathrm{T}}}\mathbf{e}_i$ is the sample mean of the error, and N_{T} is the number of test points.

Inverse problem

To solve the inverse problem $\mathbf{f}(\mathbf{x},\widehat{\boldsymbol{\omega}}) = \mathbf{y}^{\delta}$, we apply Bayesian regularization [10,11]. In this case, the posterior density $p(\mathbf{x}|\mathbf{y}^{\delta},D^{\delta})$ is given by

$$p(\mathbf{x}|\mathbf{y}^{\delta},D^{\delta}) = \frac{p(\mathbf{y}^{\delta}|\mathbf{x},D^{\delta})p(\mathbf{x})}{p(\mathbf{y}^{\delta})}. \tag{A.30}$$

Assuming that the state vector $\mathbf{x}$ is a Gaussian random vector with mean $\mathbf{x}_{\mathrm{a}}$ and covariance matrix C_{x}, i.e.,

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}_{\mathrm{a}},\mathsf{C}_{\mathrm{x}}), \tag{A.31}$$

we obtain the maximum a posteriori estimate

$$\widehat{\mathbf{x}} = \mathbf{x}_{\mathrm{MAP}} = \arg\max_{\mathbf{x}}\log p(\mathbf{x}|\mathbf{y}^{\delta},D^{\delta}) = \arg\min_{\mathbf{x}}\mathcal{P}(\mathbf{x}|\mathbf{y}^{\delta}), \tag{A.32}$$

where $\mathcal{P}(\mathbf{x}|\mathbf{y}^{\delta})$ is the posterior potential, given by

$$\mathcal{P}(\mathbf{x}|\mathbf{y}^{\delta}) = \frac{1}{2}\left\{[\mathbf{y}^{\delta}-\mathbf{f}(\mathbf{x},\widehat{\boldsymbol{\omega}})]^T\mathsf{C}_{\delta}^{-1}[\mathbf{y}^{\delta}-\mathbf{f}(\mathbf{x},\widehat{\boldsymbol{\omega}})]+(\mathbf{x}-\mathbf{x}_{\mathrm{a}})^T\mathsf{C}_{\mathrm{x}}^{-1}(\mathbf{x}-\mathbf{x}_{\mathrm{a}})\right\}, \tag{A.33}$$

and the maximum likelihood estimate

$$\widehat{\mathbf{x}} = \mathbf{x}_{\mathrm{MLE}} = \arg\max_{\mathbf{x}}\log p(\mathbf{y}^{\delta}|\mathbf{x},D^{\delta}) = \arg\min_{\mathbf{x}}\mathcal{R}(\mathbf{x}|\mathbf{y}^{\delta}), \tag{A.34}$$

where $\mathcal{R}(\mathbf{x}|\mathbf{y}^{\delta}) = (1/2)[\mathbf{y}^{\delta} - \mathbf{f}(\mathbf{x},\widehat{\boldsymbol{\omega}})]^T\mathsf{C}_{\delta}^{-1}[\mathbf{y}^{\delta} - \mathbf{f}(\mathbf{x},\widehat{\boldsymbol{\omega}})]$ is the squared residual. It is apparent that the posterior potential (A.33) coincides with the Tikhonov function (10), when (i) the whitening transformation is applied, (ii) the regularization matrix L is chosen as the Cholesky factor of $\mathsf{C}_{\mathrm{x}}^{-1}$, and (iii) the regularization parameter λ is omitted.

A.2. Neural network model with exact data

In our inversion algorithm, we use a neural network trained on the exact dataset $D = \{(\mathbf{x}_i,\mathbf{y}_i)\}_{i=1}^{N}$, where $\mathbf{y}_i = \mathbf{F}(\mathbf{x}_i)$. Such a neural network model with exact data (or noise-free output) can be treated as a special case of the noisy data model by setting $\mathsf{C}_{\delta\mathrm{N}} = \sigma_{\delta\mathrm{N}}^2\mathrm{I}_m$, and taking the limit $\sigma_{\delta\mathrm{N}}^2 \to 0$.

From the predictive distribution (cf. Eq. (A.2))

$$p(\mathbf{y}^{\delta}|\mathbf{x},\boldsymbol{\omega}) = \mathcal{N}(\mathbf{f}(\mathbf{x},\boldsymbol{\omega}),\sigma_{\delta\mathrm{N}}^2\mathrm{I}_m) = \frac{1}{(2\pi\sigma_{\delta\mathrm{N}}^2)^{m/2}}\exp\left(-\frac{1}{2\sigma_{\delta\mathrm{N}}^2}\|\mathbf{y}^{\delta}-\mathbf{f}(\mathbf{x},\boldsymbol{\omega})\|^2\right), \tag{A.35}$$

and taking into account that the Gaussian density function $\mathcal{N}(\mathbf{f}(\mathbf{x},\boldsymbol{\omega}),\sigma_{\delta\mathrm{N}}^2\mathrm{I}_m)$ converges in distribution to a Dirac delta function centered at $\mathbf{f}(\mathbf{x},\boldsymbol{\omega})$ as $\sigma_{\delta\mathrm{N}}^2 \to 0$, we obtain

$$p(\mathbf{y}|\mathbf{x},\boldsymbol{\omega}) = \lim_{\sigma_{\delta\mathrm{N}}^2\to 0} p(\mathbf{y}^{\delta}|\mathbf{x},\boldsymbol{\omega}) = \delta(\mathbf{y}-\mathbf{f}(\mathbf{x},\boldsymbol{\omega})). \tag{A.36}$$

Thus, in the limit of vanishing noise variance, the predictive distribution collapses to a Dirac delta distribution. Consequently, the likelihood $p(D|\boldsymbol{\omega})$ becomes

$$p(D|\boldsymbol{\omega}) = p(\mathbf{Y}|\mathbf{X},\boldsymbol{\omega}) = \prod_{i=1}^{N}\delta(\mathbf{y}_i-\mathbf{f}(\mathbf{x}_i,\boldsymbol{\omega})). \tag{A.37}$$

Since $\lambda \to 0$ as $\sigma_{\delta\mathrm{N}}^2, \to 0$ (the regularization term vanishes in the limit), the maximum likelihood estimate coincide with the maximum a posteriori estimate (cf. Eqs. (A.13) and (A.14))

$$\widehat{\boldsymbol{\omega}} = \boldsymbol{\omega}_{\mathrm{MLE}} = \boldsymbol{\omega}_{\mathrm{MAP}} = \arg\min_{\boldsymbol{\omega}}\left[\frac{1}{2}\sum_{i=1}^{N}\|\mathbf{y}_i-\mathbf{f}(\mathbf{x}_i,\boldsymbol{\omega})\|^2\right], \tag{A.38}$$

This equation expresses that, in the absence of noise, the model output must exactly match the true data, enforcing deterministic interpolation.

In the Laplace approximation, Eqs. (A.25)–(A.27) yield the predictive distribution

$$p(\mathbf{y}|\mathbf{x},D) = \lim_{\sigma_{\delta\mathrm{N}}^2\to 0} p(\mathbf{y}^{\delta}|\mathbf{x},D^{\delta}) = \exp\left\{-\frac{1}{2}[\mathbf{y}-\mathbf{f}(\mathbf{x},\widehat{\boldsymbol{\omega}})]^T\mathsf{C}_{\delta\mathrm{A}}^{-1}[\mathbf{y}-\mathbf{f}(\mathbf{x},\widehat{\boldsymbol{\omega}})]\right\}, \tag{A.39}$$

where $\mathsf{C}_{\delta\mathrm{A}}$ is given by Eq. (A.27).

Regarding the inverse problem, we first compute the covariance matrix of the approximation error $\mathsf{C}_{\delta\mathrm{A}}$ from the statistics of the errors $\mathbf{e}_i = \mathbf{y}_i - \mathbf{f}(\mathbf{x}_i,\widehat{\boldsymbol{\omega}})$ at a set of test points. Let $\mathbf{y}^{\delta} = \mathbf{y} + \boldsymbol{\delta}_{\mathrm{N}}$, where $\boldsymbol{\delta}_{\mathrm{N}} \sim \mathcal{N}(\mathbf{0},\mathsf{C}_{\delta\mathrm{N}})$, be the noisy data vector. We then use the result

$$p(\mathbf{y}^{\delta}|\mathbf{y}) \propto \exp\left[-\frac{1}{2}(\mathbf{y}^{\delta}-\mathbf{y})^T\mathsf{C}_{\delta\mathrm{N}}^{-1}(\mathbf{y}^{\delta}-\mathbf{y})\right] \tag{A.40}$$

along with the fundamental integral

$$\begin{aligned}&\int\exp\left[-\frac{1}{2}(\mathbf{a}+\mathbf{x})^T\mathsf{C}_1^{-1}(\mathbf{a}+\mathbf{x})\right]\exp\left(-\frac{1}{2}\mathbf{x}^T\mathsf{C}_2^{-1}\mathbf{x}\right)\mathrm{d}\mathbf{x}\\ &= (2\pi)^{\dim(\mathbf{x})/2}|\mathsf{C}_1^{-1}+\mathsf{C}_2^{-1}|^{-1/2}\exp\left[-\frac{1}{2}\mathbf{a}^T(\mathsf{C}_1+\mathsf{C}_2)^{-1}\mathbf{a}\right]\end{aligned} \tag{A.41}$$

to compute the predictive distribution for the noisy data by marginalization, i.e.,

$$p(\mathbf{y}^{\delta}|\mathbf{x},D) = \int p(\mathbf{y}^{\delta}|\mathbf{y})p(\mathbf{y}|\mathbf{x},D)\,\mathrm{d}\mathbf{y}. \tag{A.42}$$

Substituting the expressions given by Eqs. (A.39) and (A.40) into Eq. (A.42), we find

$$\begin{aligned}p(\mathbf{y}^{\delta}|\mathbf{x},D) \propto &\int\exp\left\{-\frac{1}{2}[\mathbf{y}^{\delta}-\mathbf{f}(\mathbf{x},\widehat{\boldsymbol{\omega}})+\Delta\mathbf{y}]^T\mathsf{C}_{\delta\mathrm{A}}^{-1}[\mathbf{y}^{\delta}-\mathbf{f}(\mathbf{x},\widehat{\boldsymbol{\omega}})+\Delta\mathbf{y}]\right\}\\ &\times\exp\left[-\frac{1}{2}\Delta\mathbf{y}^T\mathsf{C}_{\delta\mathrm{N}}^{-1}\Delta\mathbf{y}\right]\mathrm{d}\Delta\mathbf{y},\end{aligned} \tag{A.43}$$

while applying the integral result (A.41), we obtain

$$p(\mathbf{y}^{\delta}|\mathbf{x},D) \propto \exp\left\{-\frac{1}{2}[\mathbf{y}^{\delta}-\mathbf{f}(\mathbf{x},\widehat{\boldsymbol{\omega}})]^T\mathsf{C}_{\delta}^{-1}[\mathbf{y}^{\delta}-\mathbf{f}(\mathbf{x},\widehat{\boldsymbol{\omega}})]\right\}, \tag{A.44}$$

where $\Delta\mathbf{y} = \mathbf{y} - \mathbf{y}^{\delta}$, and as in Eq. (A.26), $\mathsf{C}_{\delta} = \mathsf{C}_{\delta\mathrm{N}} + \mathsf{C}_{\delta\mathrm{A}}$. The expression for the predictive distribution in Eq. (A.44) is identical to that in Eq. (A.25). Consequently, solving the inverse problem using Bayesian regularization (according to Eqs. (A.30) and (A.31)) results in the same estimates as those given in Eqs. (A.32) and (A.34).

A.3. Neural network model with output noise and forward model parameter error

In the presence of measurement noise $\boldsymbol{\delta}_{\mathrm{N}}$ and forward model parameter error $\boldsymbol{\delta}_{\mathrm{B}}$, the noisy data vector is defined through the successive transformation

$$\mathbf{y}^{\delta} = \mathbf{F}(\mathbf{x},\mathbf{b}) + \boldsymbol{\delta}_{\mathrm{N}} = \mathbf{F}(\mathbf{x},\overline{\mathbf{b}}) + \boldsymbol{\delta}_{\mathrm{B}} + \boldsymbol{\delta}_{\mathrm{N}}, \tag{A.45}$$

where, in view of the first-order Taylor expansion $\mathbf{F}(\mathbf{x},\mathbf{b}) \approx \mathbf{F}(\mathbf{x},\overline{\mathbf{b}}) + \mathsf{K}_{\mathrm{b}}(\mathbf{b}-\overline{\mathbf{b}})$, $\boldsymbol{\delta}_{\mathrm{B}}$ is approximated as $\boldsymbol{\delta}_{\mathrm{B}} \approx \mathsf{K}_{\mathrm{b}}(\mathbf{b}-\overline{\mathbf{b}})$, with K_{b} being the Jacobian of $\mathbf{F}$ with respect to the forward model parameter vector $\mathbf{b}$. Assuming that $\mathbf{b}$ follows a Gaussian distribution with mean $\overline{\mathbf{b}}$ and covariance matrix C_{b}, i.e., $\mathbf{b} \sim \mathcal{N}(\overline{\mathbf{b}},\mathsf{C}_{\mathrm{b}})$, we find that $\boldsymbol{\delta}_{\mathrm{B}} \sim \mathcal{N}(\mathbf{0},\mathsf{C}_{\delta\mathrm{B}})$ where $\mathsf{C}_{\delta\mathrm{B}} = \mathsf{K}_{\mathrm{b}}\mathsf{C}_{\mathrm{b}}\mathsf{K}_{\mathrm{b}}^{T}$. Furthermore, for $\boldsymbol{\delta}_{\mathrm{N}} \sim \mathcal{N}(\mathbf{0},\mathsf{C}_{\delta\mathrm{N}})$, we see that $\boldsymbol{\delta}_{\mathrm{B}} + \boldsymbol{\delta}_{\mathrm{N}} \sim \mathcal{N}(\mathbf{0},\mathsf{C}_{\delta\mathrm{B}} + \mathsf{C}_{\delta\mathrm{N}})$. Accordingly, a data representation of a neural network model that incorporates these two types of uncertainty, is given by

$$\mathbf{y}^{\delta} = \mathbf{f}(\mathbf{x},\boldsymbol{\omega}) + \boldsymbol{\delta}_{\mathrm{B}} + \boldsymbol{\delta}_{\mathrm{N}}, \quad \boldsymbol{\delta}_{\mathrm{B}} + \boldsymbol{\delta}_{\mathrm{N}} \sim \mathcal{N}(\mathbf{0},\mathsf{C}_{\delta\mathrm{B}} + \mathsf{C}_{\delta\mathrm{N}}), \tag{A.46}$$

and a training data set is defined as $\mathcal{D}^{\delta} = \{(\mathbf{x}_i,\mathbf{y}_i^{\delta})\}_{i=1}^{N}$, where $\mathbf{y}_i^{\delta} = \mathbf{F}(\mathbf{x}_i,\overline{\mathbf{b}}) + \boldsymbol{\delta}_{\mathrm{B}} + \boldsymbol{\delta}_{\mathrm{N}}$. Note that the input to the network is only $\mathbf{x}_i$, which implies that the network does not explicitly depend on $\mathbf{b}$ but instead learns an effective mapping based on the mean parameters $\overline{\mathbf{b}}$ and the statistical effects of $\mathbf{b}$ through $\boldsymbol{\delta}_{\mathrm{B}}$. This setup corresponds to learning an effective forward model that marginalizes out $\mathbf{b}$ and captures its influence through $\boldsymbol{\delta}_{\mathrm{B}}$, making the network approximate the mean behavior of the system. Comparing Eqs. (A.1) and (A.46), we deduce that the results established for a model with output noise remain valid with the replacements $\boldsymbol{\delta}_{\mathrm{N}} \longrightarrow \boldsymbol{\delta}_{\mathrm{B}} + \boldsymbol{\delta}_{\mathrm{N}}$ and $\mathsf{C}_{\delta\mathrm{N}} \longrightarrow \mathsf{C}_{\delta\mathrm{B}} + \mathsf{C}_{\delta\mathrm{N}}$.

Actually, it is possible to train the neural network to approximate $\mathbf{F}(\mathbf{x},\mathbf{b})$, capturing the variability induced by $\mathbf{b}$. In this case, the input to the neural network is $\mathbf{z} = \begin{bmatrix}\mathbf{x}\\ \mathbf{b}\end{bmatrix}$, the data representation is $\mathbf{y}^{\delta} = \mathbf{f}(\mathbf{z},\boldsymbol{\omega}) + \boldsymbol{\delta}_{\mathrm{N}}$, and a training data set is defined as $\mathcal{D}^{\delta} = \{(\mathbf{z}_i,\mathbf{y}_i^{\delta})\}_{i=1}^{N}$, where $\mathbf{y}_i^{\delta} = \mathbf{F}(\mathbf{z}_i) + \boldsymbol{\delta}_{\mathrm{N}} = \mathbf{F}(\mathbf{x}_i,\mathbf{b}_i) + \boldsymbol{\delta}_{\mathrm{N}}$ with the sample $\mathbf{b}_i$ being drawn from $\mathcal{N}(\overline{\mathbf{b}},\mathsf{C}_{\mathrm{b}})$. Drawing the sample $\mathbf{b}_i$ from a normal distribution is known as jittering the input data, which introduces small perturbations to capture variability.

Data availability

No data was used for the research described in the article.

References

[1] Israelachvili J. Intermolecular and surface forces. Elsevier; 2011, http://dx.doi.org/10.1016/c2009-0-21560-1.

[2] Berg JC. An introduction to interfaces and colloids: The bridge to nanoscience. World Scientific; 2009, http://dx.doi.org/10.1142/7579.

[3] Prieve DC, Frej NA. Total internal reflection microscopy: A quantitative tool for the measurement of colloidal forces. Langmuir 1990;6:396–403. http://dx.doi.org/10.1021/la00092a019.

[4] Prieve DC. Measurement of colloidal forces with TIRM. Adv Colloid Interface Sci 1999;82:93–125. http://dx.doi.org/10.1016/s0001-8686(99)00012-3.

[5] Doicu A, Vasilyeva A, Efremenko D, Wirth C, Wriedt T. A light scattering model for total internal reflection microscopy of geometrically anisotropic particles. J Modern Opt 2019;66:1139–51. http://dx.doi.org/10.1080/09500340.2019.1605005.

[6] Rashidi A, Domínguez-Medina S, Yan J, Efremenko D, Vasilyeva A, Doicu A, et al. Developing scattering morphology resolved total internal reflection microscopy (SMR-TIRM) for orientation detection of colloidal ellipsoids. Langmuir 2020;36:13041–50. http://dx.doi.org/10.1021/acs.langmuir.0c02482.

[7] Yan J, Efremenko DS, Vasilyeva AA, Doicu A, Wriedt T, Wirth C. Scattering morphology resolved total internal reflection microscopy (SMR-TIRM) of colloidal spheres. Comput Math Math Phys 2021;32:86–93. http://dx.doi.org/10.1007/s10598-021-09518-x.

[8] Doicu A, Efremenko DS, Wirth CL, Wriedt T. An advanced light scattering imaging model for total internal reflection microscopy considering a stratified medium. J Quant Spectrosc Radiat Transfer 2024;320:108964. http://dx.doi.org/10.1016/j.jqsrt.2024.108964.

[9] Adler A, Lionheart WRB. Uses and abuses of EIDORS: an extensible software base for EIT. Physiol Meas 2006;27(5):S25–42. http://dx.doi.org/10.1088/0967-3334/27/5/s03.

[10] Tarantola A. Inverse problem theory and methods for model parameter estimation. Society for Industrial and Applied Mathematics; 2005, http://dx.doi.org/10.1137/1.9780898717921.

[11] Rodgers CD. Inverse methods for atmospheric sounding: Theory and practice. World Scientific; 2000, http://dx.doi.org/10.1142/3171.

[12] Sidky EY, Pan X. Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization. Phys Med Biol 2008;53(17):4777–807. http://dx.doi.org/10.1088/0031-9155/53/17/021.

[13] Strokotov DI, Yurkin MA, Gilev KV, van Bockstaele DR, Hoekstra AG, Rubtsov NB, Maltsev VP. Is there a difference between T- and B-lymphocyte morphology? J Biomed Opt 2009;14(6):064036. http://dx.doi.org/10.1117/1.3275471.

[14] Romanov AV, Konokhova AI, Yastrebova ES, Gilev KV, Strokotov DI, Chernyshev AV, Maltsev VP, Yurkin MA. Spectral solution of the inverse Mie problem. J Quant Spectrosc Radiat Transfer 2017;200:280–94. http://dx.doi.org/10.1016/j.jqsrt.2017.04.034.

[15] Draper NR, Smith H. Applied regression analysis. 3rd ed.. Wiley; 1998.

[16] Buhmann MD. Radial basis functions: Theory and implementations. Cambridge University Press; 2003.

[17] Gutmann H-M. A radial basis function method for global optimization. J Global Optim 2001;19(3):201–27. http://dx.doi.org/10.1023/a:1011255519438.

[18] Rasmussen CE, Williams CKI. Gaussian processes for machine learning. MIT Press; 2006.

[19] Bishop CM. Pattern recognition and machine learning. Springer; 2006.

[20] Haykin S. Neural networks and learning machines. Pearson; 2009.

[21] Schmidhuber J. Deep learning in neural networks: An overview. Springer; 2015.

[22] Nielsen MA. Neural networks and deep learning. Determination Press; 2015.

[23] Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press; 2016.

[24] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nat 1986;323(6088):533–6. http://dx.doi.org/10.1038/323533a0.

[25] Hadamard J. Lectures on the existence and uniqueness of solutions of partial differential equations. Bull Am Math Soc 1902;8:328–79.

[26] Tikhonov AN, Goncharsky AV, Stepanov VV, Yagola AG. Numerical methods for the solution of ill-posed problems. Springer Netherlands; 1995, http://dx.doi.org/10.1007/978-94-015-8480-7.

[27] Doicu A, Trautmann T, Schreier F. Numerical regularization for atmospheric inverse problems. Springer Berlin Heidelberg; 2010, http://dx.doi.org/10.1007/978-3-642-05439-6.

[28] Adjiman C, Dallwig S, Floudas C, Neumaier A. A global optimization method, αBB, for general twice-differentiable constrained NLPs — I. Theoretical advances. Comput Chem Eng 1998;22(9):1137–58. http://dx.doi.org/10.1016/s0098-1354(98)00027-1.

[29] Piyavskii S. An algorithm for finding the absolute extremum of a function. USSR Comput Math Math Phys 1972;12(4):57–67. http://dx.doi.org/10.1016/0041-5553(72)90115-2.

[30] Shubert BO. A sequential method seeking the global maximum of a function. SIAM J Numer Anal 1972;9(3):379–88. http://dx.doi.org/10.1137/0709036.

[31] Hansen E, William Walster G. Global optimization using interval analysis. CRC Press; 2003, http://dx.doi.org/10.1201/9780203026922.

[32] Watson LT. Probability-one homotopies in computational science. J Comput Appl Math 2002;140(1–2):785–807. http://dx.doi.org/10.1016/s0377-0427(01)00473-3.

[33] Floudas C, Pardalos P, editors. Encyclopedia of optimization. Springer US; 2009, http://dx.doi.org/10.1007/978-0-387-74759-0.

[34] Törn AA. A search clustering approach to global optimization. In: Dixon LCW, Szegö GP, editors. Towards global optimizations. North-Holland Publishing Company, Amsterdam, Holland: North-Holland Publishing Company; 1978, p. 49–62.

[35] Boender CGE, Rinnooy Kan HG, Timmer GT, Stougie L. A stochastic method for global optimization. Math Program 1982;22:125–40. http://dx.doi.org/10.1007/BF01581033.

[36] Rinnooy Kan AHG, Timmer GT. Stochastic global optimization methods, part I: Clustering methods. Math Program 1987;39:27–56. http://dx.doi.org/10.1007/BF02592070.

[37] Rinnooy Kan AHG, Timmer GT. Stochastic global optimization methods, part II: Multilevel methods. Math Program 1987;39:57–78. http://dx.doi.org/10.1007/BF02592071.

[38] Tsoulos IG, Lagaris IE. MinFinder: Locating all the local minima of a function. Comput Phys Commun 2006;174:166–79. http://dx.doi.org/10.1016/j.cpc.2005.10.001.
[39] Ali MM, Storey C. Topographical multilevel single linkage. J Global Optim 1994;5(4):349–58. http://dx.doi.org/10.1007/bf01096684.
[40] Lagrange S, Delanoue N, Jaulin L. Guaranteed numerical injectivity test via interval analysis. In: Trends in constraint programming. London, UK: ISTE; 2017, p. 233–44.
[41] Lagrange S, Delanoue N, Jaulin L. On sufficient conditions of the injectivity: Development of a numerical test algorithm via interval analysis. Reliab Comput 2007;13(5):409–21. http://dx.doi.org/10.1007/s11155-007-9042-9.
[42] Gainer-Dewar A, Vera-Licona P. The minimal hitting set generation problem: Algorithms and computation. SIAM J Discrete Math 2017;31:63–100. http://dx.doi.org/10.1137/15M1055024.
[43] Hébert C, Bretto A, Crémilleux B. A data mining formalization to improve hypergraph minimal transversal computation. Fund Inform 2007;80:415–34.
[44] Dong G, Li J. Mining border descriptions of emerging patterns from dataset pairs. Knowl Inf Syst 2005;8:178–202. http://dx.doi.org/10.1007/s10115-004-0178-1.
[45] Slavík P. A tight analysis of the greedy algorithm for set cover. In: Proceedings of the twenty-eighth annual ACM symposium on theory of computing. ACM; 1996, p. 435–41. http://dx.doi.org/10.1145/237814.237991.
[46] Nemhauser GL, Wolsey LA. Integer and combinatorial optimization. Wiley-Interscience; 1988.
[47] Morozov VA. Methods for solving incorrectly posed problems. Springer New York; 1984, http://dx.doi.org/10.1007/978-1-4612-5280-1.
[48] Marsaglia G, Tsang WW. The Ziggurat method for generating random variables. J Stat Softw 2000;5:1–7. http://dx.doi.org/10.18637/jss.v005.i08.
[49] Matsumoto M, Nishimura T. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. ACM Trans Model Comput Simul 1998;1:3–30. http://dx.doi.org/10.1145/272991.272995.
[50] Dong N, Wu CH, Ip WH, Chen ZQ, Chan CY, Yung KL. An opposition-based chaotic GA/PSO hybrid algorithm and its application in circle detection. Comput Math Appl 2012;64:1886–902. http://dx.doi.org/10.1016/j.camwa.2012.03.040.
[51] Gao Y, Wang YJ. A memetic differential evolutionary algorithm for high dimensional functions optimization. In: Third international conference on natural computation. ICNC 2007, IEEE; 2007, p. 188–92.
[52] Gao WF, Liu SY. A modified artificial bee colony algorithm. Comput Oper Res 2012;39:687–97. http://dx.doi.org/10.1016/j.cor.2011.06.007.
[53] McKay MD, Beckman RJ, Conover WJ. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 1979;21:239–45. http://dx.doi.org/10.2307/1268522.
[54] Rahnamayan S, Tizhoosh HR, Salama MM. Opposition-based differential evolution for optimization of noisy problems. In: 2006 IEEE international conference on evolutionary computation. IEEE; 2006, p. 1865–72.
[55] Rahnamayan S, Tizhoosh HR, Salama MM. Opposition-based differential evolution. IEEE Trans Evol Comput 2008;12:64–79. http://dx.doi.org/10.1109/TEVC.2007.894200.
[56] Du Q, Gunzburger M, Ju L. Advances in studies and applications of centroidal voronoi tessellations. Numer Math Theory Methods Appl 2010;3:119–42. http://dx.doi.org/10.4208/nmtma.2010.32s.1.
[57] Dennis Jr JE, Gay DM, Welsch RE. An adaptive nonlinear least-squares algorithm. ACM Trans Math Software 1981;7:348–68. http://dx.doi.org/10.1145/355958.355965.
[58] Dennis Jr JE, Gay DM, Welsch RE. Algorithm 573. NL2SOL — An adaptive nonlinear least-squares algorithm. ACM Trans Math Software 1981;7:369–83. http://dx.doi.org/10.1145/355958.355966.
[59] Byrd RH, Lu P, Nocedal J, Zhu C. A limited memory algorithm for bound constrained optimization. SIAM J Sci Comput 1995;16:1190–208. http://dx.doi.org/10.1137/0916069.
[60] Liu P, Nocedal J, Waltz R. CG+. 2024, http://users.iems.northwestern.edu/~nocedal/CG+.html. [Accessed 06 August 2024].
[61] Gilbert JC, Nocedal J. Global convergence properties of conjugate gradient methods. SIAM J Optim 1992;2:21–42. http://dx.doi.org/10.1137/0802003.
[62] Moeller MF. A scaled conjugate gradient algorithm for fast supervised learning. Neural Netw 1993;6:525–33. http://dx.doi.org/10.1016/S0893-6080(05)80056-5.
[63] Hanson RJ, Krogh FT. A quadratic-tensor model algorithm for nonlinear least-squares problems with linear constraints. ACM Trans Math Software 1992;18:115–33. http://dx.doi.org/10.1145/146847.146857.
[64] Doicu A, Doicu A, Efremenko DS, Loyola D, Trautmann T. An overview of neural network methods for predicting uncertainty in atmospheric remote sensing. Remote Sens 2021;13:5061. http://dx.doi.org/10.3390/rs13245061.