

Model-Based Optical Metrology in R M.o.R.

Mark-Alexander Henn¹ and Nien Fan Zhang²

¹Engineering Physics Division, National Institute of Standards and Technology,
Gaithersburg, MD 20899, United States of America

²Statistical Engineering Division, National Institute of Standards and Technology,
Gaithersburg, MD 20899, United States of America

April 5, 2018

Abstract

Reliable optical critical dimension (OCD) metrology in the regime where the inspection wavelength λ is much larger than the critical dimensions (CDs) of the measurand is only possible using a model-based approach. Due to the complexity of the models involved, that often require solving Maxwell's equations, many applications use a library based look-up approach. Here, the best experiment-to-theory fit is found by comparing the measurement data to a library consisting of pre-calculated simulations. One problem with this approach is that it makes the accuracy of the solution dependent on the refinement of the grid. Interpolating between library values requires a uniform grid in most cases, and can also be very time-consuming.

We present an approach based on radial basis functions that is fast, accurate and most importantly works on arbitrary grids. The method is implemented in a application based on the programming language R, that additionally allows for Bayesian data analysis, and provides multiple diagnostics.¹

¹Certain commercial materials are identified in this talk in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the materials are necessarily the best available for the purpose.

1 Introduction

Generally speaking, regression analysis is a statistical process for estimating the relationships among variables, separated into independent and dependent variables. Note, that we restrict ourselves to scalar dependent variables, however we allow the independent variable to be vector-valued. In the following the independent variables will be denoted by \mathbf{x} , and the dependent variables will be denoted by y . Based on a limited number of pairs (\mathbf{x}_i, y_i) , $i = 1, \dots, k$, regression is used to determine the functional relationship between independent and dependent variables, and to make predictions about observations of y for arbitrary values of \mathbf{x} .

In many cases the functions that describe this relationship, also called **model functions**, are limited to a specific class that can be described by a few parameters, given as the components of a vector \mathbf{p} . In a simple case where both x and y are scalar one might only consider linear functions, that is functions of the form

$$f(x, \mathbf{p}) = p_1x + p_2 \quad (1)$$

in the regression. The regression problem then amounts to finding the model function's parameters p_1 , and p_2 such that the model's predicted values are close to the actual observations.

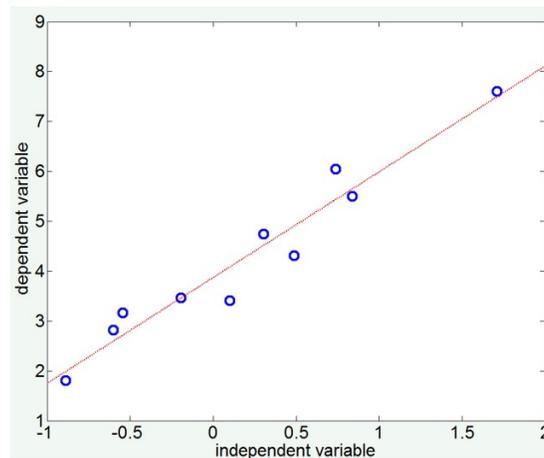


Figure 1: Example of regression using a linear function (cf. Eq. (1)), observations are marked blue, the best fit line is marked red.

One measure for closeness is the norm of the difference vector, also called the **sum of squared errors** of prediction (**SSE**)

$$SSE = \sum_{i=1}^k |f(\mathbf{x}_i, \mathbf{p}) - y_i|^2. \quad (2)$$

Note, that the minimum of the SSE, called **least-squares value**, will usually not be equal to zero due to inaccuracies in the model function, or due to the presence of measurement errors, i.e. the values y_i are noisy realizations of the model function at \mathbf{x}_i . If one has additional knowledge about those measurement errors, e.g. if one knows the variances σ_i^2 of each of the measurements y_i one can use this knowledge to weigh the different measurements accordingly, hence limiting the influence of observations that are expected to have a large error. This approach leads to the function

$$\chi^2(\mathbf{p}) = \sum_{i=1}^k \frac{1}{\sigma_i^2} \|f(\mathbf{x}_i, \mathbf{p}) - y_i\|^2, \quad (3)$$

which is also called the **weighted χ^2 (chi-square) function**. We can also write it in a more general way as:

$$\chi^2(\mathbf{p}) = (\mathbf{f}(\mathbf{x}, \mathbf{p}) - \mathbf{y})^\top \mathbf{V}^{-1} (\mathbf{f}(\mathbf{x}, \mathbf{p}) - \mathbf{y}), \quad (4)$$

with

$$\mathbf{f}(\mathbf{x}, \mathbf{p}) = (f(\mathbf{x}_1, \mathbf{p}), f(\mathbf{x}_2, \mathbf{p}), \dots, f(\mathbf{x}_k, \mathbf{p}))^\top, \quad \mathbf{y} = (y_1, y_2, \dots, y_k)^\top \quad (5)$$

and

$$\mathbf{V}^{-1} \in \mathbb{R}^{k \times k}. \quad (6)$$

The matrix \mathbf{V}^{-1} is either a pure diagonal matrix, such that

$$\mathbf{V}^{-1} = \left(\delta_{ij} \frac{1}{\sigma_i^2} \right) \in \mathbb{R}^{k \times k}$$

or it might contain non-zero off-diagonal elements if one also has information about the correlations between the different measurements.

In the following we will use regression to solve inverse problems[13, 5]. In an inverse problem one is given a set of observations and asked to determine the causal factors that produced them. In our terminology the observations are the y_i and the causal factors are both the \mathbf{x}_i and the parameter vector \mathbf{p} . However, the \mathbf{x}_i represent the known measurement conditions used to obtain the measurements y_i , while the vector \mathbf{p} represents the adjustable parameters of the measurand[16]. For a simple scattering problem, the y_i would be the measured intensities at different locations in space given in spherical coordinates (r_i, θ_i, ϕ_i) and given wavelength λ such that $\mathbf{x}_i = (r_i, \theta_i, \phi_i, \lambda)$. The vector \mathbf{p} contains information about the scatterer that is unknown such as its radius if it is assumed to be spherical or the material it is made of.

Once we establish \mathbf{x} and \mathbf{p} we minimize the function in Eq. (4) and call the values of \mathbf{p} that minimize the χ^2 function the **parametric estimates**. There might be several different values for \mathbf{p} that yield equally good, i.e. small, values for the χ^2 function. In this case we call the inverse problem **ill-posed**. One way to make the inverse problem less ill-posed is to incorporate knowledge one

has about \mathbf{p} from other experiments and add a penalty term to the χ^2 function. For example one could penalize values of \mathbf{p} that differ too much from a reasonable estimate $\hat{\mathbf{p}}$ by adding the so-called **regularization term** $\lambda \|\mathbf{p} - \hat{\mathbf{p}}\|$, with $\lambda \in \mathbb{R}_+$ to the χ^2 function. This is called **regularization**, or **Tikhonov regularization**[14], the parameter λ is called the **regularization parameter**. If we have information about the distribution of \mathbf{p} in terms of its mean and covariance we can change the penalty term accordingly, this is called the **Bayesian approach to regression**, see e.g. [16, 17, 6]. M.o.R. is an application that has been designed to solve inverse problems using both regular regression and the Bayesian approach. Details about the underlying theory are provided in Section 2, a guide on how to use the application is given in Section 3.

2 Theoretical Background

In this section we will collect some basic facts about the Bayesian approach to regression and about interpolation that will help to understand how M.o.R. works.

2.1 The Inverse Problem

Generally speaking an inverse problem is the task of calculating from a set of observations the causal factors that produced them. Several techniques can be applied to solve inverse problems [13, 7], the approach used in M.o.R is based on setting up a regression problem, in the following sense.

1 The Inverse Problem of Model-Based Metrology:

Given a vector of measurement data $\mathbf{y} \in \mathbb{R}^n$, a so-called model function

$$\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^n, \mathbf{p} \mapsto \mathbf{f}(\mathbf{p}), \quad (7)$$

that maps the parameters \mathbf{p} that we want to determine to an approximation of the measurement data, furthermore a symmetric, positive definite matrix $\mathbf{V} \in \mathbb{R}^{n \times n}$, and (if available) prior information about \mathbf{p} in terms of a mean $\boldsymbol{\mu}_{\text{pri}}$ and a covariance matrix $\boldsymbol{\Sigma}_{\text{pri}}$, find the best fit, i.e. determine the parameter vector $\hat{\mathbf{p}}$ that minimizes the **objective function** or **loss function**

$$\chi^2(\mathbf{p}) = \underbrace{(\mathbf{f}(\mathbf{p}) - \mathbf{y})^\top \mathbf{V}^{-1} (\mathbf{f}(\mathbf{p}) - \mathbf{y})}_{\text{fit to measurement data}} + \underbrace{(\mathbf{p} - \boldsymbol{\mu}_{\text{pri}})^\top \boldsymbol{\Sigma}_{\text{pri}}^{-1} (\mathbf{p} - \boldsymbol{\mu}_{\text{pri}})}_{\text{penalty term (prior knowledge)}}, \quad (8)$$

written compactly as

$$\hat{\mathbf{p}} = \operatorname{argmin} [\chi^2(\mathbf{p})]. \quad (9)$$

The above problem can be solved by applying a suitable optimization algorithm. M.o.R. uses a combination of global and local optimization algorithms[4, 15]. The local optimization algorithm makes use of the gradient of the objective function, which is given by

$$\nabla \chi^2 = 2 (\mathbf{J}_f^\top \mathbf{V}^{-1} (\mathbf{f}(\mathbf{p}) - \mathbf{y}) + \boldsymbol{\Sigma}_{\text{pri}}^{-1} (\mathbf{p} - \boldsymbol{\mu}_{\text{pri}})), \quad (10)$$

with \mathbf{J}_f being the Jacobian of the model function

$$\mathbf{J}_f = \left(\frac{\partial f_i}{\partial p_j} \right) \in \mathbb{R}^{n \times m}. \quad (11)$$

Once we have determined the best fit, we can also estimate the uncertainties in $\hat{\mathbf{p}}$ by calculating the covariance matrix using

$$\boldsymbol{\Sigma} = (\mathbf{J}_f^\top \mathbf{V}^{-1} \mathbf{J}_f)^{-1}, \quad (12)$$

or

$$\boldsymbol{\Sigma} = (\mathbf{J}_f^\top \mathbf{V}^{-1} \mathbf{J}_f + \boldsymbol{\Sigma}_{\text{pri}}^{-1})^{-1}. \quad (13)$$

if prior information is available.

2.2 Interpolation

Even though Problem 1 is easy to formulate one has to deal with a lot of challenges in real life. The most common lies in the fact that evaluating the model function might be a very time-consuming task in many cases. We therefore try to avoid evaluating it by interpolating on a grid of already calculated values.

2 The Interpolation Problem:

Assume that we have evaluated the model function \mathbf{f} at k different points^a $\mathbf{p}_i = (p_{i1}, p_{i2}, \dots, p_{im}), i = 1, \dots, k$ with

$$\mathbf{\Pi} = \begin{pmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_k \end{pmatrix} = \begin{pmatrix} p_{11} & \cdots & p_{1m} \\ \vdots & \ddots & \vdots \\ p_{k1} & \cdots & p_{km} \end{pmatrix} \in \mathbb{R}^{k \times m}, \quad (14)$$

and the corresponding function values $\mathbf{f}(\mathbf{p}_i)$ with

$$\mathbf{f}(\mathbf{\Pi}) = \mathbf{\Phi} = \begin{pmatrix} \mathbf{f}(\mathbf{p}_1)^\top \\ \vdots \\ \mathbf{f}(\mathbf{p}_k)^\top \end{pmatrix} = \begin{pmatrix} f_1(\mathbf{p}_1) & \cdots & f_n(\mathbf{p}_1) \\ \vdots & \ddots & \vdots \\ f_1(\mathbf{p}_k) & \cdots & f_n(\mathbf{p}_k) \end{pmatrix} \in \mathbb{R}^{k \times n}. \quad (15)$$

The interpolation problem then amounts to find an approximation $\tilde{\mathbf{f}}$ to the function \mathbf{f} such that

$$\tilde{\mathbf{f}}(\mathbf{p}, \mathbf{\Pi}, \mathbf{\Phi}) \approx \mathbf{f}(\mathbf{p}). \quad (16)$$

^aNote, that we use the term points in the topological sense.

Many interpolation algorithms assume that the points \mathbf{p}_i form a regular grid, that is that they are equally spaced. However, sometimes regular grids are not available, and therefore interpolation techniques such as splines cannot be applied, that is why we use the more flexible radial basis function (RBF) approach for the interpolation[9], which is based on the following reasoning.

Assume we have a set of functions $\rho_i : \mathbb{R}^m \rightarrow \mathbb{R}, i = 1, \dots, N$, such that for all $j \in \{1, \dots, n\}$, we can approximate the j -th component of \mathbf{f} , i.e. the scalar function $f_j(\mathbf{p})$ by

$$f_j(\mathbf{p}) \approx \sum_{i=1}^N a_{ji} \rho_i(\mathbf{p}). \quad (17)$$

In the RBF approach we use specific functions, that depend only on the distance of the parameter vector \mathbf{p} from the different grid points \mathbf{p}_i , and an additional hyperparameter r as

$$\rho_i(\mathbf{p}, r) = \rho(\|\mathbf{p} - \mathbf{p}_i\|, r), \quad i = 1, \dots, k. \quad (18)$$

We will later see how to adjust the hyperparameter r .

M.o.R. allows to use three different types of radial basis functions:

- (i) **Gaussian:** $\rho(\mathbf{p}, r) = \exp(-(\|\mathbf{p}\|^2 + r^2))$
- (ii) **Multiquadratic:** $\rho(\mathbf{p}, r) = \sqrt{\|\mathbf{p}\|^2 + r^2}$
- (iii) **Inverse Multiquadratic:** $\rho(\mathbf{p}, r) = \frac{1}{\sqrt{\|\mathbf{p}\|^2 + r^2}}$

Since the interpolation function should be exact if evaluated at the grid points, i.e. for all $i = 1, \dots, k$

$$f_j(\mathbf{p}_i) \stackrel{!}{=} \sum_{l=1}^k a_{jl} \rho_l(\mathbf{p}_i), \quad (19)$$

we can determine the above series' coefficients for each of the n individual components of \mathbf{f} by solving the following matrix equation:

$$\mathbf{f}_j = \mathbf{P}_0 \cdot \mathbf{a}_j \Leftrightarrow \mathbf{a}_j = \mathbf{P}_0^{-1} \cdot \mathbf{f}_j, \quad j = 1, \dots, n \quad (20)$$

with $\mathbf{f}_j = (f_j(\mathbf{p}_1), \dots, f_j(\mathbf{p}_k))^\top \in \mathbb{R}^n$, $\mathbf{a}_j = (a_{j1}, \dots, a_{jk})^\top \in \mathbb{R}^k$, and

$$\mathbf{P}_0 = \begin{pmatrix} \rho_1(\mathbf{p}_1) & \cdots & \rho_k(\mathbf{p}_1) \\ \vdots & \ddots & \vdots \\ \rho_1(\mathbf{p}_k) & \cdots & \rho_k(\mathbf{p}_k) \end{pmatrix} \in \mathbb{R}^{k \times k}. \quad (21)$$

Equation (20) needs to be solved for each \mathbf{f}_j , such that we end up with a matrix consisting of the n coefficient vectors

$$\mathbf{A} = (\mathbf{a}_1, \dots, \mathbf{a}_n)^\top \in \mathbb{R}^{n \times k}. \quad (22)$$

Using \mathbf{A} we can calculate the approximation to the model function for an arbitrary parameter vector \mathbf{p} by

$$\tilde{\mathbf{f}}(\mathbf{p}) = \mathbf{A} \cdot \mathbf{P}(\mathbf{p}), \quad \text{with } \mathbf{P}(\mathbf{p}) = (\rho_1(\mathbf{p}), \dots, \rho_k(\mathbf{p}))^\top \in \mathbb{R}^k. \quad (23)$$

We can also easily calculate the Jacobian of $\tilde{\mathbf{f}}$

$$\mathbf{J}_{\tilde{\mathbf{f}}} = \begin{pmatrix} \frac{\partial \tilde{f}_i}{\partial p_j} \end{pmatrix} \in \mathbb{R}^{n \times m} \quad (24)$$

as

$$\begin{aligned} \mathbf{J}_{\tilde{\mathbf{f}}} &= \begin{pmatrix} \frac{\partial}{\partial p_1} \left(\sum_{i=1}^k a_{1i} \rho_i(\mathbf{p}) \right) & \cdots & \frac{\partial}{\partial p_m} \left(\sum_{i=1}^k a_{1i} \rho_i(\mathbf{p}) \right) \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial p_1} \left(\sum_{i=1}^k a_{ni} \rho_i(\mathbf{p}) \right) & \cdots & \frac{\partial}{\partial p_m} \left(\sum_{i=1}^k a_{ni} \rho_i(\mathbf{p}) \right) \end{pmatrix} \\ &= \begin{pmatrix} \sum_{i=1}^k a_{1i} \left(\frac{\partial \rho_i}{\partial p_1}(\mathbf{p}) \right) & \cdots & \sum_{i=1}^k a_{1i} \left(\frac{\partial \rho_i}{\partial p_m}(\mathbf{p}) \right) \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^k a_{ni} \left(\frac{\partial \rho_i}{\partial p_1}(\mathbf{p}) \right) & \cdots & \sum_{i=1}^k a_{ni} \left(\frac{\partial \rho_i}{\partial p_m}(\mathbf{p}) \right) \end{pmatrix}. \end{aligned} \quad (25)$$

Which is equivalent to

$$\mathbf{J}_{\tilde{\mathbf{f}}} = \mathbf{A} \cdot \mathbf{J}_\rho, \quad (26)$$

with $\mathbf{A} \in \mathbb{R}^{n \times k}$ the coefficient matrix defined in Eq. (22), and

$$\mathbf{J}_\rho = \left(\frac{\partial \rho_i}{\partial p_j} \right) \in \mathbb{R}^{k \times m}. \quad (27)$$

The partial derivatives $\frac{\partial \rho_i}{\partial p_j}$ are given by

$$\frac{\partial \rho_i}{\partial p_j} = \left(\frac{\partial}{\partial p_j} \|\mathbf{p} - \mathbf{p}_i\| \right) \rho'(\|\mathbf{p} - \mathbf{p}_i\|) = \frac{(p_j - p_{ij})}{\|\mathbf{p} - \mathbf{p}_i\|} \rho'(\|\mathbf{p} - \mathbf{p}_i\|) \quad (28)$$

and therefore

$$\mathbf{J}_\rho = \underbrace{\begin{pmatrix} \frac{\rho'(\|\mathbf{p}-\mathbf{p}_1\|)}{\|\mathbf{p}-\mathbf{p}_1\|} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\rho'(\|\mathbf{p}-\mathbf{p}_k\|)}{\|\mathbf{p}-\mathbf{p}_k\|} \end{pmatrix}}_{\mathbf{P}'(\mathbf{p})} \underbrace{\begin{pmatrix} (p_1 - p_{11}) & \cdots & (p_m - p_{1m}) \\ \vdots & \ddots & \vdots \\ (p_1 - p_{k1}) & \cdots & (p_m - p_{km}) \end{pmatrix}}_{\mathbf{\Delta} \in \mathbb{R}^{k \times m}}, \quad (29)$$

since the derivative of the radial basis functions is usually given analytically, calculating the derivative takes about the same time as evaluating the radial basis functions.

Putting all of this together we have

$$\mathbf{J}_{\tilde{\mathbf{f}}} = \mathbf{A} \cdot \underbrace{\mathbf{P}'(\mathbf{p}) \cdot \mathbf{\Delta}}_{\mathbf{J}_\rho}, \quad (30)$$

and

$$\begin{aligned} \nabla \chi^2 &= 2 \left(\mathbf{J}_{\tilde{\mathbf{f}}}^\top \mathbf{V}^{-1} \left(\tilde{\mathbf{f}}(\mathbf{p}) - \mathbf{y} \right) + \mathbf{\Sigma}_{\text{pri}}^{-1} (\mathbf{p} - \boldsymbol{\mu}_{\text{pri}}) \right) \\ &= 2 \left([\mathbf{A} \cdot \mathbf{P}'(\mathbf{p}) \cdot \mathbf{\Delta}]^\top \mathbf{V}^{-1} \left(\tilde{\mathbf{f}}(\mathbf{p}) - \mathbf{y} \right) + \mathbf{\Sigma}_{\text{pri}}^{-1} (\mathbf{p} - \boldsymbol{\mu}_{\text{pri}}) \right) \end{aligned} \quad (31)$$

Several measures can be taken to speed up the computations

- Use the symmetry of the radial basis functions to reduce computations to populate \mathbf{P}_0
- Since each of the n components of \mathbf{f} is assumed to be independent, use parallelization to solve the n equations Eq. (20)

We can finally formulate the problem we want to solve.

3 The Inverse Problem Using Interpolation:

Given:

- $\mathbf{y} \in \mathbb{R}^n$, a vector of measurement data
- $\mathbf{\Pi} \in \mathbb{R}^{k \times m}$, a matrix consisting of k grid points $\mathbf{p}_i \in \mathbb{R}^m$
- $\mathbf{\Phi} \in \mathbb{R}^{k \times n}$, a matrix with corresponding function values $\mathbf{f}(\mathbf{p}_i) \in \mathbb{R}^n$
- $\mathbf{V} \in \mathbb{R}^{n \times n}$, a symmetric, positive definite matrix
- $\boldsymbol{\mu}_{\text{pri}} \in \mathbb{R}^m$, a vector containing an estimate for \mathbf{p}
- $\boldsymbol{\Sigma}_{\text{pri}} \in \mathbb{R}^{m \times m}$, a matrix containing the uncertainties for \mathbf{p}

we solve for:

$$\hat{\mathbf{p}} = \underset{\mathbf{p}}{\text{argmin}} \left[\left(\tilde{\mathbf{f}}(\mathbf{p}, \mathbf{\Pi}, \mathbf{\Phi}) - \mathbf{y} \right)^\top \mathbf{V}^{-1} \left(\tilde{\mathbf{f}}(\mathbf{p}, \mathbf{\Pi}, \mathbf{\Phi}) - \mathbf{y} \right) + \left(\mathbf{p} - \boldsymbol{\mu}_{\text{pri}} \right)^\top \boldsymbol{\Sigma}_{\text{pri}}^{-1} \left(\mathbf{p} - \boldsymbol{\mu}_{\text{pri}} \right) \right]. \quad (32)$$

2.3 Optimizing Hyperparameters

The basis functions $\phi_i(\mathbf{p}, r)$ in Eq. (18) depend not only on \mathbf{p} , but additionally on the so-called **hyperparameter**, or **nuisance parameter** r , that can have a strong impact on the quality of the interpolation. Determining for which $r \in \mathbb{R}$ the interpolation gives the best results is in M.o.R. done using leave-one-out cross-validation (LOOCV)[11] and [10], based on the following reasoning.

Let the matrix of grid points be $\mathbf{\Pi} \in \mathbb{R}^{k \times m}$, and the corresponding matrix of function values $\mathbf{\Phi} \in \mathbb{R}^{k \times n}$. Denote by $\mathbf{\Pi}_{(i)} \in \mathbb{R}^{(k-1) \times m}$ and $\mathbf{\Phi}_{(i)} \in \mathbb{R}^{(k-1) \times n}$ the respective matrices with the i -th row removed, and by \mathbf{p}_i and \mathbf{f}_i the entries of the i -th row. We can then use the matrices $\mathbf{\Pi}_{(i)}$ and $\mathbf{\Phi}_{(i)}$ to approximate the function values for \mathbf{p}_i and calculate the difference to \mathbf{y}_i , this can be done for all $i = 1, \dots, k$, and we can define a function that measures the accuracy of these approximations:

$$l(r) = \sum_{i=1}^k \left\| \tilde{\mathbf{f}}(r, \mathbf{p}_i, \mathbf{\Pi}_{(i)}, \mathbf{\Phi}_{(i)}) - \mathbf{f}_i \right\|^2. \quad (33)$$

The optimal hyperparameter \hat{r} is then found by minimizing the function in Eq. (33). We use a particle-swarm optimization algorithm, see Ref. ([2]), that has already been implemented in an R package[1].

2.4 Minimizing the χ^2 Function

M.o.R. uses a hybrid optimization technique, that combines elements from stochastic global optimization algorithms, and gradient-based methods [8] to determine the solution to Problem 3. We use a combination of R packages to minimize the χ^2 function, [1, 12]. We first apply the particle swarm optimization

with a limited number of iterations to find a good estimate for the global minimum and then use the preliminary best fit value as a starting point for the gradient-based local optimization algorithm. Based on that value we then calculate the covariance matrix using Eqs. (12) and (13).

2.5 Monte Carlo

In M.o.R. the Monte Carlo (MC) method is used to determine how systematic errors influence the parametric estimates and their uncertainties. Systematic errors might be due to inaccuracies and/or oversimplifications of the model used to process the measurement data. However, one might be interested to estimate the influence of those systematic errors, without including their cause in the model, mostly due to limited computational and time resources.

Instead one can draw several, say n_s , realizations of those systematic errors by slightly perturbing the model and use these data sets as an input to the optimization algorithm that uses the idealized model. For each realization i we get an parameter estimate $\hat{\mathbf{p}}_i$, based on the n_s estimates we can then calculate the mean parameter vector as

$$\bar{\mathbf{p}} = \frac{1}{n_s} \sum_{i=1}^{n_s} \hat{\mathbf{p}}_i \quad (34)$$

and the sample covariance $\mathbf{Q} = (q_{jk}) \in \mathbb{R}^{m \times m}$ with

$$q_{jk} = \frac{1}{n_s - 1} \sum_{i=1}^{n_s} (p_{ij} - \bar{p}_j)(p_{ik} - \bar{p}_k). \quad (35)$$

2.6 Bootstrapping

Sometimes it is hard to estimate the measurement errors, i.e. the matrix \mathbf{V} in Problem 1 is unknown, and estimating the parametric uncertainties based on Eqs. (12) and (13) is therefore not possible. In that case we can still estimate the parametric uncertainties in a bit more computational expensive way by applying bootstrapping[3], based on the following idea.

We first run the usual optimization algorithm to determine $\hat{\mathbf{p}}$. However, we set $\mathbf{V} = \mathbf{I}_{n_2}$, the $n \times n$ dimensional unit matrix. We then calculate the residual vector $\epsilon = \mathbf{f}(\hat{\mathbf{p}}) - \mathbf{y}$, create a permutation ϵ^* of the residual vector, generate a noisy realization of the input data by setting

$$\mathbf{y}^* = \mathbf{y} + \epsilon^*, \quad (36)$$

and solve the optimization problem using \mathbf{y}^* as an input to obtain $\hat{\mathbf{p}}$. By repeating this n_s times, we can again calculate the sample mean and covariance using Eqs. (34) and (35).

3 Using M.o.R.

We now want to describe the usage of M.o.R., first we show which data format is supported, and then show how to fit the data, and use the other features of M.o.R.

3.1 Data Format

M.o.R. was designed to accept comma-separated values (CSV) files for all involved matrices and vectors. Figure 2 presents an example of how the data needs to be formatted in order to be able to be processed in the program.

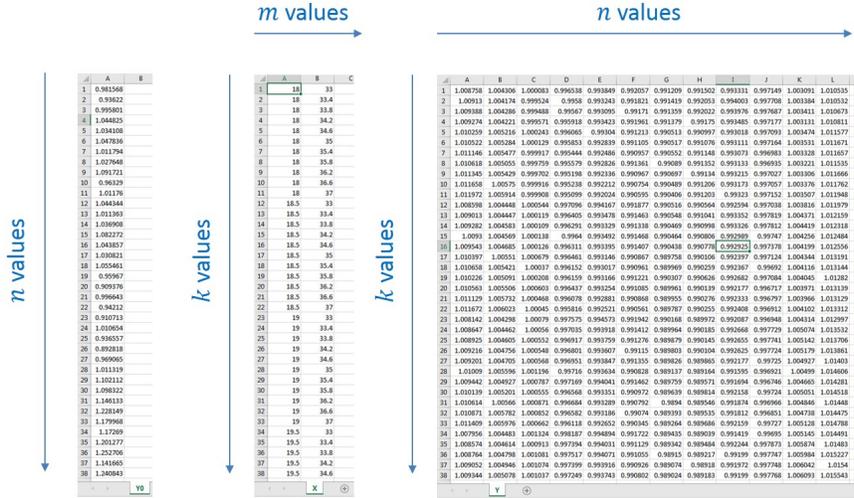


Figure 2: Acceptable format for y (left), Π (middle), and Φ (right).

3.2 Standard Usage

Once the user made sure the data is in the right format, it needs to be uploaded individually, i.e. Π , Φ , and y into M.o.R using the respective **Upload** and **browse** buttons. If no V^{-1} is specified, M.o.R. assumes an i.i.d. error model with $\mu = 0$, and σ defined in the **Provide Sigma for V matrix** field, see Fig. 3. The user can also provide a specific V , or rather V^{-1} . M.o.R. automatically detects the dimensions of the data, and distinguishes two cases:

- 1) $V^{-1} \in \mathbb{R}^n$: M.o.R. treats the entries as the diagonal entries of V^{-1}
- 2) $V^{-1} \in \mathbb{R}^{n \times n}$: M.o.R. treats the entries as the matrix V^{-1}

If prior knowledge shall be used, the user needs to provide $\mu_{\text{pri}} \in \mathbb{R}^m$, and $\Sigma_{\text{pri}} \in \mathbb{R}^{m \times m}$. Once the data is specified M.o.R. prints out the dimensions of the

M.o.R.

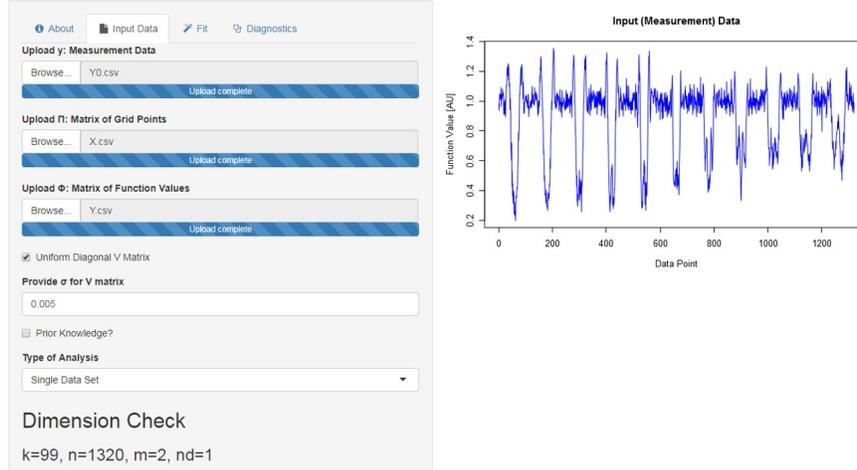


Figure 3: M.o.R. *Input Data* screen with completed data upload.

different vectors and matrices. Here k , n , and m correspond to the dimensions as given in Problem 3, and nd corresponds to the number of data sets uploaded, that is the number of columns of \mathbf{y} . Unless the Monte Carlo analysis shall be performed, M.o.R. sets $nd = 1$. The user then has to choose the type of analysis they want to perform in the **Type of Analysis** field, and can then proceed to the *Fit* screen, see Fig. 4. Note that if $nd > 1$, choosing the **Single Data Set** or **Bootstrapping** options will only perform the regression for the first data set. For more information about the different choices see Sections 3.3 and 3.5.

On the *Fit* screen, the user first needs to specify which type of radial basis function to use for the interpolation in the **Type of RBF** field, see Section 2.2. Afterwards the choice of the hyperparameter r needs to be specified, M.o.R. can either be given a specific value for r to use, or it can determine the best value, within user-defined boundaries, and with a maximum number of iterations specified using the **Upper Bound for r** and **Maximum # of Iterations** sliders. The value of r is set/optimized by pushing the **Set/Optimize Hyperparameter r and Determine \mathbf{A}** button. The used value of r and the corresponding $l(r)$, see Eq. (33) are printed on screen. In the same step M.o.R. determines the coefficient matrix \mathbf{A} , see Eq. (22).

Once r and \mathbf{A} are determined, the actual minimization of the χ^2 function can be performed. If not otherwise specified, M.o.R. sets the bounds for the parameters to be the min/max values of the grid points in $\mathbf{\Pi}$. The user can change these by unchecking the **Default Bounds** box and using the **Range for Parameter # i** sliders to set the bounds for each of the m parameters individually. Again, the user can specify the maximum number of iterations, the latter individually for the PSO and gradient-based parts using the **Maximum # of PSO Iterations** and **Maximum # of Iterations**, respectively.

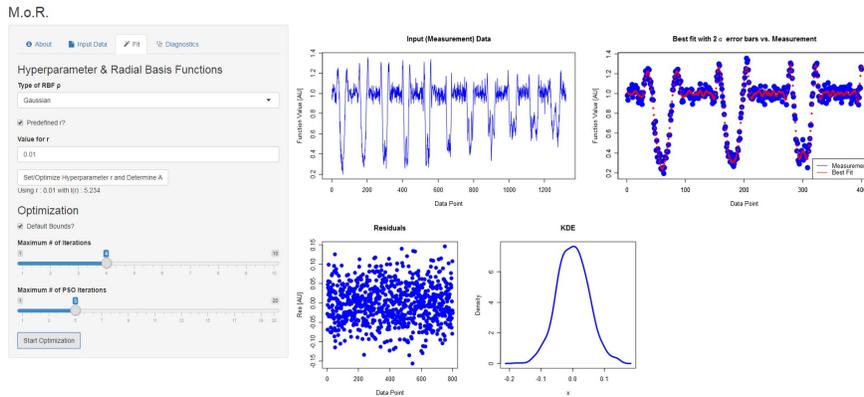


Figure 4: M.o.R. *Fit* screen after finished optimization.

Finally, the *Diagnostics* screen provides information about the fit in terms of the reduced χ^2 value, i.e. Eq. (4) divided by the degrees of freedom $n - m$, the best fit parameters, and the estimated covariance matrix, based on Eqs. (12) and (13). Furthermore M.o.R. provides a plot of the residuals and a plot of the kernel density estimate (KDE) based on the residuals to enable the user to check their assumptions about the initial error distribution. If the user wants additional information, they can also use M.o.R. to provide a plot of the χ^2 surface in dependence on two parameters. If the number of parameters m is larger than 2, the remaining parameters will be fixed to their best fit values. It is also possible to download the results for further studies, i.e. the best fit parameters, the covariance matrix, and the matrix with the χ^2 surface values if it has been calculated.

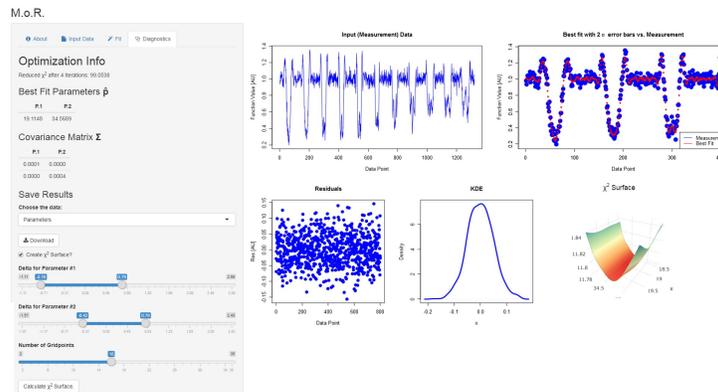


Figure 5: M.o.R. *Diagnostics* screen with χ^2 surface (lower right).

3.3 Monte Carlo Error Estimation

If the Monte Carlo option has been chosen, M.o.R. will determine the best fit for each of the nd individual data sets. The fit screen shows the individual estimated parameter values and the corresponding χ^2 values. The residual and KDE plots are for the final data set. The *Diagnostics* screen gives both the

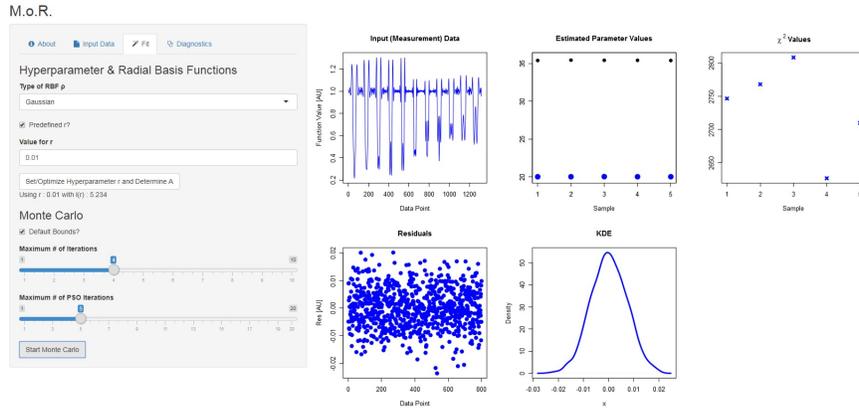


Figure 6: M.o.R. *Fit* screen after finished Monte Carlo optimization.

sample covariance matrix based on the nd parametric estimates using Eqs. (34) and (35), and the approximated covariance matrix based on Eqs. (12) and (13) for the final data set. The χ^2 surface, if plotted, is also based on the final data set.

3.4 Bootstrapping

If the bootstrapping option has been chosen, M.o.R. will use the bootstrap algorithm described in Section 2.6 to estimate the parametric uncertainties, based on the number of samples n_s specified by the user with the **Number of Bootstrap Samples** slider. Once the calculations are finished, the *Fit* screen shows the individual estimated parameter values and the corresponding χ^2 values using the n -dimensional identity matrix as \mathbf{V}^{-1} . The residual and KDE plots are for the final data set. The *Diagnostics* screen gives the sample covariance matrix based on the n_s parametric estimates using Eqs. (34) and (35). The χ^2 surface, if plotted, is based on the final data set.

3.5 Saving the Results

The user is given the opportunity to save the results of the data analysis using the **Save Results** dialogue. Here the user can choose whether they want to save the best fit parameters, the residuals, the covariance matrix, or the values of the

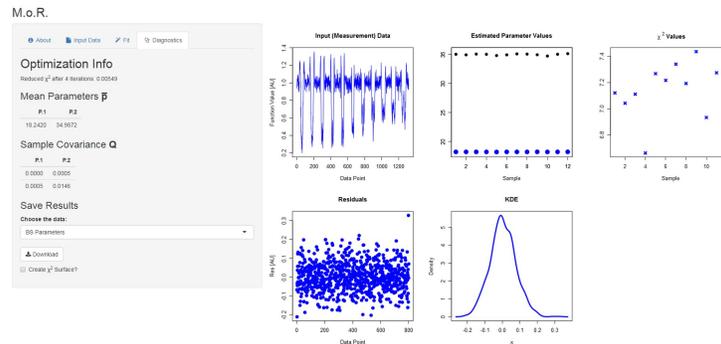


Figure 7: M.o.R. Diagnostics screen after finished bootstrapping.

χ^2 surface if it has been calculated for further investigations. All data are saved as CSV files.

Acknowledgments

The authors would like to thank Gabriel Sarmanho for helping with the R specific questions.

References

- [1] C. BENDTSEN., *pso: Particle swarm optimization*. <https://CRAN.R-project.org/package=pso>, 2012. R package version 1.0.3.
- [2] R. EBERHART AND J. KENNEDY, *A new optimizer using particle swarm theory*, Proceedings of the Sixth International Symposium on Micro Machine and Human Science, (1995).
- [3] B. EFRON, *Bootstrap Methods: Another Look at the Jackknife*, Ann. Statist., 7 (1979).
- [4] R. FLETCHER, *Practical methods of optimization*, John Wiley & Sons, 2013.
- [5] C. W. GROETSCH, *Inverse problems: activities for undergraduates*, Cambridge University Press, 1999.
- [6] M.-A. HENN, R. M. SILVER, J. S. VILLARRUBIA, N. F. ZHANG, H. ZHOU, B. M. BARNES, B. MING, AND A. E. VLADÁR, *Optimizing hybrid metrology: rigorous implementation of bayesian and combined regression*, Journal of Micro/Nanolithography, MEMS, and MOEMS, 14 (2015).
- [7] J. KAIPIO AND E. SOMERSALO, *Statistical and Computational Inverse Problems*, Springer Science & Business Media, Berlin, 2006.

- [8] C. T. KELLEY, *Iterative methods for optimization*, SIAM Frontiers in Applied Mathematics, (1999).
- [9] D. LOWE, *Multi-variable functional interpolation and adaptive networks*, Complex Systems.
- [10] M. MONGILLO, *Choosing basis functions and shape parameters for radial basis function methods*, SIAM Undergraduate Research Online, 4 (2011), pp. 190–209.
- [11] A. W. MOORE, *Cross-validation for detecting and preventing overfitting*, School of Computer Science Carneigie Mellon University, (2001).
- [12] J. C. NASH, R. VARADHAN, AND G. GROTHENDIECK, *Package 'optimx'*. <ftp://cran.r-project.org/pub/R/web/packages/optimx/optimx.pdf>. Accessed: 2017-06-19.
- [13] A. TARANTOLA, *Inverse problem theory and methods for model parameter estimation*, SIAM, 2005.
- [14] A. N. TIKHONOV, *On the stability of inverse problems*, Dokl. Akad. Nauk SSSR, 39 (1943).
- [15] T. WEISE, *Global optimization algorithms-theory and application*, Self-published, 2 (2009).
- [16] N. F. ZHANG, R. M. SILVER, H. ZHOU, AND B. M. BARNES, *Improving optical measurement uncertainty with combined multitool metrology using a Bayesian approach*, Applied optics, 51 (2012).
- [17] N. F. ZHANG, R. M. SILVER, H. ZHOU, AND B. M. BARNES, *Use of Bayesian Statistics to Improve Optical Measurement Uncertainty by Combined Multi-Tool Metrology*, Advances in Mathematical and Computational Tools in Metrology and Testing X, (2015).