T A

Č R

Research report

Advanced mathematical and statistical methods in evaluating instrumented indentation measurements

Souhrnná výzkumná zpráva projektu TAČR TJ02000203

Anna Charvátová Campbell[1], Zdeňka Geršlová[2], Vojtěch Šindlář[2], Radek Šlesinger[1], Jiří Šperka[1], Gejza Wimmer[2], Stanislav Zámečník[2]

[1]Department of Primary Nanometrology and Technical Length, Czech Metrology Institute
[2]Department of Mathematics and Statistics, Masaryk University

31 March 2021

CZECH METROLOGY INSTITUTE

MUNI

Contents

Part I

Research project

Chapter 1

Project overview

1.1 Basic project information

Project name in English:
Advanced mathematical and statistical methods in evaluating instrumented indentation measurements

Project name in Czech:
Pokročilé matematické a statistické metody ve vyhodnocování měření instrumentovanou indentací

Project identification code: TJ02000203

Project acronym: MathForIndentation

Key words in English:
Instrumented indentation; uncertainties; Oliver-Pharr method; orthogonal data regression; experimental design, Niget software, R (programming language).

Key words in Czech:
Instrumentovaná indentace; nejistoty; Oliverova-Pharrova metoda; ortogonální regrese; návrh experimentů, Niget software, R (programovací jazyk).

Funding agency: Technology Agency of the Czech Republic

Call for proposals the project is submitted in: ZÉTA 2

Project participants:

Český metrologický institut (CMI)
Main participant (Czech Metrology Institute, www.cmi.cz)

Masarykova univerzita (MU)
Other participant (Masaryk University, www.muni.cz)

Start of the project solution: April 2019

Completion of the project solution: March 2021

1.2 Project team

Researchers

Mgr. Radek Šlesinger, Ph.D.	Lead solver, CMI
Mgr. Jiří Šperka, Ph.D.	Member of the Research Team, CMI
Mgr. Zdeňka Geršlová	Other solver, MU
Mgr. Vojtěch Šindlář	Member of the Research Team, MU

Mentors

Mgr. Anna Charvátová Campbell, Ph.D.	CMI
prof. RNDr. Gejza Wimmer, DrSc.	MU

Other personnel

Mgr. Stanislav Zámečník MU

1.3 Background and state of knowledge

Progress in nanotechnology and the development of new materials (nanocomposites, thin layers, nanoparticles) and currently also the study of biological samples require good quality methods for the analysis of local mechanical properties. Instrumented indentation testing (also known as depth-sensing indentation, continuous-recording indentation, ultra-low-load indentation, and nanoindentation [10]) is one of the most common techniques for the study of a wide range of mechanical properties of the materials, especially hardness and Young's modulus but also yield stress, viscoelastic parameters, material hardening and other. It consists, in its most common form, of pressing a hard tip with geometrically well-defined shape (typically diamond) into the sample while continuously measuring both load and indentation depth. The maximum indentation depths range typically from hundreds of nanometers up to tens of micrometers; typical loads range from tens of micronewtons up to several newtons. This technique enable detailed point-to-point measurements localised on the sample surface and also the mapping of mechanical properties of nonhomogeneous materials thanks to a lateral resolution on the order of several micrometers.

Hardness and Young's modulus are the two most commonly studied mechanical properties. Currently the standard procedure for their determination from the load-depth data is the method by Oliver and Pharr [27]. This method is standardized in the ISO 14577-1 standard (Metallic materials — Instrumented indentation test for hardness and materials parameters). The procedure is fairly complex including regressions of load-depth curves. However, the ISO standard does not list all details of the procedure, such as the choice of interval for the regression or the method of regression itself. This can lead to differences between results coming from different implementations.

The most commonly used regression method is least squares, which is also recommended in the ISO standard. However, this allows to treat only errors in the dependent variable y. While this is sufficient for many applications, in instrumented indentation the errors in both dependent and independent variable are inevitable. Furthermore we can expect also correlations between dependent and independent variables and between independent variables among each other. The inclusion of correlations has been partially addressed in the metrological setting for correlations between x and y pairs or straight lines in e.g. [23] and [16]. In instrumented indentation the bias of least squares method due to errors in the input data has been addressed in [15]. To the authors' best knowledge the question of correlations in indentation data has not been treated so far.

The Oliver-Pharr method makes use of the geometric shape of the tip. The dependence of the area of the cross section of the tip at a given height on this height is called area function. For its determination a series of indentations are measured on a reference sample, resulting in a set of discrete depth-area pairs. Details are described in section 8. The use of these discrete data via e.g. interpolation is possible, however, in practice the data are fitted by a suitable function. The most common choice is a fractional polynomial as suggested by Oliver and Pharr [27], but ordinary polynomials can be used as well [8]. A rigorous discussion how to choose the function and e.g. its degree is missing so far, to the best of our knowledge. Different measurement schemes can be used. Two examples are clustered data, consisting of multiple indentations at different loads, and equally spread data. The impact of the choice of the measurement scheme has not been studied so far either. As the data are all measured on one and the same reference sample, the mechanical properties of this sample act as a source of correlation between different variables leading to a highly nontrivial covariance matrix which must be correctly treated in the fitting procedure.

Monte Carlo analysis of uncertainties as described in [11] is based on propagating probability distribution functions of input variables. Data sets of input variables are drawn randomly according to their probability distribution functions and evaluated as if they were real measurements. The result is an empirical distribution function of the output variables. In nanoindentation this can be used and is implemented in Niget software. However, the computation time is long and grows rapidly with the number of input variables making the method impractical for every day use. Other methods for generating random numbers are available which can cover the range of the values more effectively, such as Latin Hypercube Sampling [25] or Orthogonal sampling [28] are not mentioned in metrological literature, to our best knowledge.

1.4 Objectives and achieved goals

This research report presents the design, implementation and verification of mathematically correct methods suitable for the key phases of indentation measurements, particularly:

- calibration of the tip contact area function,
- nonlinear regression for correlated data with errors in both x and y variables,
- uncertainty analysis throughout the whole evaluation process.

All developed methods are suitable for indentation metrology. Nonlinear regression has been verified on datasets from literature [21, 22]. All methods have been integrated into Niget software for complex analysis of indentation data, partly in the main graphical application, partly as standalone scripts. Moreover, an independently usable method for nonlinear regression of correlated data has been implemented for the statistical software R for use by a wider statisticians community. All the software created is freely distributable under a free software license.

The computing demands are moderate allowing a quick analysis. Some computations can require a significant amount of memory.

The presented measurement evaluation methods are based on the following achievements:

- Nonlinear orthogonal regression allowing general correlations between x and y data, without restrictions on the fitted function.
- Estimations of uncertainties.
  Latin Hypercube Sampling has been used as a new approach to generate pseudo-random numbers. The resulting empirical distributions can be used for further computations in Niget software.

- Design of experiments.
  Using the general nonlinear orthogonal regression developed in this project it has been possible to compare the statistical performance of different functions used for as area function as well as for different measurement setups. Preliminary recommendations for the measurement setup for the calibration area functions can be given, though more data from different instruments will be needed.

  Statistically suitable and optimalised measurement procedure (minimize uncertainty of calculated contact area) for the calibration of the tip area function using reference sample with related recommendations is presented. Suitable representation of the area function is proposed.

Part II

Introduction

Chapter 2

Evaluation of instrumented indentation measurements

In this section we list the most important aspects for the evaluation of hardness and Young's modulus using the Oliver-Pharr method [27]. More details can be found in literature, for an introduction see e.g. [7]. The Oliver-Pharr method and the calibration of the nanoindentor are standardized in the ISO standards ISO 14577-1:2015 [13] and ISO 14577-2:2015 [14].

2.1 Determination of the indentation hardness and Young's modulus

The Oliver Pharr method works only with the unloading part of the load-depth data, the tip area function and, optionally, with the Poisson's ratio of the sample and tip and the Young's modulus of the tip. The area function must be determined separately, either by geometrical considerations or experimentally, e.g. by the method described in section 8. The modulus of the tip as well as the two Poisson's ratios are usually taken from literature. The procedure is as follows:

1. A suitable part of the unloading curve (h_i, F_i) for $i = 1, \ldots, N$ is fitted by a power law function
$$F = \alpha(h - h_\mathrm{p})^m. \tag{2.1}$$
Here α, h_p and m are parameters which are found by regression. The ISO standard recommends a range of 20–98 %, however this may be adapted to the given sample.

2. The auxiliary parameter ε is calculated from the power m
$$\varepsilon = m\left(1 - \frac{2(m-1)\Gamma\left(\frac{m}{2(m-1)}\right)}{\sqrt{\pi}\Gamma\left(\frac{1}{2(m-1)}\right)}\right), \tag{2.2}$$
Γ is the Gamma-function.

3. The slope of the power law function at maximum load F_{max} is calculated is
$$S = \left.\frac{\mathrm{d}F}{\mathrm{d}h}\right|_{F=F_\mathrm{max}}. \tag{2.3}$$

4. The contact depth is calculated as
$$h_c = h_{max} - \varepsilon\frac{F_\mathrm{max}}{S}. \tag{2.4}$$

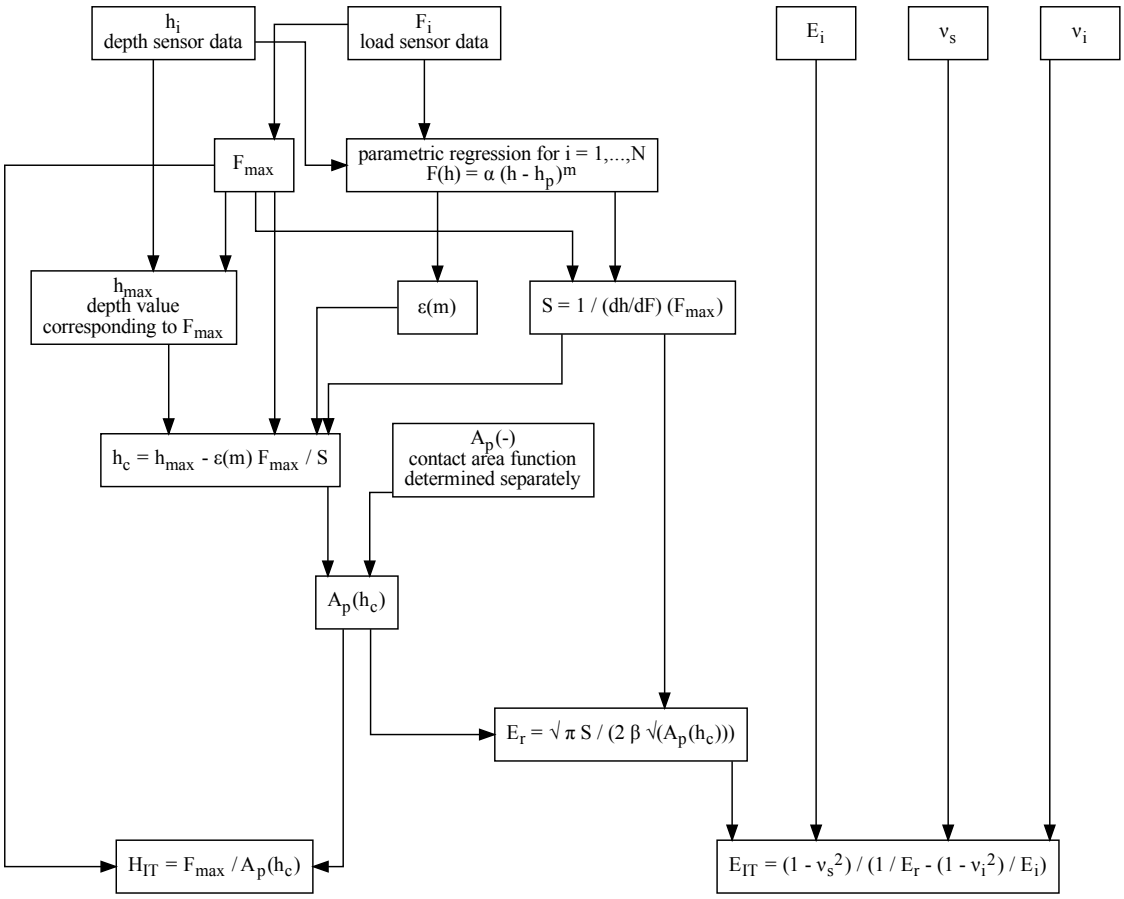


Figure 2.1: Computation schema of indentation modulus and hardness.

5. The contact area A_p is evaluated at the contact depth h_c.

6. The indentation hardness can be evaluated as

$$H_\mathrm{IT} = \frac{F_\mathrm{max}}{A_\mathrm{p}(h_\mathrm{c})}. \tag{2.5}$$

7. The reduced modulus of elastic contact can be calculated as

$$E_\mathrm{r} = \sqrt{\pi}\frac{S}{2\beta\sqrt{A_\mathrm{p}(h_\mathrm{c})}}, \tag{2.6}$$

   where β is a constant used to account for the difference between a conical tip and the real tip.

8. The indentation modulus is

$$E_\mathrm{IT} = \frac{1-\nu^2}{\frac{1}{E_\mathrm{r}} - \frac{1-\nu_\mathrm{i}^2}{E_\mathrm{i}}}. \tag{2.7}$$

   Here ν is the Poisson's value of the sample and ν_i and E_i are the Poisson's value and the modulus of the tip.

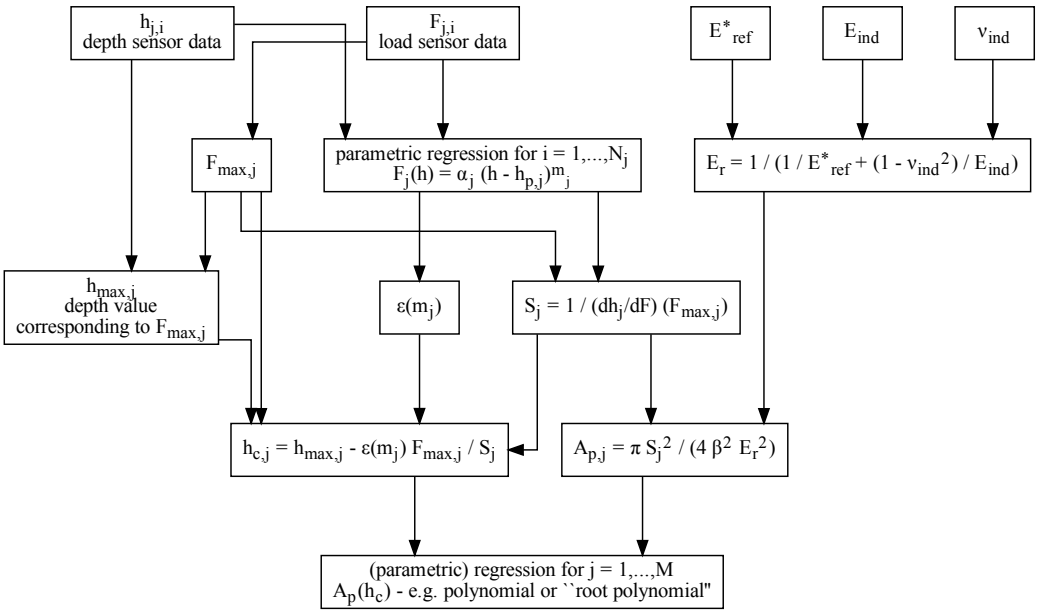


Figure 2.2: Computation schema of contact area function calibration.

2.2 Calibration of the tip area function

The most popular method for calibrating the tip area function is by performing a set of indentations on a reference sample with known Young's modulus E_{IT} or plain strain modulus E^*. Other methods e.g. using atomic force microscopy exist but are not studied in this project.

The contact area function calibration involves fitting a curve through a series of M points given by the tuples $((h_{\mathrm{c}})_j, (A_{\mathrm{p}})_j)$ for $j = 1, \ldots, M$. Each such tuple is typically determined from a single indentation unloading curve, represented by depth-load values $((h_j)_i, (F_j)_i)$ for $i = 1, \ldots, N_j$. (Although alternative approaches exist, where multiple points can be extracted from a single indent, thus saving time needed to perform the necessary measurements, the function fitting is present anyway.) For each unloading curve the contact depth and slope as described in the previous section are determined. The inverse relation to equation (2.6) is used and the contact area is calculated as

$$A_{\mathrm{p}} = \frac{\pi}{4\beta^2}\frac{S^2}{E_{\mathrm{r}}^2}. \tag{2.8}$$

The reduced modulus of contact can be calculated either using Young's modulus or the plain strain modulus of the sample

$$\frac{1}{E_{\mathrm{r}}} = \frac{1-\nu^2}{E_{\mathrm{IT}}} + \frac{1-\nu_{\mathrm{i}}^2}{E_{\mathrm{i}}} \tag{2.9}$$

or

$$\frac{1}{E_{\mathrm{r}}} = \frac{1}{E^*} + \frac{1-\nu_{\mathrm{i}}^2}{E_{\mathrm{i}}}, \tag{2.10}$$

depending on the information available. The schema in Figure 2.2 essentially captures the calculations.

Note that the values of contact depth and area are not independent, and neither are the contact area values at different depths. Actually, multiple correlation is always present in the contact area calibration data:

1. For each point i, the contact depth h_{ci} and the respective area A_{p_i} are correlated via α, m, and $F_{\max}$.

2. For each pair of points i, j, the respective contact areas A_{p_i} and A_{p_j} are correlated via the reduced contact modulus of the reference sample E_{r}.

We are aware that the procedures investigated in this project (nonlinear function fitting, uncertainty propagation, handling correlated data) are still being further developed and improved, and alternative approaches and methods exist and have been under investigation. However, one can reasonably assume that any of the alternatives requires the same apparatus when measurement uncertainty becomes involved.

Chapter 3

Niget software

Niget is a free open source software for the general analysis of nanoindentation data. The current version of Niget software can be downloaded from `http://nanometrologie.cz/niget/`.

Niget has a graphical user interface which allows for comfortable usage and visualization.

Niget is written in the C language, with its graphical user interface based on the GTK2 toolkit. Its data plotting capabilities draw on the libraries from the Gwyddion scanning probe microscopy data analysis software [26].

The first version of Niget software with range of included analysis methods has been presented in 2019 in an original software publication [5], freely available in open access format. The aspect of free software license has several advantages from metrology point of view:

- Users can have insight to all data processing procedures.
- Users can evaluate their data in a reproducible way and compare results of their measurements
- Users can read the source code and possibly contribute to development or suggest software improvement.

The software implements several methods for the analysis of nanoindentation data, namely the Oliver-Pharr analysis, tangent method, Hertz analysis, Oliver's two slopes method, differential hardness, stiffness evaluation and elastic-plastic work analysis. It attempts to use as few built-in constants as possible and to leave all other parameters for the user to choose. The individual evaluation tools are independent, they only use shared functions, data types, and structures. The open code allows a user to follow through all steps of the evaluation.

Development of Niget is still under progress, with features being regularly added. Currently, only several formats are supported for the input of load depth curves and tip area functions. For processing data from an unsupported instrument, the data should be exported from the instrument software and converted to one of the supported text formats. For further work with data already preprocessed (defining important segments of the load-depth curve) in Niget, an own file format is used.

Internally, Niget consists of two principal components:

- A base software library providing data structures and evaluation procedures. The library is licensed under LGPLv2.1, which allows dynamic linking by other software.
- A graphical user interface program using functionality provided by the library. Niget GUI is licensed under GPLv2.1.

Due to its open nature, Niget provides a convenient platform for including new methods for data evaluation as well as for inclusion of existing software packages for specific computation tasks. When investigating existing software packages to be included in Niget, two key criteria are followed:

- a free license, compatible with LGPL, allowing further redistribution of the software
- a proven implementation, preferably in C or Fortran, with detailed documentation.

State at the beginning of the project Basic uncertainty analysis is implemented in Niget. Curve fitting is performed in most cases using ODRPACK95 software package [36]. This allows for the consideration of uncertainties in both x and y variables but does not allow for including correlation between variables. Uncertainty propagation is calculated using Gauss's law of uncertainty propagation. Basic Monte Carlo treatment of uncertainties as described in GUM-S1 [11] is partially implemented but due to its large time demands it is not very practical.

Part III

Mathematical and statistical methods for data regression and uncertainty propagation

Chapter 4

Parametric nonlinear regression

Due to the nature of data this project has focused on, in this report we only deal with methods and software applicable to fitting nonlinear functions, and do consider tools for straight-line fitting.

In so called nonlinear least squares (NLS) method, the goal is to minimize the sum of squared vertical residuals (distances between the y-values of data points and the values of the fitted function). However, this method is not suitable for data where there are errors also in the explanatory data (x), or they also have an uncertainty (e.g. when both variables come from measurement, none of which can be considered as accurate and precise).

The goal is then to minimize the sum of squared orthogonal residuals (the actual Euclidean distances between the points and the curve). Such a method is then called Orthogonal Distance Regression (ODR).

4.1 Overview of existing methods and software

ODRPACK A pioneering effort for fitting errors-in-variables data by nonlinear functions was ODRPACK, the algorithm presented in [3]. Its software implementation [4] was written in FORTRAN 77, and is available at `https://www.netlib.org/odrpack/`, distributed as public domain. ODRPACK provides a number of features, namely:

- both ODR and NLS using an implementation of Levenberg-Marquardt method,
- both scalar and vector functions of multiple variables,
- both explicit and implicit functions,
- weighted regression, allowing to set weights for individual values of both explanatory and response variable.

This concept, however, does not allow to include arbitrary covariance of the input data; only diagonal covariance matrices can be treated, typically by prescribing weights of the individual values as squared reciprocals of their uncertainties.

The original ODRPACK software has later been rewritten in Fortran 95 as ODRPACK95 [36] (`https://dl.acm.org/doi/abs/10.1145/1268776.1268782`, available as public domain), utilizing features of the more modern programming language dialect, and extending the functionality of the original algorithm with bounded regression, which allows to set bounds on the parameter estimates. Behavior of the algorithm for unbounded regression remained unchanged.

Nowadays, ODRPACK is used by `scipy.odr` Python module (`https://docs.scipy.org/doc/scipy/reference/odr.html`), and ODRPACK95 is included in Igor Pro data analysis software[1].

[1] `https://www.wavemetrics.com/products/igorpro/dataanalysis/curvefitting/errorsinvariables`

ODRPACK95 has been included in Niget as the default fitting method for nonlinear functions.

WTLS and CCC In [22], a variant of WTLS (Weighted Total Least Squares) algorithm for functions other than straight-line was presented. The algorithm was implemented in MATLAB, and is not restricted to any particular form of the covariance matrix of the variables. However, it is usable only for linear models (that is, linear in parameters; linear combinations of terms nonlinear in x, e.g. polynomials, are allowed). The WTLS algorithm has been included in INRIM CCC (Calibration Curve Computing) software version 1.3, which can be obtained for free at `https://www.inrim.eu/research-development/quality-life/ccc-software`.

A new version 2.0 of the CCC software has recently been presented [21], which features an improved WTLS code.

ONLS The implementation of orthogonal nonlinear least squares regression (ONLS) in the R software is available in `onls` package [30] also based on `ODRPACK`. The package allows to fit and plot an ONLS model and compute orthogonal residuals for the estimated function. Common generics functions (as `summary` for printing results, `confint` for computing confidence intervals, etc.) are also part of the package. The main function `onls` enables computing orthogonal weighted least squares estimation through `weights` argument, but using general input covariance matrix is not possible.

4.2 New algorithm

Fitting nonlinear function with uncertainties in both, the dependent and independent variables through minimization of orthogonal distance principle is commonly used in nanoindentaion (and is also implemented in `ODRPACK`).

An alternative approach to this problem has been proposed by the project mentor Gejza Wimmer. The method is based on linearization of a nonlinear differentiable function by Taylor series. Using this procedure we convert the original nonlinear problem to linear regression model with type II. linear constraints, whose statistical properties are principally described in [17] and extended in [18]. The advantage of the method is both the possibility of implementation any covariance matrix of input variables and straightforward obtaining the output covariance matrix of estimated parameters. The main idea of the algorithm, linearization of a model via Taylor series expansion and using iteration procedure to estimate the parameters, is a generalization of a method for ellipse fitting presented in [20] and [35].

4.2.1 Theoretical background

Consider a problem of fitting a function to data with errors occurring in both variables, such as the measurements of two physical quantities. Let's assume data consisting of N pairs of observed values, where the observed measurement x_i is a realization of random variable X_i and the observed measurement y_i is a realization of random variable Y_i. Then (according notation in [35])

$$X_i = \mu_i + \varepsilon_{X_i},\ Y_i = \nu_i + \varepsilon_{Y_i} \qquad \text{for } \ i = 1, \ldots, N,$$

where μ_i and ν_i are the (unknown) true values of measured quantities and measurements errors $\varepsilon_{X_i},\ \varepsilon_{Y_i}$ are independent random variables satisfying $\varepsilon_{X_i} \sim N\left(0, u_{X_i}^2\right), \varepsilon_{Y_i} \sim N\left(0, u_{Y_i}^2\right)$.

In vector notation we have

$$\mathbf{X} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{pmatrix} = \begin{pmatrix} \mu_1 + \varepsilon_{X_1} \\ \mu_2 + \varepsilon_{X_2} \\ \vdots \\ \mu_N + \varepsilon_{X_N} \end{pmatrix} = \boldsymbol{\mu} + \boldsymbol{\varepsilon_X}, \qquad \mathbf{Y} = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_N \end{pmatrix} = \begin{pmatrix} \nu_1 + \varepsilon_{Y_1} \\ \nu_2 + \varepsilon_{Y_2} \\ \vdots \\ \nu_N + \varepsilon_{Y_N} \end{pmatrix} = \boldsymbol{\nu} + \boldsymbol{\varepsilon_Y}.$$

So, we consider an observation vector

$$\begin{pmatrix} \mathbf{X} \\ \mathbf{Y} \end{pmatrix} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \\ Y_1 \\ Y_2 \\ \vdots \\ Y_N \end{pmatrix},$$

for which it holds

$$\mathrm{E}\begin{pmatrix} \mathbf{X} \\ \mathbf{Y} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{\nu} \end{pmatrix}, \qquad \mathrm{cov}\begin{pmatrix} \mathbf{X} \\ \mathbf{Y} \end{pmatrix} = \begin{pmatrix} \mathbf{\Sigma_X} & \mathbf{\Sigma_{XY}} \\ \mathbf{\Sigma_{YX}} & \mathbf{\Sigma_Y} \end{pmatrix} = \mathbf{\Sigma}, \tag{4.1}$$

where $\mathbf{\Sigma}$ is $2N \times 2N$ positive semidefinite matrix.

Suppose the relationship between $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ is given by nonlinear function g dependent on m-dimensional vector of parameters $\boldsymbol{p}$, then unknown parameters of the considered model are $\mu_1, \mu_2, \ldots, \mu_N, \nu_1, \nu_2, \ldots, \nu_N, p_1, p_2, \ldots, p_m$ associated through relationships

$$\nu_i = g(\mu_i, p_1, \ldots, p_m), \qquad i = 1, 2, \ldots, N, \tag{4.2}$$

where g is a known twice continuously differentiable function. So we have a regression model (4.1) with nonlinear constraints (4.2) on parameters.
Let

$$\boldsymbol{\mu}^{(0)} = \begin{pmatrix} \mu_1^{(0)} \\ \mu_2^{(0)} \\ \vdots \\ \mu_N^{(0)} \end{pmatrix}, \qquad \boldsymbol{\nu}^{(0)} = \begin{pmatrix} \nu_1^{(0)} \\ \nu_2^{(0)} \\ \vdots \\ \nu_N^{(0)} \end{pmatrix}, \qquad \boldsymbol{p}^{(0)} = \begin{pmatrix} p_1^{(0)} \\ p_2^{(0)} \\ \vdots \\ p_N^{(0)} \end{pmatrix}$$

be the initial values of the parameters. If this values are close enough to the real values of parameters, we can linearize the constraints (4.2) by first-order Taylor expansion about initial values $\boldsymbol{p}^{(0)}, \boldsymbol{\mu}^{(0)}, \boldsymbol{\nu}^{(0)}$ and for $i = 1, 2, \ldots, N$ we get

$$\begin{aligned} \nu_i - g(\mu_i, p_1, \ldots, p_m) = \nu_i^{(0)} - g(\mu_i^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)}) + (\nu_i - \nu_i^{(0)}) - g'_{\mu_i}(\mu_i^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)})(\mu_i - \mu_i^{(0)}) + \\ - g'_{p_1}(\mu_i^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)})(p_1 - p_1^{(0)}) - g'_{p_2}(\mu_i^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)})(p_2 - p_2^{(0)}) + \cdots + \\ - g'_{p_m}(\mu_i^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)})(p_m - p_m^{(0)}) = 0, \end{aligned}$$

where $g'_{p_i}(\cdot)$ denotes partial derivative of a function g with respect to p_i in given point.
Assuming

$$\Delta\boldsymbol{\mu}^{(0)} = \begin{pmatrix} \mu_1 - \mu_1^{(0)} \\ \mu_2 - \mu_2^{(0)} \\ \vdots \\ \mu_N - \mu_N^{(0)} \end{pmatrix}, \qquad \Delta\boldsymbol{\nu}^{(0)} = \begin{pmatrix} \nu_1 - \nu_1^{(0)} \\ \nu_2 - \nu_2^{(0)} \\ \vdots \\ \nu_N - \nu_N^{(0)} \end{pmatrix}, \qquad \Delta\boldsymbol{p}^{(0)} = \begin{pmatrix} p_1 - p_1^{(0)} \\ p_2 - p_2^{(0)} \\ \vdots \\ p_N - p_N^{(0)} \end{pmatrix},$$

the vector notation for linearized constraints is

$$\mathbf{b}_{N,1}^{(0)} + \mathbf{B}_1^{(0)} \begin{pmatrix} \Delta\boldsymbol{\mu}^{(0)} \\ \Delta\boldsymbol{\nu}^{(0)} \end{pmatrix} + \mathbf{B}_2^{(0)} \Delta\boldsymbol{p} = \mathbf{0}_{N,1}, \tag{4.3}$$

where

$$\mathbf{b}_{N,1}^{(0)} = \begin{pmatrix} \nu_1^{(0)} - g(\mu_1^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)}) \\ \nu_2^{(0)} - g(\mu_2^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)}) \\ \vdots \\ \nu_N^{(0)} - g(\mu_N^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)}) \end{pmatrix},$$

$$\mathbf{B}_1^{(0)} = \left(\begin{array}{cccc|c} -g'_{\mu_1}(\mu_1^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)}) & 0 & \cdots & 0 & \\ 0 & -g'_{\mu_2}(\mu_2^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)}) & \cdots & 0 & \mathbf{I}_{N,N} \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & \cdots & -g'_{\mu_N}(\mu_N^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)}) & \end{array} \right)_{N,2N},$$

$$\mathbf{B}_2^{(0)} = \begin{pmatrix} -g'_{p_1}(\mu_1^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)}) & -g'_{p_2}(\mu_1^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)}) & \cdots & -g'_{p_m}(\mu_1^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)}) \\ -g'_{p_1}(\mu_2^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)}) & -g'_{p_2}(\mu_2^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)}) & \cdots & -g'_{p_m}(\mu_2^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)}) \\ \vdots & \vdots & \cdots & \vdots \\ -g'_{p_1}(\mu_N^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)}) & -g'_{p_2}(\mu_N^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)}) & \cdots & -g'_{p_m}(\mu_N^{(0)}, p_1^{(0)}, \ldots, p_m^{(0)}) \end{pmatrix}_{N,m},$$

and also $\operatorname{rank}(\mathbf{B}_1^{(0)}) = N$, $\operatorname{rank}(\mathbf{B}_2^{(0)}) = m$, $\operatorname{rank}(\mathbf{B}_1^{(0)}, \mathbf{B}_2^{(0)}) = N$. We rewrite model (4.1) using the new parameters $\Delta\boldsymbol{\mu}^{(0)}$, $\Delta\boldsymbol{\nu}^{(0)}$, $\Delta\boldsymbol{p}^{(0)}$. We get

$$\mathrm{E} \begin{pmatrix} \mathbf{X} - \boldsymbol{\mu}^{(0)} \\ \mathbf{Y} - \boldsymbol{\nu}^{(0)} \end{pmatrix} = \begin{pmatrix} \Delta\boldsymbol{\mu}^{(0)} \\ \Delta\boldsymbol{\nu}^{(0)} \end{pmatrix}, \qquad \operatorname{cov} \begin{pmatrix} \mathbf{X} - \boldsymbol{\mu}^{(0)} \\ \mathbf{Y} - \boldsymbol{\nu}^{(0)} \end{pmatrix} = \boldsymbol{\Sigma},$$

where an observation vector and unknown parameters are

$$\begin{pmatrix} \mathbf{X} - \boldsymbol{\mu}^{(0)} \\ \mathbf{Y} - \boldsymbol{\nu}^{(0)} \end{pmatrix}, \qquad \begin{pmatrix} \Delta\boldsymbol{\mu}^{(0)} \\ \Delta\boldsymbol{\nu}^{(0)} \end{pmatrix},$$

and system of linear constraints is given by (4.3).

Using procedure described above we converted the original nonlinear model to a linear regression model with type II constraints on parameters. BLUE (best linear unbiased estimators) of parameters $\Delta\boldsymbol{\mu}^{(0)}$, $\Delta\boldsymbol{\nu}^{(0)}$, $\Delta\boldsymbol{p}^{(0)}$ in such a model are (according to [18]):

$$\begin{pmatrix} \widehat{\Delta\boldsymbol{\mu}^{(0)}} \\ \widehat{\Delta\boldsymbol{\nu}^{(0)}} \end{pmatrix} = \left(\mathbf{I}_{2N,2N} - \boldsymbol{\Sigma} \left(\mathbf{B}_1^{(0)} \right)^\top \mathbf{Q}_{11}^{(0)} \mathbf{B}_1^{(0)} \right) \begin{pmatrix} \mathbf{X} - \boldsymbol{\mu}^{(0)} \\ \mathbf{Y} - \boldsymbol{\nu}^{(0)} \end{pmatrix} - \boldsymbol{\Sigma} \left(\mathbf{B}_1^{(0)} \right)^\top \mathbf{Q}_{11}^{(0)} \mathbf{b}^{(0)}, \tag{4.4}$$

$$\widehat{\Delta\boldsymbol{p}^{(0)}} = -\mathbf{Q}_{21}^{(0)} \mathbf{B}_1^{(0)} \begin{pmatrix} \mathbf{X} - \boldsymbol{\mu}^{(0)} \\ \mathbf{Y} - \boldsymbol{\nu}^{(0)} \end{pmatrix} - \mathbf{Q}_{21}^{(0)} \mathbf{b}^{(0)}, \tag{4.5}$$

where

$$\begin{pmatrix} \mathbf{Q}_{11}^{(0)} & \mathbf{Q}_{12}^{(0)} \\ \mathbf{Q}_{21}^{(0)} & \mathbf{Q}_{22}^{(0)} \end{pmatrix} = \begin{pmatrix} \mathbf{B}_1^{(0)} \boldsymbol{\Sigma} \left(\mathbf{B}_1^{(0)} \right)^\top & \mathbf{B}_2^{(0)} \\ \left(\mathbf{B}_2^{(0)} \right)^\top & \mathbf{0} \end{pmatrix}^{-1} = \mathbf{M}_{\mathbf{B}}^{-\mathbf{1}}. \tag{4.6}$$

Estimators of the original parameters $\boldsymbol{\mu}$, $\boldsymbol{\nu}$, $\boldsymbol{p}$ are

$$\widehat{\boldsymbol{\mu}^{(0)}} = \boldsymbol{\mu}^{(0)} + \widehat{\Delta\boldsymbol{\mu}^{(0)}}, \qquad \widehat{\boldsymbol{\nu}^{(0)}} = \boldsymbol{\nu}^{(0)} + \widehat{\Delta\boldsymbol{\nu}^{(0)}}, \qquad \widehat{\boldsymbol{p}^{(0)}} = \boldsymbol{p}^{(0)} + \widehat{\Delta\boldsymbol{p}^{(0)}}.$$

The linearization and estimation procedure is repeated until the required accuracy (through selected convergence criterion) is reached. Let k denote the iteration step in which the convergence is achieved. Then resulting estimate of function parameters is $\widehat{\boldsymbol{p}^{(k)}}$ (an estimate obtained in k-th iteration step) with covariance matrix $-\mathbf{Q}_{22}^{(k)}$.

4.2.2 Procedure description

The algorithm consist of following parts:

(i) Initialization

- Set the initial values for the model parameters $\boldsymbol{\mu}^{(0)}$, $\boldsymbol{\nu}^{(0)}$. These are measurements taken from data.
- Set the starting values of the parameter estimates $\boldsymbol{p}^{(0)}$.
- Check logical argument indicating if we want to upgrade starting values of parameters $\boldsymbol{p}^{(0)}$.
  If *true*, starting values $\boldsymbol{p}^{(0)}$ are upgraded with the NLS fit (in R the `nlsLM` function from `minipack.lm` package is used).
  If *false*, the original starting values $\boldsymbol{p}^{(0)}$ are used (no upgrade is done).
- Set iteration step counter, $k = 0$.
- Set the limit for the number of iteration steps $k_{\max}$. If not otherwise stated, $k_{\max} = 100$.
- Set the iteration stopping threshold ϑ.
- Enter the iteration cycle (ii).

(ii) Iteration cycle

- Evaluate the vector $\mathbf{b}^{(k)}$ and matrices $\mathbf{B}_1^{(k)}$ and $\mathbf{B}_2^{(k)}$ (given as in equation 4.3).
- Evaluate matrices $\mathbf{Q}_{11}^{(k)}$, $\mathbf{Q}_{21}^{(k)}$, $\mathbf{Q}_{22}^{(k)}$ (given in 4.6) using Cholesky decomposition of a top-left block of matrix $\mathbf{M_B}$ (according to paragraph 4.3.1).
- Evaluate $\left(\Delta\widehat{\boldsymbol{\mu}^{(k)}}, \Delta\widehat{\boldsymbol{\nu}^{(k)}}\right)^\top$ by (4.4).
- Evaluate $\Delta\widehat{\boldsymbol{p}^{(k)}}$ by (4.5).
- Set $\boldsymbol{\mu}^{(k+1)} = \boldsymbol{\mu}^{(k)} + \Delta\widehat{\boldsymbol{\mu}^{(k)}}$.
- Set $\boldsymbol{\nu}^{(k+1)} = \boldsymbol{\nu}^{(k)} + \Delta\widehat{\boldsymbol{\nu}^{(k)}}$.
- Set $\boldsymbol{p}^{(k+1)} = \boldsymbol{p}^{(k)} + \Delta\widehat{\boldsymbol{p}^{(k)}}$.
- Update the iteration counter, $k = k + 1$.
- Proceed to check iteration criterium (iii).

(iii) Iteration criterium

- Check, if the following is true:
  There exists at least one $j = 1, 2, \ldots, m$ (where m is the length of vector $\boldsymbol{p}^{(k)}$) such that $|p_j^{(k+1)} - p_j^{(k)}|/|p_j^{(k)}| > \vartheta$ and $k < k_{\max}$.
  If *true*, continue in a cycle and go to step (ii) again.
  If *false*, stop the iteration cycle.

The algorithm has been called *OEFPIL*, which abbreviates "Optimal Estimate of Function Parameters by Iterated Linearization".

4.2.3 Confidence and prediction intervals

Assuming normality for the observation vector, $(1-\alpha)100\%$ confidence interval of function $g(x, p_1, \ldots, p_m)$ in some (exact) point x is defined as

$$\mathcal{CI}_{1-\alpha} = \left(g(x, \widehat{p_1^{(k-1)}}, \ldots, \widehat{p_m^{(k-1)}}) \pm u_{1-\alpha/2} \sqrt{\boldsymbol{\omega}_x^\top \{-\mathbf{Q}_{22}^{(k)}\} \boldsymbol{\omega}_x} \right),$$

where

$$\boldsymbol{\omega}_x = \begin{pmatrix} g'_{p_1}(x, p_1^{(k-1)}, \ldots, p_m^{(k-1)}) \\ g'_{p_2}(x, p_1^{(k-1)}, \ldots, p_m^{(k-1)}) \\ \vdots \\ g'_{p_m}(x, p_1^{(k-1)}, \ldots, p_m^{(k-1)}) \end{pmatrix}$$

and $u_{1-\alpha/2}$ is $(1-\alpha/2)$ quantile of standard normal distribution.

By plotting confidence intervals for a sufficiently dense grid of points we obtain pointwise confidence band of the estimated function.

Analogously, we can construct the prediction intervals. Assume another measurement Y_c in some point c and let it be normally distributed with variance σ_c^2, so $Y_c = g(c, p_1, \ldots, p_m) + \varepsilon_c$, where $\varepsilon_c \sim N(0, \sigma_c^2)$. Then $(1-\alpha)100\%$ prediction interval for Y_c is defined as

$$\mathcal{PI}_{1-\alpha} = \left(g(c, \widehat{p_1^{(k-1)}}, \ldots, \widehat{p_m^{(k-1)}}) \pm u_{1-\alpha/2} \sqrt{\boldsymbol{\omega}_c^\top \{-\mathbf{Q}_{22}^{(k)}\} \boldsymbol{\omega}_c + \sigma_c^2} \right).$$

Using prediction intervals for grid of points we obtain the pointwise prediction band.

4.3 Algorithm implementation

The algorithm was first implemented in R as a rather straightforward rewrite of the originally proposed variant for power-law fitting of indentation unloading curves. This implementation already showed very fast convergence rate of the algorithm (usually less than 10 iterations) for a large part of testing data. Despite the very promising results, in some cases there were problems with determining the input values of parameters close enough to the true values (which is a condition for using linearization via Taylor series expansion - see [18]). A solution to this was proposed, based on previous experience at CMI, to run a NLS fit with the original starting values, and use the NLS result as starting values for the OEFPIL algorithm. Because the standard `nls` function in R uses Gauss-Newton method, which is not robust against bad starting values, the `nls.LM` function from `minpack.lm` package was implemented into algorithm instead. The `nls.LM` is based on Levenberg-Marquardt algorithm[2], which is consistent with most NLS methods in other software (ODRPACK included). The issue with problematic convergence from far-from-optimum starting values has essentially been resolved by the described procedure for the indentation unloading curves.

After improvement and testing algorithm for power-law relationship in nanoindentation, the generalized version of the algorithm was developed. This generalization allows to use the procedure to any (twice continuously differentiable) function and is implemented in the R package `OEFPIL` described in Section 4.4.

While investigating the issues, a key step was identified to be the calculation of inverse of the system matrix $\mathbf{M_B}$ (defined in equation (4.6)). The standard `solve` procedure used to calculate the inverse sometimes failed due to poor conditioning of the matrix $\mathbf{M_B}$ (κ sometimes greater than 10^8). Replacing this procedure by the generalized inverse (Moore-Penrose pseudoinverse)

[2] https://rmazing.wordpress.com/2012/07/05/a-better-nls/

provided by the `ginv` function improved the algorithm behavior with problematic matrices, but a more efficient method has been proposed applying the block and symmetry structure of the matrix $\mathbf{M_B}$. Consider a formula for block matrix inverse [2, Proposition 3.9.7, eq. (3.9.17)]:

$$\begin{pmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{pmatrix}^{-1} = \begin{pmatrix}\mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\\ -(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} & (\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\end{pmatrix}. \quad (4.7)$$

Assign a left upper block of $\mathbf{M_B}$ matrix as $\mathbf{M}$ (i.e. $\mathbf{M} = \mathbf{B}_1\mathbf{\Sigma}\mathbf{B}_1^\top$), then we have $\mathbf{A} = \mathbf{M}$, $\mathbf{B} = \mathbf{B}_2$, $\mathbf{C} = \mathbf{B}_2^\top$, and $\mathbf{D} = \mathbf{0}$, so the relation (4.7) simplifies to

$$\begin{aligned}\begin{pmatrix}\mathbf{M} & \mathbf{B}_2\\ \mathbf{B}_2^\top & 0\end{pmatrix}^{-1} &= \begin{pmatrix}\mathbf{M}^{-1}+\mathbf{M}^{-1}\mathbf{B}_2(-\mathbf{B}_2^\top\mathbf{M}^{-1}\mathbf{B}_2)^{-1}\mathbf{B}_2^\top\mathbf{M}^{-1} & -\mathbf{M}^{-1}\mathbf{B}_2(-\mathbf{B}_2^\top\mathbf{M}^{-1}\mathbf{B}_2)^{-1}\\ -(-\mathbf{B}_2^\top\mathbf{M}^{-1}\mathbf{B}_2)^{-1}\mathbf{B}_2^\top\mathbf{M}^{-1} & (-\mathbf{B}_2^\top\mathbf{M}^{-1}\mathbf{B}_2)^{-1}\end{pmatrix}\\ &= \begin{pmatrix}\mathbf{M}^{-1}-\mathbf{M}^{-1}\mathbf{B}_2(\mathbf{B}_2^\top\mathbf{M}^{-1}\mathbf{B}_2)^{-1}\mathbf{B}_2^\top\mathbf{M}^{-1} & \mathbf{M}^{-1}\mathbf{B}_2(\mathbf{B}_2^\top\mathbf{M}^{-1}\mathbf{B}_2)^{-1}\\ (\mathbf{B}_2^\top\mathbf{M}^{-1}\mathbf{B}_2)^{-1}\mathbf{B}_2^\top\mathbf{M}^{-1} & -(\mathbf{B}_2^\top\mathbf{M}^{-1}\mathbf{B}_2)^{-1}\end{pmatrix}.\end{aligned}$$

A rather straightforward implementation of the algorithm computing the respective blocks of the inverse separately was suggested, which also provided a noticeable speed-up in the calculation. However, it still hit numerical issues caused by poor conditioning of some of the matrices appearing in the calculation (namely of the lower-right block) in some cases, like with polynomial regression of the tip contact area function 8. The key contribution to the high condition number was identified to be the matrix $\mathbf{B}_2$ (provided by derivatives of the objective function along the estimated parameters). The computation procedure has therefore been reworked in order to avoid product expressions involving multiple instances of $\mathbf{B}_2$, greatly benefiting from using singular value decomposition. This led to the final structure of the algorithm, which provides both speed and numeric stability, and is described below.

4.3.1 System matrix inverse

Let $\mathbf{\Sigma}$ be a covariance matrix of the input data written in a block form as in equation (4.1) and $\mathbf{B}_1 = (\mathbf{B}_{11}\ \mathbf{I})$, where $\mathbf{B}_{11}$ is a diagonal matrix of partial derivatives and $\mathbf{I}$ is identity matrix. Then

$$\begin{aligned}\mathbf{M} = \mathbf{B}_1\mathbf{\Sigma}\mathbf{B}_1^\top &= \begin{pmatrix}\mathbf{B}_{11} & \mathbf{I}\end{pmatrix}\begin{pmatrix}\mathbf{\Sigma_X} & \mathbf{\Sigma_{XY}}\\ \mathbf{\Sigma_{XY}}^\top & \mathbf{\Sigma_Y}\end{pmatrix}\begin{pmatrix}\mathbf{B}_{11}\\ \mathbf{I}\end{pmatrix} =\\ &= \left(\mathbf{B}_{11}\mathbf{\Sigma_X}\mathbf{B}_{11} + \mathbf{\Sigma_{XY}}^\top\mathbf{B}_{11} + \mathbf{B}_{11}\mathbf{\Sigma_{XY}} + \mathbf{\Sigma_Y}\right).\end{aligned}$$

As the matrix $\mathbf{B}_{11}$ is diagonal, given by a vector $\mathbf{b}_1$, we have an explicit formula for elements of $\mathbf{M}$:

$$\mathbf{M}_{ij} = \mathbf{b}_{1i}\mathbf{b}_{1j}\mathbf{\Sigma_X}_{ij} + \mathbf{b}_{1j}\mathbf{\Sigma_{XY}}_{ji} + \mathbf{b}_{1i}\mathbf{\Sigma_{XY}}_{ij} + \mathbf{\Sigma_Y}_{ij}.$$

Assume that $\mathbf{\Sigma}$ is symmetric positive definite; since $\mathbf{B}_1$ has full row rank, $\mathbf{M} = \mathbf{B}_1\mathbf{\Sigma}\mathbf{B}_1^\top$ is also symmetric positive definite matrix [9]. Moreover $\mathbf{M}$ is usually well-conditioned, so we can compute its Cholesky factorization $\mathbf{M} = \mathbf{L}_M\mathbf{L}_M^\top$. (This will also allow to employ triangular substitutions instead of multiplicating by matrix inverses at some places.)

Denote by $\mathbf{E}$ the solution of $\mathbf{L}_M\mathbf{X} = \mathbf{B}_2$. Then we have

$$\mathbf{E} = \mathbf{L}_M^{-1}\mathbf{B}_2 \quad \text{and} \quad \mathbf{E}^\top = \mathbf{B}_2^\top(\mathbf{L}_M^\top)^{-1}.$$

Since $\mathbf{M}^{-1} = (\mathbf{L}_M^\top)^{-1}\mathbf{L}_M^{-1}$, we obtain

$$\mathbf{M}^{-1}\mathbf{B}_2 = ((\mathbf{L}_M^\top)^{-1}\mathbf{L}_M^{-1})(\mathbf{L}_M\mathbf{E}) = (\mathbf{L}_M^\top)^{-1}\mathbf{E} \quad \text{and} \quad \mathbf{B}_2^\top\mathbf{M}^{-1} = \mathbf{E}^\top\mathbf{L}_M^{-1},$$

hence

$$\begin{pmatrix}\mathbf{M} & \mathbf{B}_2\\ \mathbf{B}_2^\top & \mathbf{0}\end{pmatrix}^{-1} = \begin{pmatrix}(\mathbf{L}_M^\top)^{-1}\mathbf{L}_M^{-1} - (\mathbf{L}_M^\top)^{-1}\mathbf{E}(\mathbf{E}^\top\mathbf{E})^{-1}\mathbf{E}^\top\mathbf{L}_M^{-1} & (\mathbf{L}_M^\top)^{-1}\mathbf{E}(\mathbf{E}^\top\mathbf{E})^{-1}\\ (\mathbf{E}^\top\mathbf{E})^{-1}\mathbf{E}^\top\mathbf{L}_M^{-1} & -(\mathbf{E}^\top\mathbf{E})^{-1}\end{pmatrix}.$$

Due to the contribution of $\mathbf{B}_2$, matrix $\mathbf{E}$ is often ill-conditioned, and we should therefore avoid inverting/factorizing matrices involving multiple products of $\mathbf{E}$, where the condition number grows with the power of $\mathbf{E}$.

Our suggestion is to calculate the (thin) singular value decomposition of $\mathbf{E} = \mathbf{U}_E\mathbf{S}_E\mathbf{V}_E^\top$ with

- $\mathbf{U}_E$: orthogonal columns, dimension (n, m), $\mathbf{U}_E^\top\mathbf{U}_E = \mathbf{I}_n$,
- $\mathbf{S}_E$: diagonal, dimension (m, m), and
- $\mathbf{V}_E^\top$: orthogonal, dimension (m, m),

assuming n is the number of data points, m is the number of parameters, $n \geq m$.
We then have:

$$\begin{aligned}
\mathbf{E}^\top\mathbf{E} &= (\mathbf{V}_E\mathbf{S}_E\mathbf{U}_E^\top)(\mathbf{U}_E\mathbf{S}_E\mathbf{V}_E^\top) = \mathbf{V}_E\mathbf{S}_E^2\mathbf{V}_E^\top,\\
(\mathbf{E}^\top\mathbf{E})^{-1} &= \mathbf{V}_E\mathbf{S}_E^{-2}\mathbf{V}_E^\top,\\
(\mathbf{E}^\top\mathbf{E})^{-1}\mathbf{E}^\top &= (\mathbf{V}_E\mathbf{S}_E^{-2}\mathbf{V}_E^\top)(\mathbf{V}_E\mathbf{S}_E\mathbf{U}_E^\top) = \mathbf{V}_E\mathbf{S}_E^{-1}\mathbf{U}_E^\top,\\
\mathbf{E}(\mathbf{E}^\top\mathbf{E})^{-1} &= (\mathbf{U}_E\mathbf{S}_E\mathbf{V}_E^\top)(\mathbf{V}_E\mathbf{S}_E^{-2}\mathbf{V}_E^\top) = \mathbf{U}_E\mathbf{S}_E^{-1}\mathbf{V}_E^\top,\\
\mathbf{E}(\mathbf{E}^\top\mathbf{E})^{-1}\mathbf{E}^\top &= (\mathbf{U}_E\mathbf{S}_E^{-1}\mathbf{V}_E^\top)(\mathbf{V}_E\mathbf{S}_E\mathbf{U}_E^\top) = \mathbf{U}_E\mathbf{U}_E^\top.
\end{aligned}$$

Substituting back into the inverse formula, we obtain

$$\begin{aligned}
\begin{pmatrix}\mathbf{M} & \mathbf{B}_2\\ \mathbf{B}_2^\top & \mathbf{0}\end{pmatrix}^{-1} &= \begin{pmatrix}(\mathbf{L}_M^\top)^{-1}\mathbf{L}_M^{-1} - (\mathbf{L}_M^\top)^{-1}(\mathbf{U}_E\mathbf{U}_E^\top)\mathbf{L}_M^{-1} & (\mathbf{L}_M^\top)^{-1}(\mathbf{U}_E\mathbf{S}_E^{-1}\mathbf{V}_E^\top)\\ (\mathbf{V}_E\mathbf{S}_E^{-1}\mathbf{U}_E^\top)\mathbf{L}_M^{-1} & -(\mathbf{V}_E\mathbf{S}_E^{-2}\mathbf{V}_E^\top)\end{pmatrix}\\
&= \begin{pmatrix}(\mathbf{L}_M^\top)^{-1}\mathbf{L}_M^{-1} - ((\mathbf{L}_M^\top)^{-1}\mathbf{U}_E)(\mathbf{U}_E^\top\mathbf{L}_M^{-1}) & ((\mathbf{L}_M^\top)^{-1}\mathbf{U}_E)(\mathbf{S}_E^{-1}\mathbf{V}_E^\top)\\ (\mathbf{V}_E\mathbf{S}_E^{-1})(\mathbf{U}_E^\top\mathbf{L}_M^{-1}) & -(\mathbf{V}_E\mathbf{S}_E^{-1})(\mathbf{S}_E^{-1}\mathbf{V}_E^\top)\end{pmatrix}.
\end{aligned}$$

Denote $\mathbf{F} = \mathbf{V}_E\mathbf{S}_E^{-1}$, obtained by dividing columns of $\mathbf{V}_E$ by the singular values, and $\mathbf{G} = (\mathbf{L}_M^\top)^{-1}\mathbf{U}_E$, calculated by solving the triangular system $\mathbf{L}_M^\top\mathbf{G} = \mathbf{U}_E$. Then we get

$$\begin{pmatrix}\mathbf{M} & \mathbf{B}_2\\ \mathbf{B}_2^\top & \mathbf{0}\end{pmatrix}^{-1} = \begin{pmatrix}(\mathbf{L}_M^\top)^{-1}\mathbf{L}_M^{-1} - \mathbf{G}\mathbf{G}^\top & \mathbf{G}\mathbf{F}^\top\\ \mathbf{F}\mathbf{G}^\top & -\mathbf{F}\mathbf{F}^\top\end{pmatrix}.$$

Procedure:

1. Calculate Cholesky decomposition of $\mathbf{M} = \mathbf{L}_M\mathbf{L}_M^\top$.
2. Calculate $\mathbf{E}$ by solving $\mathbf{L}_M\mathbf{E} = \mathbf{B}_2$ (use a triangular solve procedure if available[3]).
3. Calculate a thin SVD[4] of $\mathbf{E} = \mathbf{U}_E\mathbf{S}_E\mathbf{V}_E^\top$.
4. Calculate $\mathbf{F} = \mathbf{V}_E\mathbf{S}_E^{-1}$ (divide columns of $\mathbf{V}_E$ by respective singular values σ_i).
5. Calculate $\mathbf{G}$ by solving $\mathbf{L}_M^\top\mathbf{G} = \mathbf{U}_E$ (use a triangular transpose solve procedure if available[5]).
6. Calculate $\mathbf{Q}_{21} = \mathbf{F}\mathbf{G}^\top$.
7. Calculate $\mathbf{Q}_{11} = (\mathbf{L}_M^\top)^{-1}\mathbf{L}_M^{-1} - \mathbf{G}\mathbf{G}^\top$ (use procedures for calculating symmetric positive matrix inverse from a Cholesky factor[6]).
8. $\mathbf{Q}_{22} = \mathbf{F}\mathbf{F}^\top$.

[3] LAPACK: `dtrsm("L", "L", "N", "N", ...)`, R : `forwardsolve`

[4] LAPACK: `dgesvd`/`dgesdd`; if higher stability is desired, use more sophisticated procedures such as `dgesvdq` or `dgejsv` (only in newer LAPACK; check version using `ilaver`), R : `svd`

[5] LAPACK: `dtrsm("L", "L", "T", "N", ...)`, R : `backsolve`

[6] LAPACK: `dpotri`, R : `chol2inv`

4.4 R package

R is a free software environment for statistical computing and graphics. Due to the large number of packages dedicated to various problems from different fields, it is used by a wide (not only) statisticians community (for more details about R see [29]). Because of the wide usability of the generalized form of the algorithm in metrology (especially in nanoindentation) and in other applications, the package `OEFPIL` was developed for use in R environment. The package is available on CRAN [37] and can be installed using `install.packages("OEFPIL")`.

The main function named `OEFPIL` allows obtaining estimates of parameters of any (twice continuously differentiable) nonlinear function together with their covariance matrix. The function uses the following arguments:

`data`	an input data file,
`form`	a symbolic description of a model,
`start.val`	a list of initial values of estimating parameters,
`CM`	an input covariance matrix of data,
`max.iter`	maximum number of iterations,
`see.iter.val`	a logical argument indicating if results in every step of iteration should be displayed and saved,
`save.file.name`	a name of the file for exporting results,
`th`	a value indicating threshold necessary for the iteration stoppage,
`signif.level`	a significance level for the confidence interval computing,
`useNLS`	a logical argument indicating if pre-calculation of initial values of parameters via `nlsLM()` is required.

Because of linearization of nonlinear function via Taylor series assumes initial values of parameters close enough to the real values (see [18], VI.2), using `useNLS = TRUE` is highly recommended to avoid non-convergence of algorithm or biased results.

The output of this function is an object of class `'OEFPIL'`, which is a list containing estimated values of parameters, the estimated covariance matrix, total number of iterations and other components used for further calculations or graphical functions. Other details about specification of arguments or output format are in the `OEFPIL` documentation (in R just write `?OEFPIL`).

Basic generics functions in variation for `'OEFPIL'` objects are also part of the package. This includes, for example, `summary` for displaying results, `coef` and `vcov` for extracting parameters and covariance matrix of the estimated function, respectively. Custom functions for computing confidence intervals for parameters (`confInt.OEFPIL`) and confidence and prediction pointwise bands for estimated function (`confBands.OEFPIL`) was implemented. Both of them allow computing multiple confidence intervals at once by setting the vector of significance levels for intervals through the argument `signif.level`. The confidence and prediction intervals are computed under the normality assumption (see subsection 4.2.3).

The `steam` data from `MASS` library was chosen for basic illustration of functions from `OEFPIL` package. The data frame contains 14 temperature (`Temp`) and pressure (`Press`) measurements in a saturated steam driven experimental device. The diagonal input covariance matrix with the same variance for x and y variable was used in the example (this corresponds to the setting $D_i = 1$ in [4], where the same data set is used).

```
## Example use of OEFPIL() function for steam data

library(OEFPIL)
library(MASS)

## defining initial values of parameters
startsteam <- list(b1 = 5, b2 = 8, b3 = 290)
```

```
## creating covariance matrix
k <- nrow(steam)
CM <- diag(1, 2 * k, 2 * k)

EstFun <- OEFPIL(steam, Press ~ b1 * 10 ^ (b2 * Temp / (b3 + Temp)),
                 start.val = startsteam, CM = CM, useNLS = TRUE)

## displaying results via summary function
summary(EstFun)
Summary of the result:

Press ~ b1 * 10^(b2 * Temp/(b3 + Temp))

    Param Est          Std Dev   CI Bound 2.5 %   CI Bound 97.5 %
b1   4.487870       0.4828491         3.541503          5.434237
b2   7.188155       0.5900662         6.031646          8.344663
b3 221.837783      31.6081218       159.887003        283.788564

 Estimated covariance matrix:
           b1         b2        b3
b1  0.2331432  0.2296195  13.43054
b2  0.2296195  0.3481782  18.46313
b3 13.4305405 18.4631318 999.07337

 Number of iterations: 8
```

The summary contains basic results for the estimated parameters. There is a symbolic description of the used model on the first line and two tables. In the first table are estimation of parameters with their standard deviation (i.e. uncertainty) in `Param Est` and `Std Dev` column, respectively. Other columns contain lower and upper bounds of the confidence interval for a selected level of significance (if the `signif.level` argument is missing, the default value 0.05 is used). The second summary table contains the estimated covariance matrix for parameters. The total number of iterations is displayed on the last line of the summary output.

In the steam data example, the estimated values of parameters $\hat{b}_1 = 4.488, \hat{b}_2 = 7.188, \hat{b}_3 = 221.838$ are in accordance with values of $\hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3$ from Example 2.3 in [4]. The function `ortresiduals.OEFPIL` for computing orthogonal sum of squares (e.g. sum of the squared shortest Euclidean distances between data points and the estimated function) is also available in the package. This function allows individual choice of length of minimization interval via `min.c` argument (too small value leads to narrow interval for minimization and misleading results). In addition to the total orthogonal sum of squares (`SSort`), there are values of orthogonal residuals (`o.resid`) and x coordinates of points, where the minimal distance is realized (`x.ores`), in the output. The resulting orthogonal sum of squares for steam data example is 15.26281, again in accordance with [4].

```
## Example output of ortresiduals.OEFPIL() function for steam data

ortresiduals.OEFPIL(EstFun)
Argument 'min.c' was not defined. Value 5.25 was used for the
   calculation.
$x.ores
 [1] -0.1041182   9.7131854  19.3427929  29.9771897  42.4031270
     50.9102407  59.8553762 68.7104743  78.6414442  84.0272203
     89.4955942  96.3598298 100.5014133 105.1662985

$o.resid
 [1] 0.32998264 0.54728391 0.90021218 0.02586705 2.50727234 0.92927799
```

```
    0.14610776 1.29625685 1.36208906 0.97453696 0.50506184 1.36096635
    0.50174256 0.16638001

$SSort
[1] 15.26281
```

The package offers three functions described below to display the results from `OEFPIL` in graphical form. Two of them are created using the `ggplot2` library (for more details, see [33] or [34]), allowing users to add other components into the graph or modify the output in the usual way. For illustration of using graphical functions, the input covariance matrix of steam data was modified to see the confidence bands better.

`plot.OEFPIL` - a basic plot of estimated function, original data points and confidence or prediction pointwise bands of the resulting curve (the choice of interval type is possible through `interval` argument). The function allows adding the same arguments as base `plot` function.

```
## Example of using plot function on 'OEFPIL' object

## new covariance matrix for steam data
CM2 <- diag(c(rep(12,n), rep(14,n)))
EstF2 <- OEFPIL(steam, Press ~ b1 * 10^(b2 *Temp/(b3 + Temp)),
   start.val = startsteam, CM = CM2, useNLS = TRUE)

## to plot confidence interval set interval = "conf"
plot(EstF2, signif.level = 0.05, interval = "conf")
```

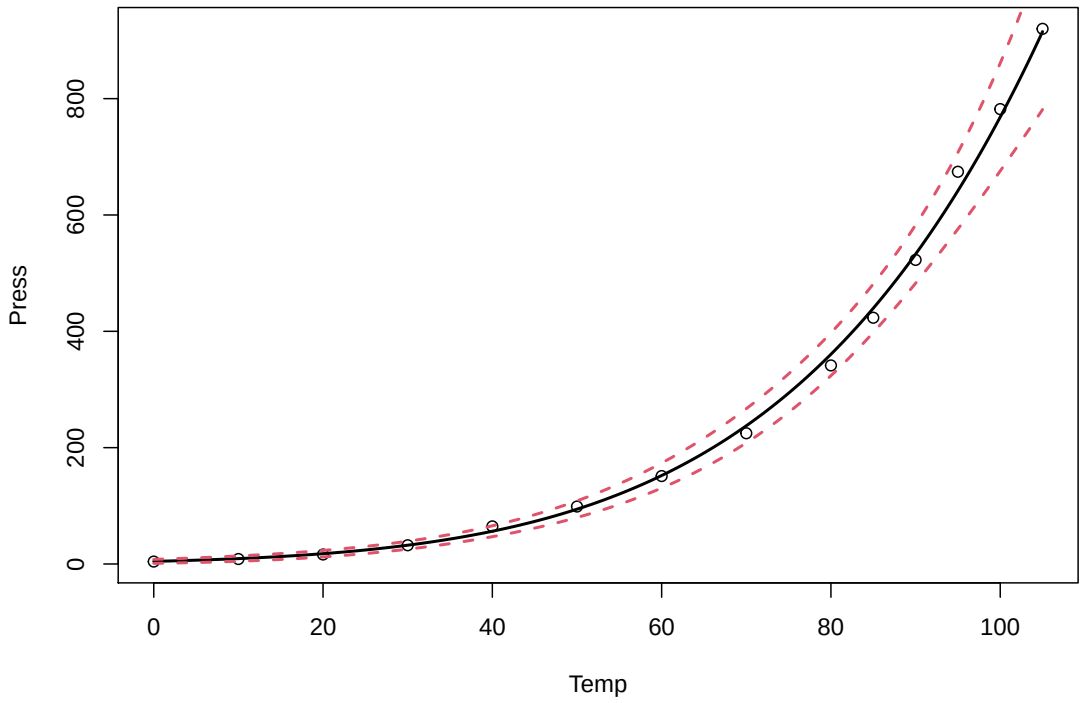


Figure 4.1: Example output of basic plot for an OEFPIL object – Data points with fitted curve (black) and 95% pointwise confidence band (red).

`curvplot.OEFPIL` - a graph of estimated function including original data points, with optional adding confidence pointwise bands of the resulting curve. The function allows to display more confidence intervals with different significance levels in one plot (Figure 4.2).

`paramplot.OEFPIL` - a graph of the estimated values of the parameters with error bars (i.e. $\pm$ standard deviation). This function is a useful tool for comparing results from different models (e.g. different degrees of polynomials used for estimation or different input uncertainties

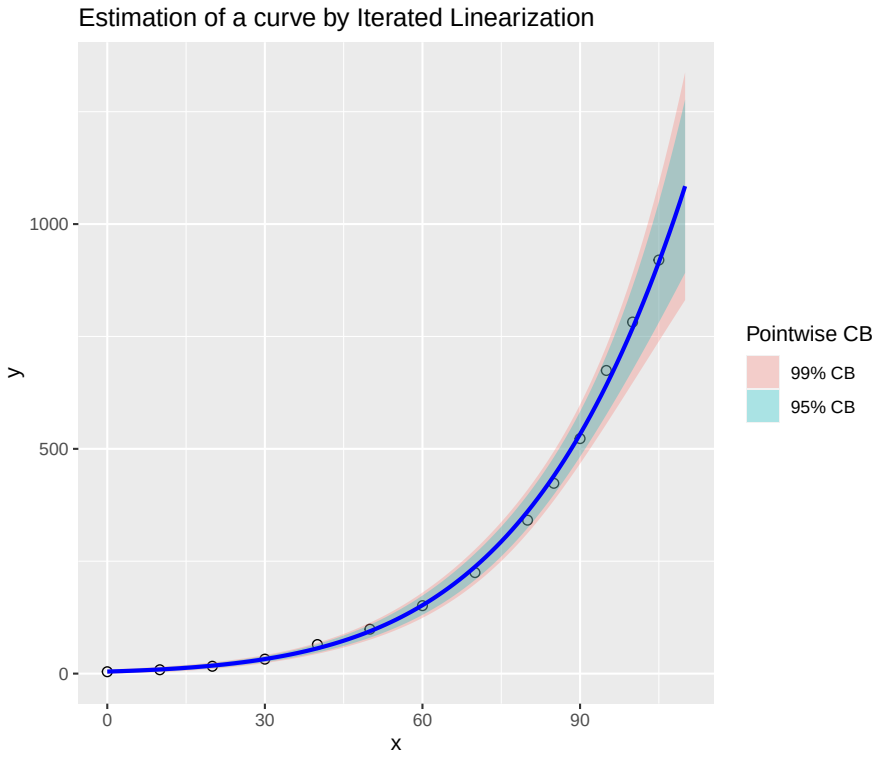


Figure 4.2: Example output of `curvplot.OEFPIL` function – Estimated curve (blue line) with two different confidence bands.

of variables). Due to possible large differences in estimated parameters units, the windows for the individual parameter estimation do not have the same scale. In the illustration figure 4.3 we compare the previous two models for steam data, where the difference is only in the input covariance matrix. The graph shows that estimated values of parameters are in both cases similar, but their uncertainty significantly differs.

```
paramplot.OEFPIL(EstFun, EstF2)
## EstF2 is model with higher input variance of variables and
## is assigned as model 2 in the figure
```

The special version of `OEFPIL` function adapted for using in nanoindentation for unloading curve fitting was implemented in the package as `NanoIndent.OEFPIL`. The indentation depth h and load F are considered as x and y variable, respectively. The `form` argument of this function is predefined as power law function $F = \alpha(h - h_p)^m$ (according to equation 2.1). The initial values of parameters m, α, h_p are now optional argument, if missing, the relationships used for computation are

$$\begin{aligned} m^{(0)} &= 1.5, \\ h_p^{(0)} &= 0.9\, h_{min}, \\ \alpha^{(0)} &= \frac{F_{max}}{(h_{max} - h_p^{(0)})^{m^{(0)}}}, \end{aligned}$$

where h_{min} is minimum of depth measured at unloading curve.

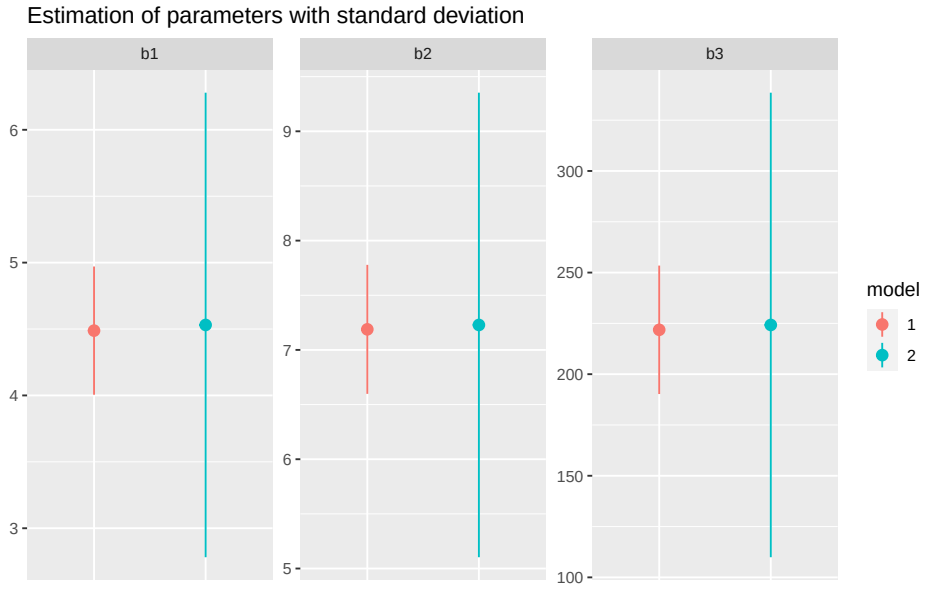


Figure 4.3: Example output of `paramplot.OEFPIL` function (two models with different input uncertainties).

There are some other input arguments:

`lcut`, `ucut`	allows to choose range of the unloading curve used for fitting,
`unload.data`	a logical value indicating, if only unloading part of the curve is present in `data`,
`uh`, `uF`	standard deviation of depth and load (it is not necessary to set whole covariance matrix)

The argument settings for fitting unloading curve using $20 - 98\%$ its range with diagonal covariance matrix (with 0.5 uncertainty for depth and 0.001 for load), using data containing only unloading part of the curve, shows following example:

```
## Example use of NanoIndent.OEFPIL() function
## silicaBerk data are example data implemented in the package

output.NI <- NanoIndent.OEFPIL(silicaBerk, unload.data = TRUE,
   ucut = 0.98, lcut = 0.2, uh = 0.5, uF = 0.001, signif.level = 0.05,
   useNLS = TRUE)

## display results via summary()
summary(output.NI)
Summary of the result:

f ~ alpha * (h - hp)^m

        Param Est          Std Dev   CI Bound 2.5 %   CI Bound 97.5 %
alpha  0.00671548    0.0008994143       0.00495266         0.0084783
m      1.22203004    0.0319298976       1.15944859         1.2846115
hp    31.18649493    0.6092014305      29.99248207        32.3805078

 Estimated covariance matrix:
              alpha             m            hp
alpha  8.089462e-07 -2.869859e-05  0.0005422115
m     -2.869859e-05  1.019518e-03 -0.0191398285
```

```
hp      5.422115e-04 -1.913983e-02  0.3711263829

 Number of iterations: 6
```

The summary output has the same structure as for `OEFPIL` function. In this case, resulting estimation of parameters is $\hat{\alpha} = 0.0067$, $\hat{m} = 1.22$, $\hat{h_p} = 31.18$ obtained in 6 iteration steps.

For more examples and details about functions, see the package documentation [37].

4.5 C library

After successful validation of algorithm performance on indentation unloading curves, the implementation of the general algorithm for arbitrary functions in C language was sterted. It was created as a separate project, intended for eventual inclusion in Niget software when ready.

For most vector and matrix calculations, the de facto standard interfaces for calculations in linear algebra have been employed: BLAS (Basic Linear Algebra Subprograms, low-level operations, `http://www.netlib.org/blas/`), and LAPACK (Linear Algebra PACKage, higher-level routines, `http://www.netlib.org/lapack/`, see also [1]). BLAS and LAPACK provide specialized routines (e.g. for symmetric or triangular matrices) that allow efficient implementation of the algorithm.

For convenient interoperation with BLAS and LAPACK routines, matrices are represented as 1-dimensional arrays in column-major order. That is, the entry in the i-th row and j-th column of a matrix $\mathbf{A}$ of m rows can be accessed as `A[j*m + i]`.

The library provides a single main function:

```
void oefpil(void *fcn, void *data,
            int np, double *p, double *pcov,
            int n, const double *x, const double *y, const double *xycov,
            int maxit, double tol, int printlevel, FILE *rptfile,
            int *info, int *niter)
}
```

with its arguments given as follows:

`void *fcn`	function that evaluates function values and derivatives of the fitted function (see below)
`void *data`	pointer to optional data passed to `fcn`
`int np`	number of function parameters
`double *p`	dimension np; on input: starting parameter values; on output: estimated parameter values
`double *pcov`	dimension $(np)^2$; on output: covariance matrix of estimated parameters
`int n`	number of points
`const double *x`	dimension n; x-values of the points
`const double *y`	dimension n; y-values of the points
`const double *xycov`	dimension $(2n)^2$; covariance matrix of the input values
`double tol`	relative tolerance for convergence criterion
`int printlevel`	level of verbosity; 0: no output, 1: basic iteration progress, 2: iteration progress including parametr covariance, 3: debug output
`FILE *rptfile`	file where output should be sent (e.g. `stdout`); can be NULL (no output)
`int *info`	
`int *niter`	

The evaluation function `fcn` is then assumed in the following form:

```
void fcn(void *data, int n, int np, const double *x, const double *beta,
         double *wf, double *wfx, double *wfp)
```

with the arguments:

`void *data`	void pointer to optional data passed to `fcn`
`int n`	number of points
`int np`	number of function parameters
`const double *x`	dimension n; x-values of the points
`const double *p`	dimension np; values of the parameters
`double *wf`	dimension n; on output: function values in the points
`double *wfx`	dimension n; on output: derivatives of the function along x in the points
`double *wfp`	dimension $n \cdot np$; on output: derivatives of the function along the parameters

Please note that the function signatures may change in future versions with future development.

The library uses CMake software collection (`https://cmake.org`) for building, which also simplifies its integration in other software.

Since the library has developed into an actually usable software, it has been decided to keep it as a standalone project, freely available at `https://gitlab.com/cmi6014/oefpil/` under MIT license. Compared to the R package, which aims for ease of use and availability for the wide community of R users, the C library can provide greater computation performance and is available to software developers for inclusion in their software.

4.6 Validation of algorithm performance

In this section we demonstrate usability and performance of OEFPIL on several examples.

4.6.1 NIST reference data for nonlinear regression

At `https://www.itl.nist.gov/div898/strd/nls/nls_main.shtml`, NIST has published a collection of datasets as well as reference values used for evaluating software for calculating nonlinear regression. Even when the formulation of the problem to be solved by OEFPIL is different, so the results are not fully comparable with the reference values, these model problems can be used to demonstrate OEFPIL's ability to reach a solution for different combinations of data and nonlinear models. In this case, a diagonal covariance matrix (with different values σ_x^2 and σ_y^2 for particular datasets) was used for OEFPIL. For the majority of NIST datasets the algortihm showed fast convergence to the values close to the reference (selected examples are presented in following paragraphs). It shows usability of OEFPIL for a wide range of nonlinear functions including exponentials, rationals or trigonometric functions. With changing uncertainties σ_x and σ_y the estimated parameters often was the same, only the estimated variance increased with the input uncertainties.

Misra1d data The phenomenon of increasing uncertainties of parameter estimates is illustrated by Figure 4.4, where data representing volume (y) and pressure (x) from NIST study regarding dental research in monomolecular adsorption (*misra1d*) are presented. The relationship between x and y is

$$y = \frac{\beta_1 \beta_2 x}{1 + \beta_2 x}$$

and we consider diagonal covariance matrix with following uncertainties:

- model 1: $\sigma_x = 0.1,\ \sigma_y = 0.2$,

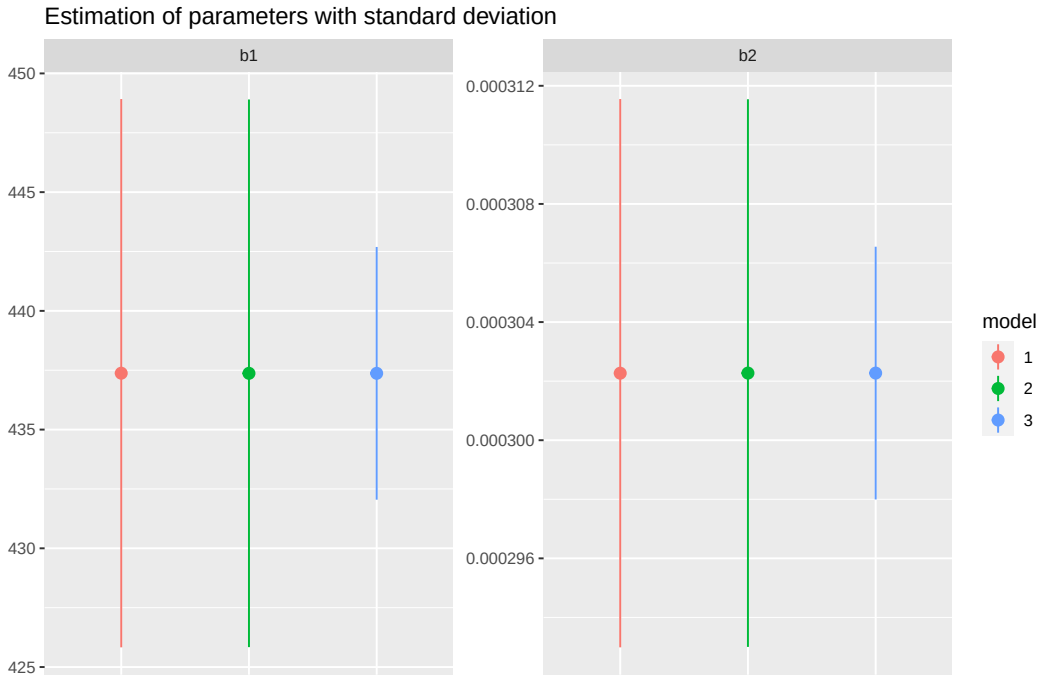


Figure 4.4: Comparison of models with different input uncertainties for misra1d NIST data (`paramplot.OEFPIL` output).

- model 2: $\sigma_x = 0.01,\ \sigma_y = 0.2,$
- model 3: $\sigma_x = 0.001,\ \sigma_y = 0.1.$

For all these models the estimation of parameters β_1 and β_2 is essentially the same (see Table 4.1), but its uncertainty growths noticeably with uncertainties of input variables.

Table 4.1: Estimates of parameters with different models for Misra1d data.

	Model 1	Model 2	Model 3	NIST certified value
β_1	437.3748	437.3698	437.3697	437.36970754
β_2	3.022691e-04	3.022732e-04	3.022732e-04	3.0227324449e-04

Chwirut2 data This data contains the result of a NIST study involving ultrasonic calibration. The ultrasonic response y and the metal distance x are estimated by

$$y = \frac{\exp(-\beta_1 x)}{(\beta_2 + \beta_3 x)}.$$

By setting $\sigma_x = 0.05$ and $\sigma_y = 0.5$ the function estimated by OEFPIL differs in some areas from NLS fit (see Figure 4.5) due to the use of other minimization criterion (as mentioned in introduction of this chapter). Despite this differences, in most of the cases considered, both methods give similar results and with decreasing input uncertainty the OEFPIL curve approaches to the NLS curve.

Roszman1 data These data come from a NIST study involving quantum defects in iodine atoms and are represented by number of quantum defects y and the excited energy state x. The

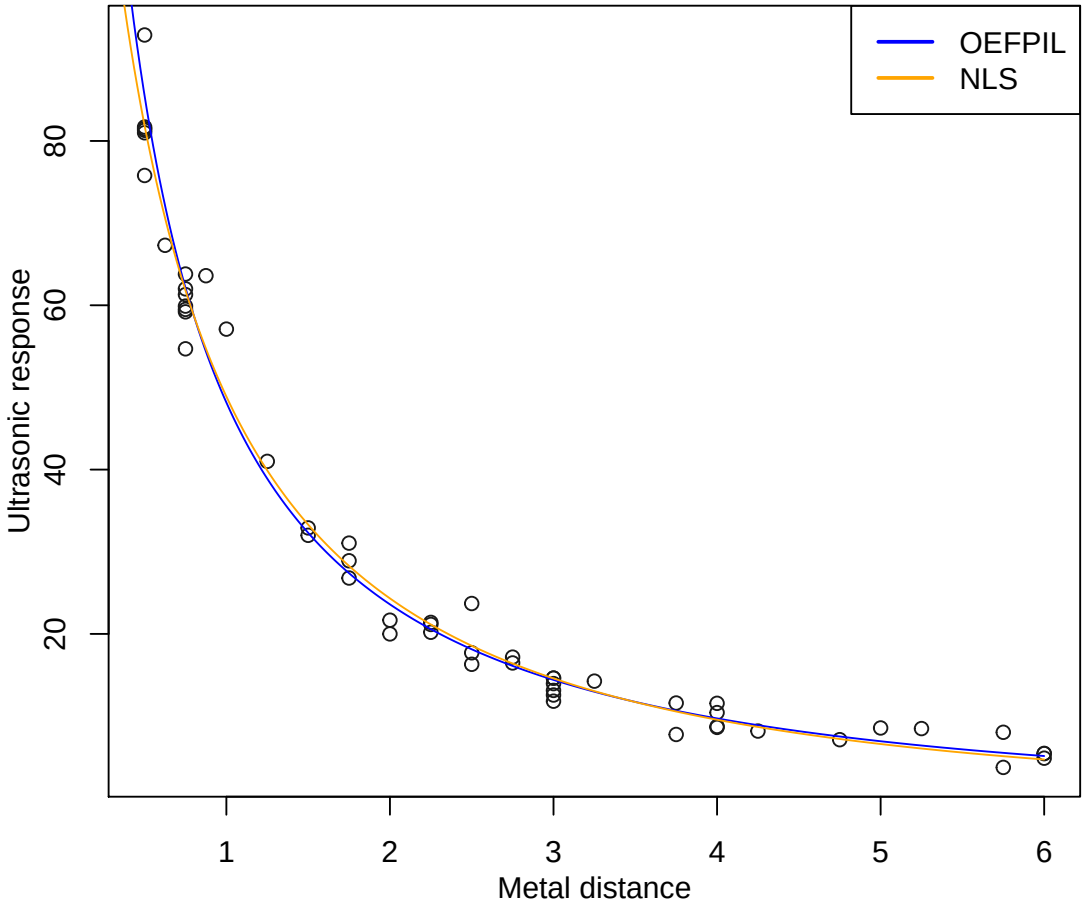


Figure 4.5: Fitting NIST Chwirut2 data using NLS (orange line) and OEFPIL with $\sigma_x = 0.05$ and $\sigma_y = 0.5$ (blue line).

relationship between these two variables is considered as

$$\beta_1 - \beta_2 x - \frac{\arctan\left(\frac{\beta_3}{x-\beta_4}\right)}{\pi}.$$

As shown in Table 4.2 the OEFPIL parameter estimates (with input uncertainties $\sigma_x = 5$ and $\sigma_y = 0.005$) are very close to the NLS estimates and the fitted function is essentially the same (the curves for these two methods completely overlap in Figure 4.6).

Table 4.2: Estimates of parameters for Roszman data.

	OEFPIL	NIST certified value
β_1	2.016828e-01	2.0196866396e-01
β_2	-6.150549e-06	-6.1953516256e-06
β_3	1.205522e+03	1.2044556708e+03
β_4	-1.820444e+02	-1.8134269537e+02

4.6.2 Examples for INRIM WTLS/CCC software

In [22] and [21] several examples were presented that included data with nontrivial covariance matrices.

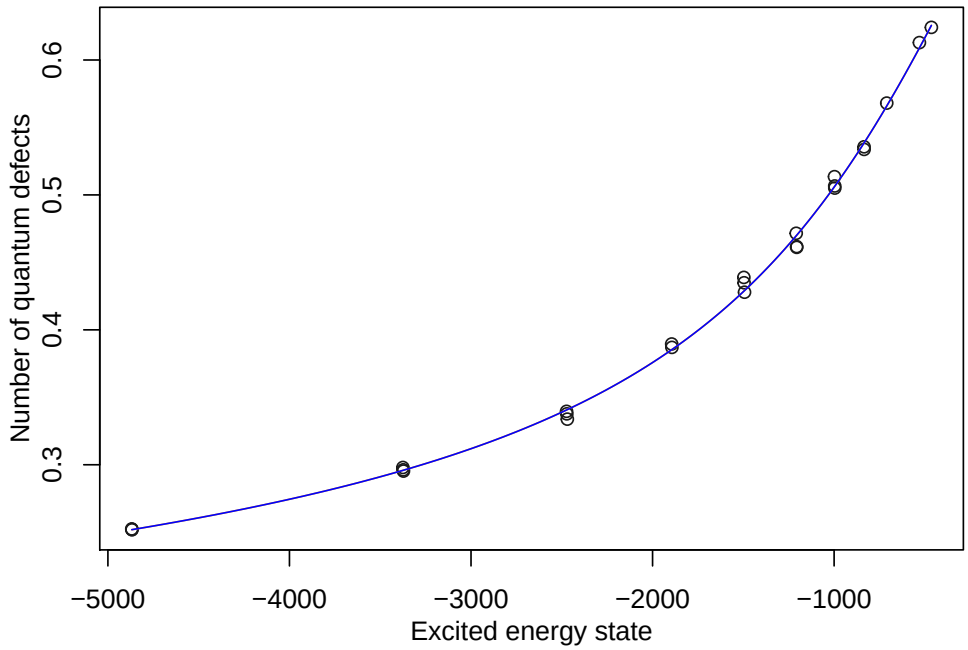


Figure 4.6: Example of Roszman1 data fitting using $\sigma_x = 5$ and $\sigma_y = 0.005$ (the blue line denotes OEFPIL as well as NLS fit).

Gas chromatometer calibration An example of gas chromatometer calibration in [22] involved data with (a) diagonal covariance matrix containing only variance of variables on the diagonal and (b) covariance matrix with correlations present between some of the y values, to be fitted by a second-order polynomial $ax^2 + bx + c$. Tables 4.3 and 4.4 show a comparison between the WTLS software and OEFPIL. Both the estimated parameters and their uncertainties and covariances show a very good agreement in both cases. The convergence was achieved in 4 and 5 iteration steps, respectively (with preprocessing inital values using NLS method).

Table 4.3: Gas chromatometer calibration – comparison of reference values (no correlation assumed), WTLS, and OEFPIL for diagonal covariance matrix.

	ISO 6413	WTLS	OEFPIL
a	-4.1096×10^{-13}	-4.0373×10^{-13}	-4.0865×10^{-13}
b	2.4403×10^{-5}	2.4400×10^{-5}	2.4401×10^{-5}
c	-1.4037×10^{-4}	-1.2895×10^{-4}	-1.3110×10^{-4}
u_a	1.895×10^{-13}	1.895×10^{-13}	1.8951×10^{-13}
u_b	5.901×10^{-8}	5.901×10^{-8}	5.9003×10^{-8}
u_c	1.175×10^{-3}	1.175×10^{-3}	1.1748×10^{-3}
$u_{a,b}$	-1.020×10^{-20}	-1.020×10^{-20}	-1.0202×10^{-20}
$u_{a,c}$	4.667×10^{-17}	4.667×10^{-17}	4.6692×10^{-17}
$u_{b,c}$	-2.057×10^{-11}	-2.057×10^{-11}	-2.0578×10^{-11}

Flow meter calibration In [21] an example of flow meter calibration was presented to demonstrate improvements in the WTLS algorithm, with a number of correlations in the x data as well as uncertainties in the y data. As shown Table 4.5, OEFPIL's results are in a very good match with the newer CCC software. Also in this case the OEPFIL converges within 5 iteration steps (using NLS preprocessing of initial values).

Table 4.4: Gas chromatometer calibration – comparison of reference values (no correlation assumed), WTLS, and OEFPIL for covariance matrix with correlations in y values.

	ISO 6413	WTLS	OEFPIL
a	-4.1096×10^{-13}	-4.2247×10^{-13}	-4.2273×10^{-13}
b	2.4403×10^{-5}	2.4403×10^{-5}	2.4403×10^{-5}
c	-1.4037×10^{-4}	-1.3538×10^{-4}	-1.3569×10^{-4}
u_a	1.895×10^{-13}	1.804×10^{-13}	1.804×10^{-13}
u_b	5.901×10^{-8}	5.644×10^{-8}	5.643×10^{-8}
u_c	1.175×10^{-3}	1.174×10^{-3}	1.174×10^{-3}
$u_{a,b}$	-1.020×10^{-20}	-9.106×10^{-21}	-9.104×10^{-21}
$u_{a,c}$	4.667×10^{-17}	4.304×10^{-17}	4.306×10^{-17}
$u_{b,c}$	-2.057×10^{-11}	-1.961×10^{-11}	-1.962×10^{-11}

Table 4.5: Flow meter calibration – comparison of CCC 1.3, CCC 2.0, and OEFPIL.

	CCC 1.3	CCC 2.0	OEFPIL
a	$-13.950\,485\,4$	$-14.474\,789\,9$	$-14.477\,072\,0$
b	$-9.989\,667\,8$	$-9.589\,981\,1$	$-9.588\,479\,8$
c	$4.874\,164\,9$	$4.795\,202\,7$	$4.794\,964\,4$
d	$-0.196\,308\,5$	$-0.192\,837\,0$	$-0.192\,827\,0$
e	$0.001\,566\,5$	$0.001\,526\,9$	$0.001\,526\,8$
u_a	$1.758\,23$	$1.758\,48$	$1.758\,62$
u_b	$1.238\,67$	$1.238\,77$	$1.238\,73$
u_c	$2.385\,04 \times 10^{-1}$	$2.385\,13 \times 10^{-1}$	$2.385\,03 \times 10^{-1}$
u_d	$1.106\,79 \times 10^{-2}$	$1.106\,82 \times 10^{-2}$	$1.106\,79 \times 10^{-2}$
u_e	$1.340\,73 \times 10^{-4}$	$1.340\,76 \times 10^{-4}$	$1.340\,74 \times 10^{-4}$

Chapter 5

Effective uncertainty propagation

It has long been known that the common "linear" uncertainty propagation, based on the Taylor expansion of the investigated function, truncated as described e.g. in [12] does not produce reliable results when applied to nonlinear functions. Due to the nonlinearity, the mean of the function value no longer needs to equal the function evaluated at the input means (because it is affected by the variance of the input), and the estimate of the output variance might also be distorted.

An option to overcome the issues related to the linear uncertainty propagation is provided by using a Monte Carlo approach (see e.g. [11]). However, Monte Carlo in a naïve implementation may become very computationally demanding with increasing number of input variables.

Various approaches to mitigate deficiencies of the above methods have been investigated for a long time. Here we present some of the methods that might be applied in our present calculation setting.

5.1 Higher-order uncertainty propagation

An option to increase quality of estimates obtained by linear uncertainty propagation is to use a greater part of the Taylor expansion of the function of interest, see e.g. [19] or [23]. This approach involves using higher-order derivatives of the function as well as higher moments of the distributions of the input quantities, depending on the specific setting. While the calculation proceeds in a form of a closed-form expression, the maximum order of the expansion is limited as the number of terms grows rapidly. Nevertheless, already second-order expansion can provide a significant improvement in terms of both mean and (co)variance estimate.

5.1.1 SOERP

The paper [6] presented a computation procedure for uncertainty propagation through a function of multiple variables when better information on the distribution of the (independent) input quantities is available. As input, it requires:

- value of the function evaluated at the means of the input parameters,
- values of the first- and second-order (this giving the name to the procedure) derivatives of the function evaluated at the means of the input parameters,
- higher moments (up to 8) of probability distributions of the input parameters.

As output it produces the first 4 moments (mean, variance, skewness, kurtosis) of the distribution of the result.

A computer program called `SOERP` (Second Order Error Propagation) was also presented in this paper. The original FORTRAN code is not available, but the computation procedure

presented in the paper has been implemented in a Python module `SOERP` (`https://github.com/tisimst/soerp`). This module also provides additional functions for convenient use of the SOERP method.

5.1.2 R Propagate

In order to utilize the power of SOERP, knowledge of higher moments of the input distributions is required. It is quite common that the information on the input quantities is limited to basic characteristics of the (multivariate) normal distribution – the mean and the (co)variance. This case is treated by the `propagate` package for R (`https://cran.r-project.org/web/packages/propagate/index.html`). The function `propagate` allows calculating error propagation of multiple normal inputs through a scalar function using three methods. Assume that $f\colon \mathbb{R}^n \to \mathbb{R}$ is a function with the Jacobian $\mathbf{J}$ and Hessian $\mathbf{H}$, $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ with covariance matrix $\mathbf{\Sigma}$, and $y = f(\mathbf{x})$:

1. first-order (linear) error propagation:

$$E[y] = f(\mathbf{x}), \tag{5.1}$$

$$\sigma_y^2 = \mathbf{J}\mathbf{\Sigma}_\mathbf{x}\mathbf{J}^\top, \tag{5.2}$$

2. second-order error propagation:

$$E[y] = f(\mathbf{x}) + \frac{1}{2}\operatorname{tr}(\mathbf{H}\mathbf{\Sigma}), \tag{5.3}$$

$$\sigma_y^2 = \mathbf{J}\mathbf{\Sigma}\mathbf{J}^\top + \frac{1}{2}\operatorname{tr}(\mathbf{H}\mathbf{\Sigma}\mathbf{H}\mathbf{\Sigma}), \tag{5.4}$$

3. a Monte Carlo method.

As noted in the package documentation [31], its second-order propagation method yields exactly the same results (estimates of mean and variance) as `SOERP` for Python. Our tests also demonstrated that the results of the second-order propagation are also very close to the results of Monte Carlo.

As can be verified when experimenting with the package, this form of second-order propagation provides visible improvement over the standard first-order propagation while being quite straightforward to implement and very fast to compute using just a few matrix operations.

5.2 Latin Hypercube Sampling

Latin Hypercube Sampling (LHS) [25] is an alternative to the more common Monte Carlo (MC) method for simulations. In standard MC, for each input variable a set of N random values is selected independently. In LHS, the range of the input distribution is split into intervals of equal probability and one sample is chosen from each interval. This leads to lower sampling errors for a given N compared to MC. The sampling procedure can be extended to higher-dimensional data, with "Latin hypercube" standing for a generalization of the concept of Latin square[1].

The first software implementation of LHS was written in late 1970s in FORTRAN 77, and has been repeatedly improved and extended. This LHS software features a collection of functions for generating univariate data with a number of distributions that are frequently encountered, and allows to combine them in multivariate distributions based on prescribed correlations between individual variables.

The current `LHS` code, written in Fortran 90 and distributed under LGPL version 2.1 license, is presently included as a part of the `Dakota` software toolkit for optimization and uncertainty

[1] `https://en.wikipedia.org/wiki/Latin_square`

quantification (`https://dakota.sandia.gov/`). It can be used as a standalone command-line program commanded by a keyword file, or as a software library when calling its routines from custom code. The user's guide [32] also provides description of the method and further references.

LHS was later extended to Orthogonal Sampling (based on so-called orthogonal arrays) [28], which should allow even greater reduction in the number of samples to obtain a usable representation of the required distribution, but no software implementation has been known to the authors, which would provide comparable features to `LHS`.

Part IV

Application in instrumented indentation

Chapter 6

Estimating parameters of the unloading curve

In order to facilitate pre-fitting the unloading curve data with ordinary NLS using Levenberg-Marquardt method as well as to use OEFPIL function fitting algorithm (main `OEFPIL` function was described earlier), a new function fitting framework has been created in Niget software to provide support for fitting algorithm selection. The new framework enables to include new software components for function fitting:

- For ordinary NLS fitting, CMinpack (`http://devernay.free.fr/hacks/cminpack`, a reimplementation in C of a proven and widely used FORTRAN MINPACK code) has been included in Niget.
- The ODRPACK95 interface has been reworked to match the new framework.
- The OEFPIL C library has been included in Niget.

Moreover, the Oliver-Pharr analysis tool was modified appropriately to include selection of the fitting algorithm. To ensure better convergence (and also aiming to achieve some calculation speed-up by saving more computationally costly iterations of the advanced algorithms), both ODR and OEFPIL fitting are internally preceded by a NLS fit. Screenshot of Niget software in figure 6.1 shows new modified improved version of Oliver-Pharr analysis tool.

The fitting algorithm selection allows to easily compare results provided by ODR function and OEFPIL function (assuming constant uncertainties for both h and F). Although based on different concepts, in result both algorithms yield parameter estimates that to a surprising accuracy See Chapter 9 for details.

Including contact point uncertainty In nanoindentation the contact point stands for the point where the tip touches the sample. For the subsequent calculations, the load and depth data are shifted so that the contact point corresponds to zero load and zero depth. The determination of the contact point introduces not only a contribution to the uncertainty of the individual datapoints, but also correlation among them. The covariance matrix for the depth values on the unloading curve thus becomes $\mathbf{\Sigma}_h$ as follows

$$\mathbf{\Sigma}_h = \begin{pmatrix} uh^2_{\text{contact}} + uh^2_{\text{noise}} & uh^2_{\text{contact}} & \dots & uh^2_{\text{contact}} \\ uh^2_{\text{contact}} & uh^2_{\text{contact}} + uh^2_{\text{noise}} & \dots & uh^2_{\text{contact}} \\ \vdots & \vdots & \ddots & \vdots \\ uh^2_{\text{contact}} & uh^2_{\text{contact}} & \dots & uh^2_{\text{contact}} + uh^2_{\text{noise}} \end{pmatrix}, \tag{6.1}$$

where uh_{noise} is the noise RMS (Root Mean Square) of the depth sensor signal and uh_{contact} is the uncertainty of the depth of first contact. An analogous form can be derived for the

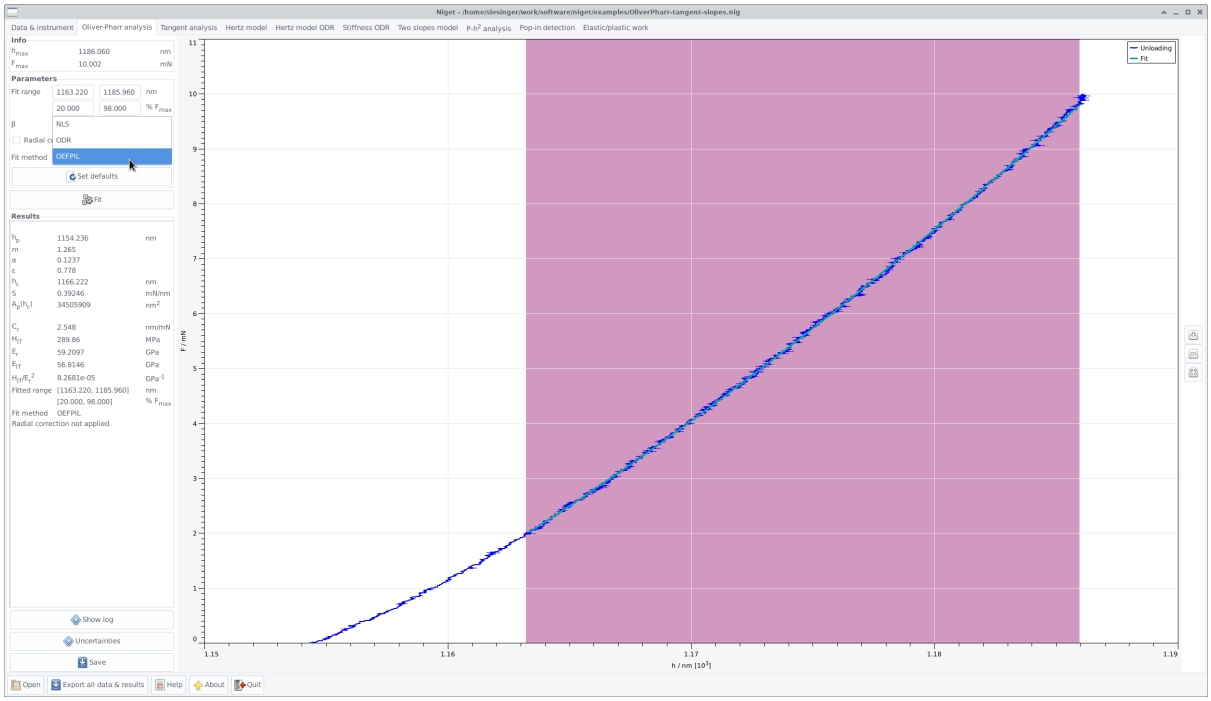


Figure 6.1: Screenshot of Niget software demonstrating new modified improved version of Oliver-Pharr analysis tool and custom selection of algorithm.

covariance matrix of load values. The option to specify these uncertainties has been added to Niget software, and the covariance matrices as presented above are taken into consideration when using OEFPIL for the fitting of the unloading curve.

Chapter 7

Uncertainty propagation

7.1 Matrix-based uncertainty propagation

The basic functionality of the `propagate` function from the `propagate` package for R has been implemented in a simple C library and included in Niget. It is also available separately at `https://gitlab.com/cmi6014/unc-propag`. The present version of the library allows to calculate first- and second-order error propagation for scalar functions, and, in addition, first-order propagation for vector functions using a straightforward generalization of formulas (5.1) and (5.2):

$$E[\mathbf{y}] = f(\mathbf{x}), \tag{7.1}$$

$$\mathbf{\Sigma_y} = \mathbf{J}\mathbf{\Sigma_x}\mathbf{J}^\top, \tag{7.2}$$

where $f\colon \mathbb{R}^n \to \mathbb{R}^m$ is a function with the Jacobian $\mathbf{J}$, $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ with covariance matrix $\mathbf{\Sigma_x}$, and $\mathbf{y} = (y_1, \ldots, y_m) = f(\mathbf{x})$.

The possibility to generalize the second-order propagation formulas (5.3) and (5.4) to vector functions is being investigated.

The uncertainty estimation framework of the Oliver-Pharr tool was then rewritten to make use of this library.

7.2 Monte Carlo simulations

The `LHS` library (see Section 5.2) for effective generation of pseudorandom data for simulations has been included in Niget to extend the capabilities of Monte Carlo-based uncertainty estimations. It is now possible to introduced correlations among the variables, and using LHS also provides greater computation efficiency. The Monte Carlo uncertainty estimation of the Oliver-Pharr tool has been adapted to use the library.

Pseudorandom data generation has been implemented in two variants. In both cases pseudorandom values for the uncertainty of the maximum load and depth, Poisson's ratios of the sample and the tip, Young's modulus of the tip, and coefficients of the fractional polynomial fit of the area function are generated. In addition,

- the "short" version includes the generation of the power-law fit parameters α, m and h_p,
- The "full" variant (an experimental feature at present, more time-consuming and demanding more memory) generates also individual depth and load values on the fitted part of the unloading curve, and calculates the fit for each pseudorandom data set, from which the parameters α, m and h_p are extracted.

Chapter 8

Calibration of the contact area function

In the task of calibration of the tip area function as described in Section 2.2, the problem structure involves correlated data with uncertainties if a proper solution is desired. It is therefore ideally suited for solving using OEFPIL.

As the computation schema presented in Figure 2.2 suggests, the structure of the complete covariance matrix to be presented to OEFPIL is nontrivial (all the blocks are square, dimension (M, M) where M is the number of data points):

$$\begin{pmatrix} \mathbf{\Sigma}_{h_c} & \mathbf{\Sigma}_{h_c A_p} \\ \mathbf{\Sigma}_{h_c A_p} & \mathbf{\Sigma}_{A_p} \end{pmatrix}. \tag{8.1}$$

The blocks $\mathbf{\Sigma}_{h_c}$ and $\mathbf{\Sigma}_{h_c A_p}$ are diagonal (there is no correlation between $h_{\mathrm{c},i}$ and $h_{\mathrm{c},j}$ or $A_{\mathrm{p},j}$ for $i \neq j$), and $\mathbf{\Sigma}_{A_p}$ has all entries non-zero (all values of the contact area are correlated through the properties of the indenter and the reference sample).

8.1 New software tools

Incorporating the contact area calibration functionality in the graphical user interface of Niget would require substantial rewrite of the software due to the need of simultaneous evaluation of multiple indentation curves. It was therefore decided to implement the tip area calibration as a collection of command line tools. Two forms of the tip area function are presently supported: polynomial and fractional polynomial, with an arbitrary number of terms.

For tool configuration and storing intermediate data, JSON (`https://www.json.org/`) file format is employed, which is both human-writeable and easy to process in most programming languages. An example JSON file containg the description of an area function is shown in Listing 8.1. Loading the contact area functions as saved by the area calibration tool has been implemented in present version of Niget.

Listing 8.1: Example JSON area file describing a polynomial function with 3 terms.

```
{
    "name": "Contact area function test",
    "indenter": "Berkovich BV123",
    "sample": "Fused Silica",
    "date": "2021-03-27 22:11",
    "fcntype": "poly0",
    "nterms": 3,
    "params": [
        347.65997951992944,
        26.675159492250888,
```

```
        0.0039982258752850955
    ],
    "paramcov": [
        328.19304797236151,
        -5.1391505746785455,
        0.019809014661914454,
        -5.1391505746785455,
        0.11664499161235975,
        -0.00038360135724398 23,
        0.019809014661914454,
        -0.00038360135724398 23,
        1.5650449109512977e-6
    ]
}
```

The new software tools for contact area function calibration essentially capture all the techniques investigated in this project:

1. fitting unloading curves with errors in both variables, including covariance of parameter estimates,
2. propagation of uncertainty from multiple sources for producing depth-Area data together with covariance information, and
3. fitting the area function through correlated data (and calculating the uncertainty of the area function evaluation).

The new tools are the following:

area-preprocess Takes a list of files saved from Niget (having the unloading section of the data defined), and based on the fit settings for the Oliver-Pharr evaluation tool, estimates the power-law parameters for each curve. The estimates for $h_{\max}$ and $F_{\max}$ are presently based on user's experience, and should be provided in a configuration file.

Example use:
`area-preprocess list.txt op-cfg.json area-prep-cfg.json prep.json`

area-calibrate Takes preprocessed data from *area-calibrate*, information on the tip and reference sample, and calculates the area function in the desired form (polynomial or fractional polynomial, any number of terms). The complete structure of the covariance matrix as presented in Equation (8.1) is used in the calculation. The C library for uncertainty propagation presented in the previous section is used to calculate the entries of the covariance matrix.

Example use:
`area-calibrate prep.json tip.json sample.json area-cal-cfg.json area.json`

Apart from the area function description in *area.json*, additional files containing the depth-area data as well as their covariance matrix (each block separately) are saved in the working directory for further analysis using external tools.

area-eval A simple tool to evaluate the provided tip area function at a specified h_c with given uncertainty. This tool also uses the C library for uncertainty propagation.

Example use:
`area-eval area.json 100.0 0.5`

Example configuration files for all the tools above are distributed with Niget.

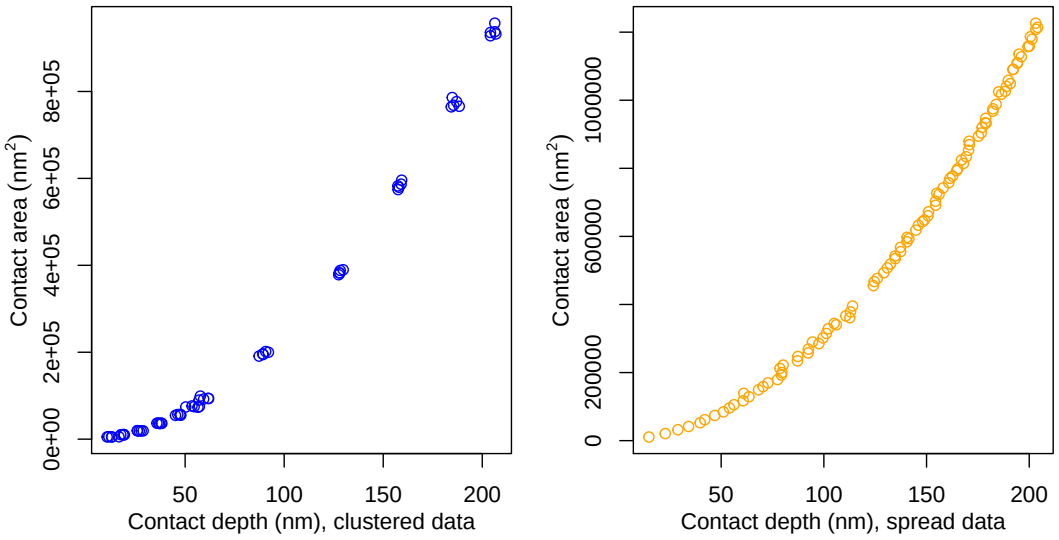


Figure 8.1: Example of clustered (blue points) and spread (orange pints) data set.

8.2 Evaluation of different area function forms

The new methods and tools developed in this project can provide a better insight in choosing a suitable form of the tip area function. Two basic forms of the area function have been considered:

- "fractional polynomial" (FP)

$$A_{\mathrm{p}}(h) = a_1 h^2 + a_2 h + a_3 h^{\frac{1}{2}} + a_4 h^{\frac{1}{4}} + \ldots a_n h^{\frac{1}{2^{n-2}}}, \quad n \in \mathbb{N},$$

- polynomial (P)

$$A_{\mathrm{p}}(h) = a_1 h + a_2 h^2 + \cdots + a_n h^n, \quad n \in \mathbb{N}.$$

Functions with the number of additive terms n from 3 up to 6 were calculated for several collections of area calibration data. There were two principal types of the calibration data:

- "clustered": 12 groups of 5 indents were measured at different maximum loads (with individual contact depths and areas slightly varying in each group due to measurement and sample imperfections),
- "spread": each indent was measured using a different maximum load, providing $(h_{\mathrm{c}}, A_{\mathrm{p}})$ pairs covering (not necessarily uniformly) the whole calibration interval.

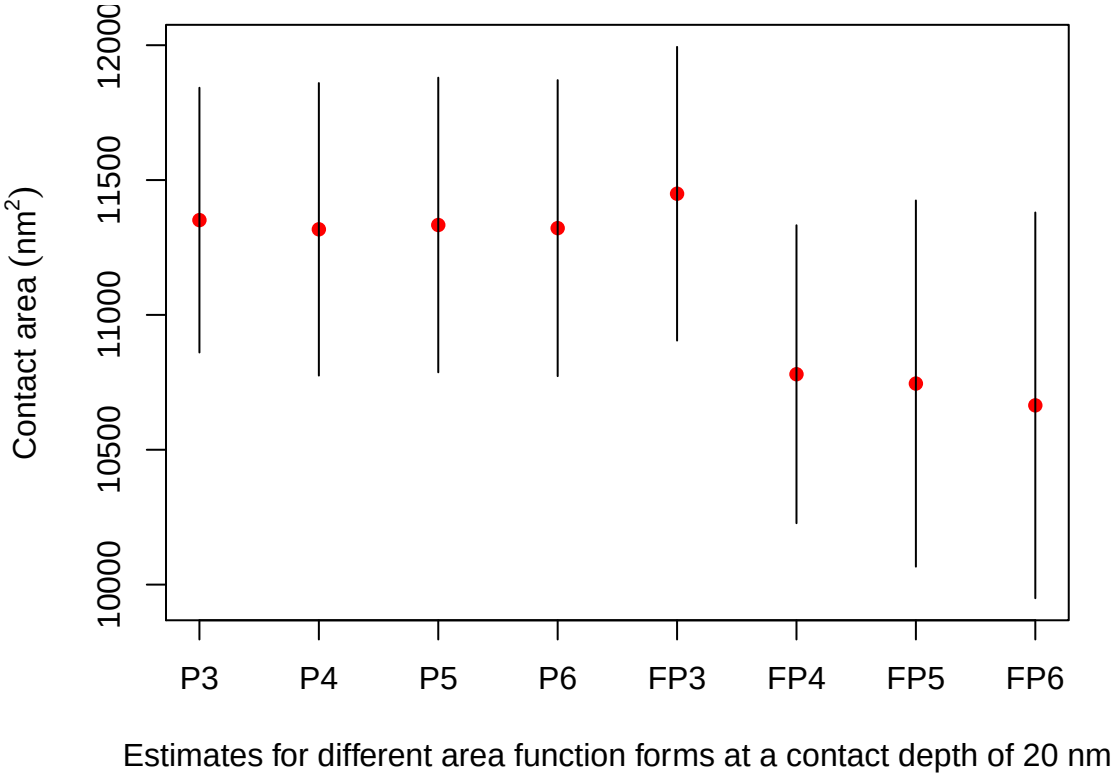
These two types of data are demonstrated in Figure 8.1.

8.2.1 Comparison of evaluated functions at different contact depths

The different forms of the tip area function were first evaluated on a testing set of clustered and spread data, and the values and uncertainties of the functions evaluated at contact depths of 20, 50, 100, and 200 nm were compared. Figures 8.2 and 8.3 show their behavior on clustered data, and figures 8.4 and 8.5 show their behavior on spread data. The numbers in the function form label stand for the number of additive terms in the respective form.

Some findings can be drawn from this comparison:

1. All the function forms yield comparable uncertainties of the evaluated area at all given contact depths (except of FP6 for small depths, where some data files showed an increase of uncertainty – see Figure 8.4).



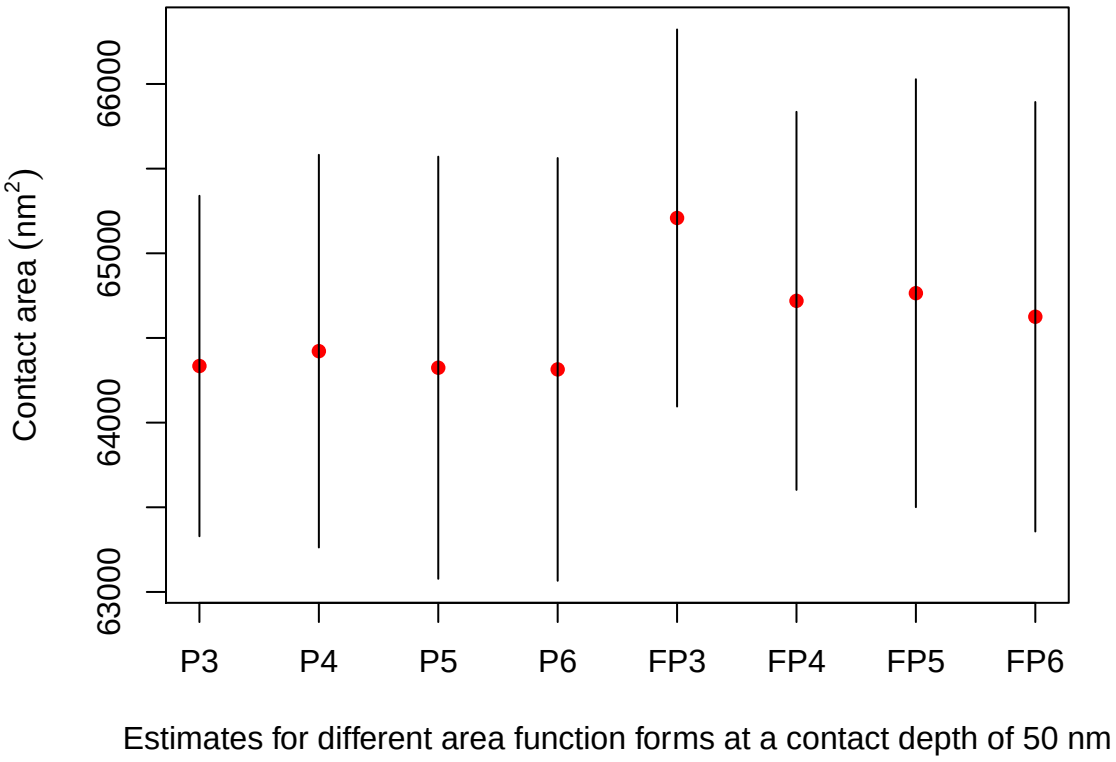
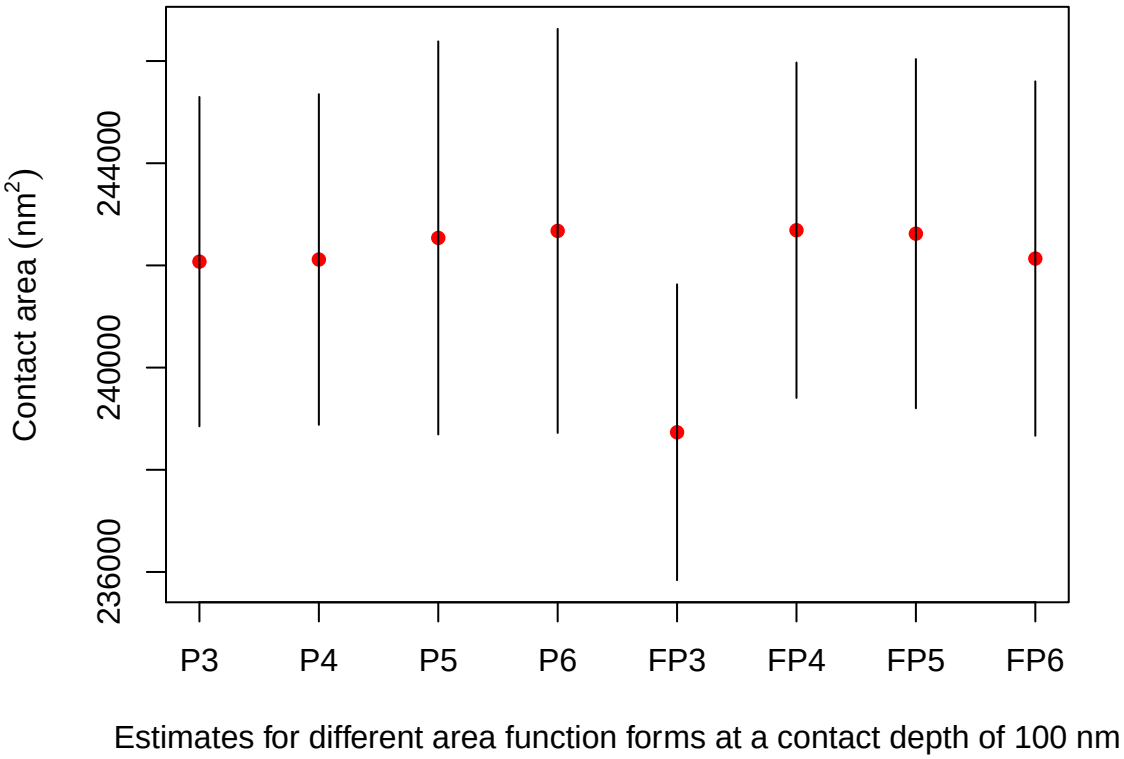


Figure 8.2: Estimates of area function using different polynomials (P) and fractional polynomials (FP) with 95% CI at small contact depths, clustered data.



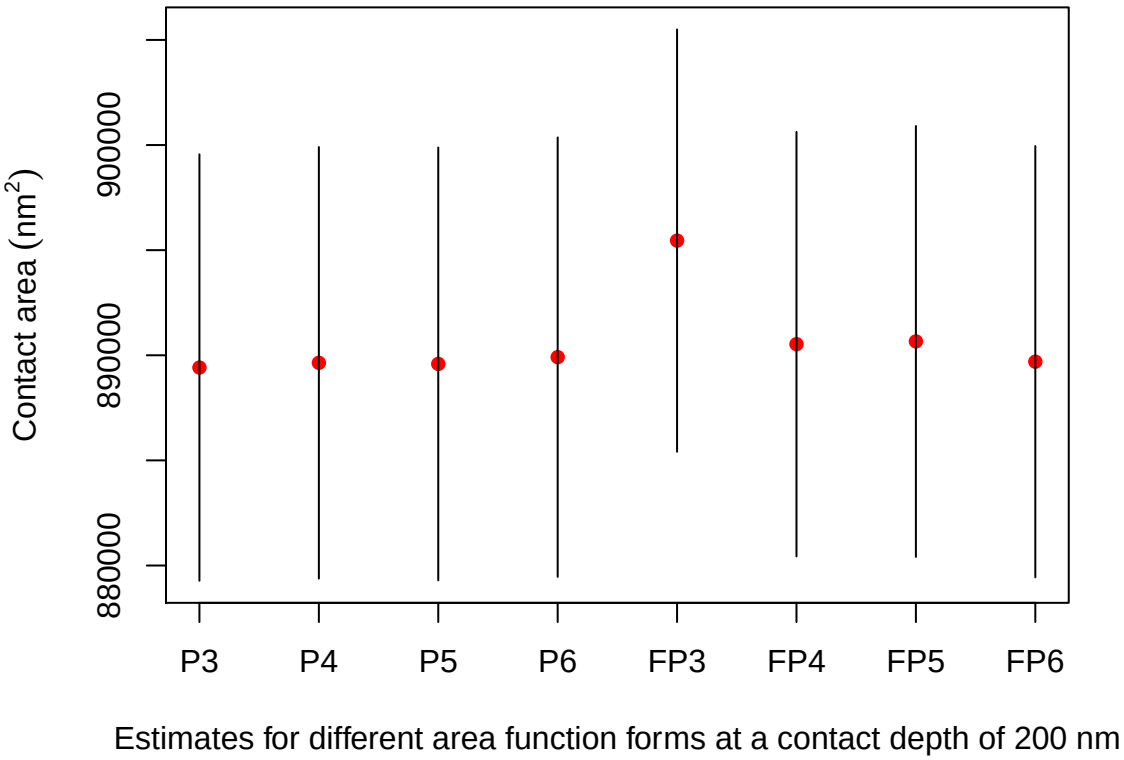
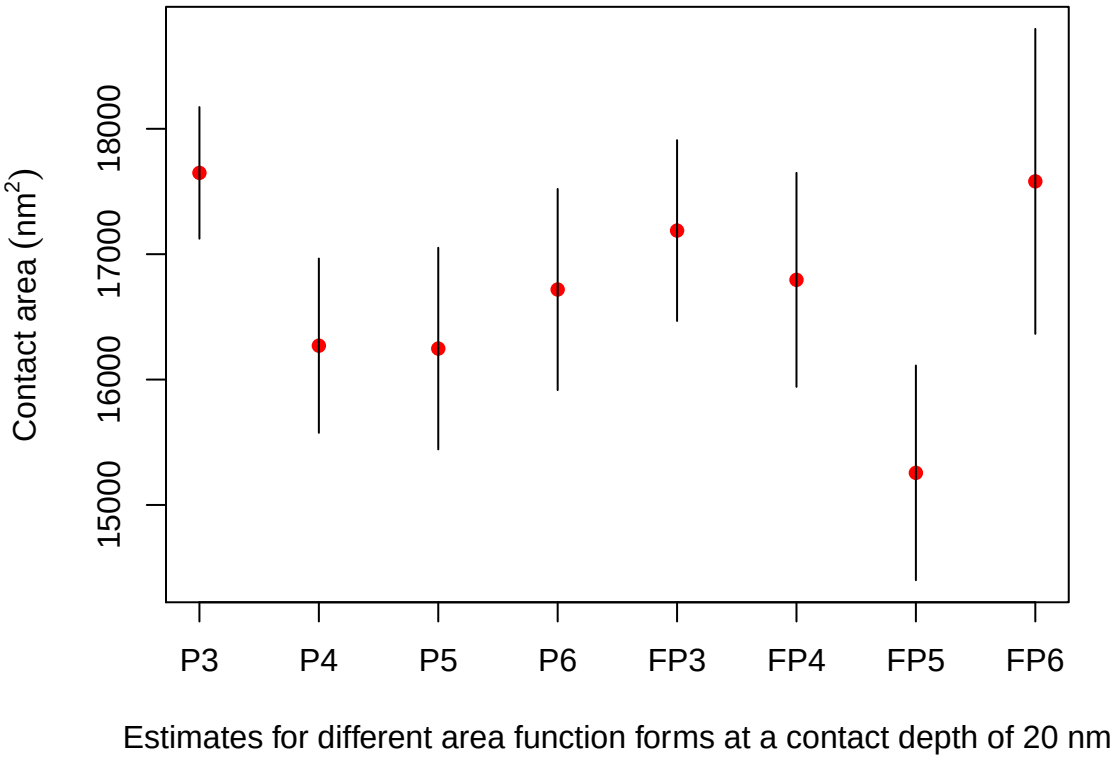


Figure 8.3: Estimates of area function using different polynomials (P) and fractional polynomials (FP) with 95% CI at higher contact depths, clustered data.



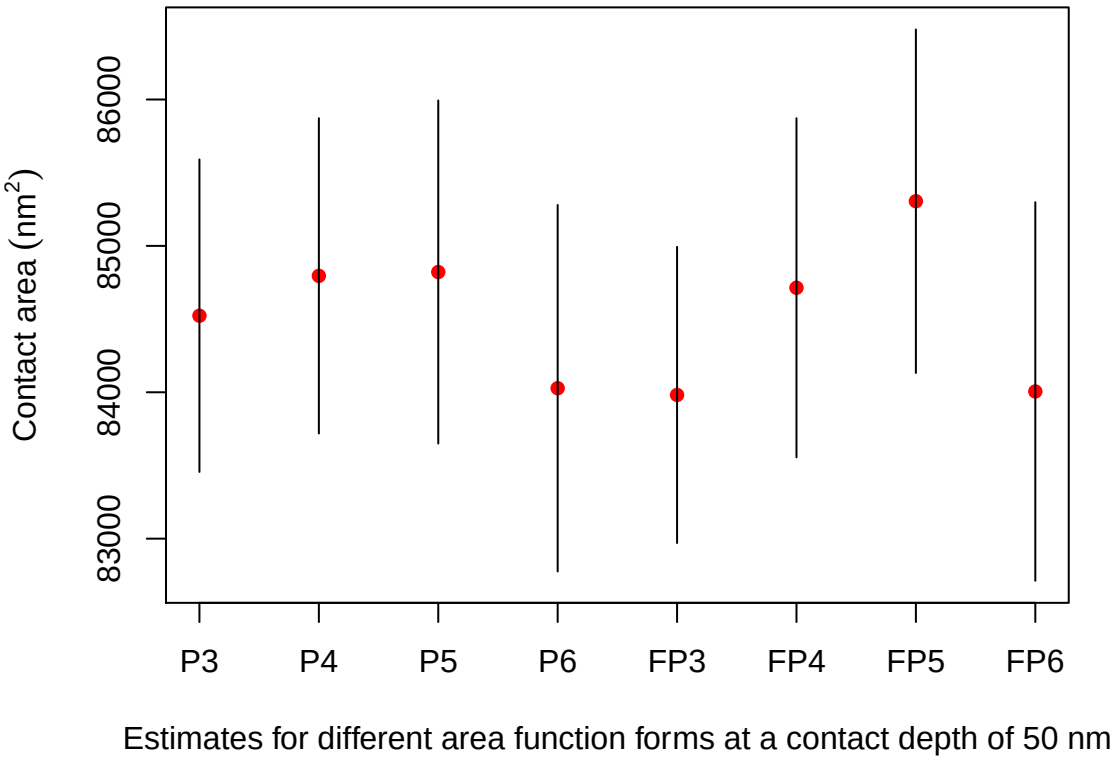
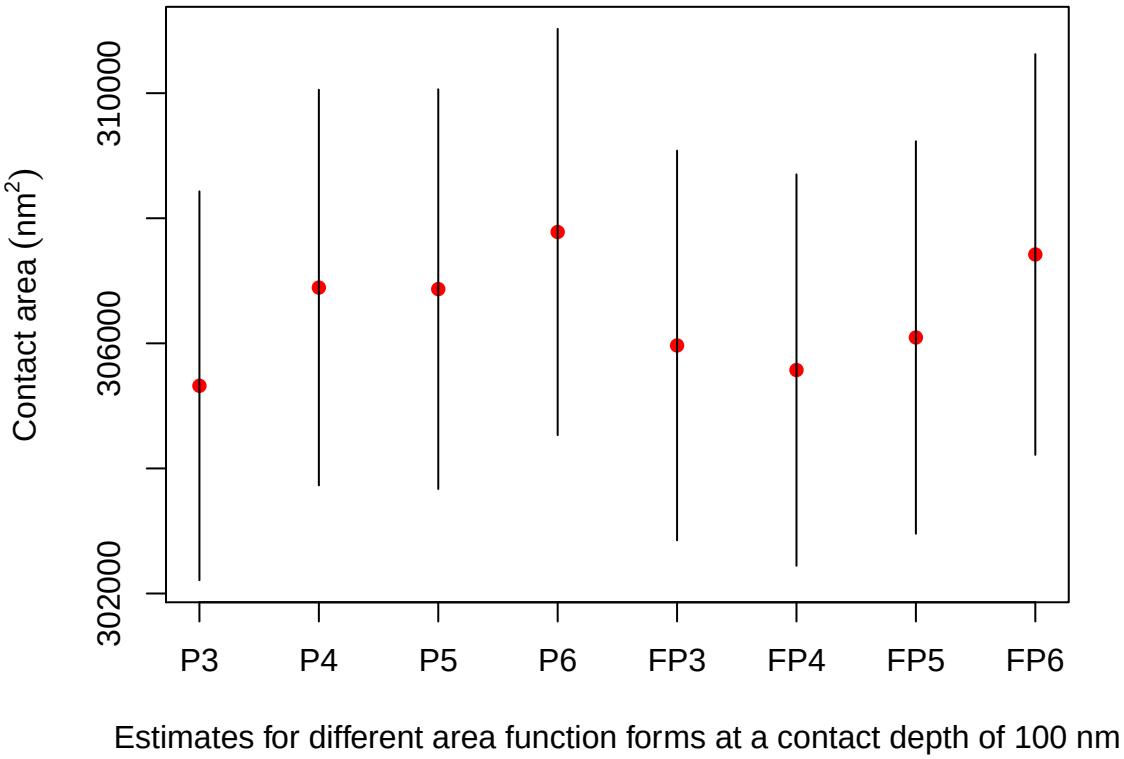


Figure 8.4: Estimates of area function using different polynomials (P) and fractional polynomials (FP) with 95% CI at small contact depths, spread data.



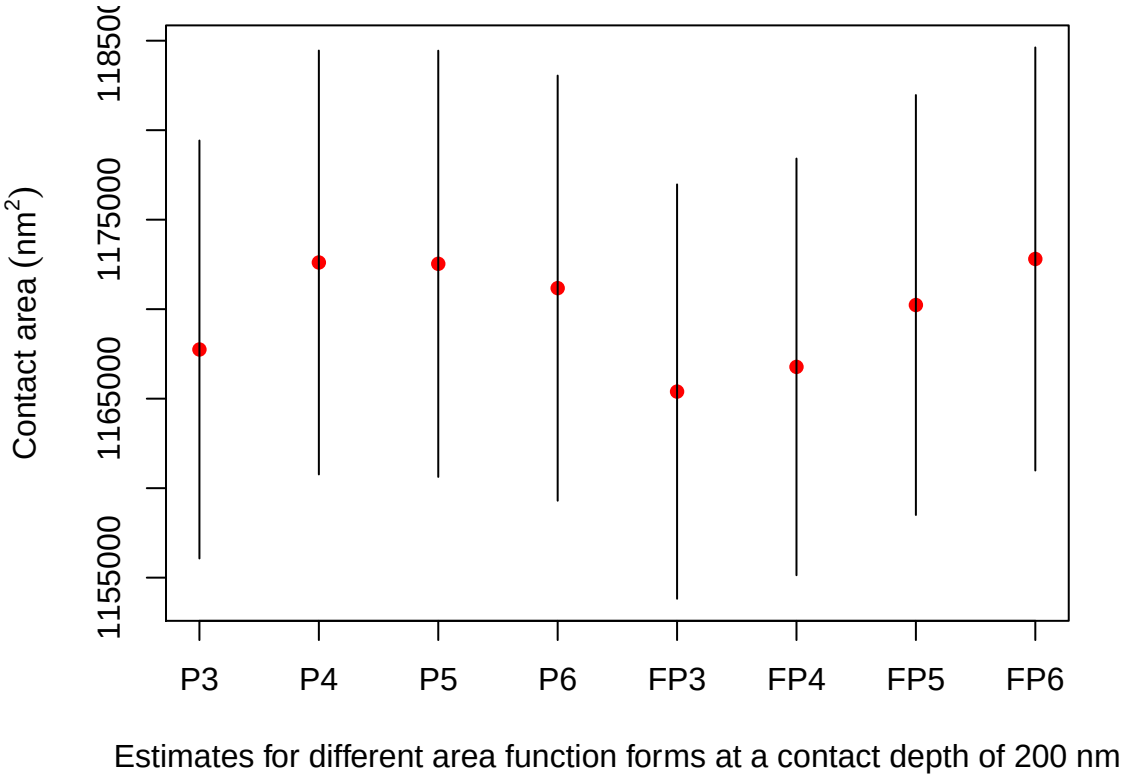


Figure 8.5: Estimates of area function using different polynomials (P) and fractional polynomials (FP) with 95% CI at higher contact depths, spread data.

2. With fractional polynomials, 3 terms are often not sufficient, and the function tends to deviate most from the other forms.

3. At low contact depth, fractional polynomials of higher term number tend to depart quickly from reasonable values. Figure 8.6 shows a comparison between polynomials and fractional polynomials with 4 or 5 terms. Polynomials do not show this behaviour as they approach zero due to the absence of a constant term. For measurements at the lowest depths, the polynomial form of the area function should therefore be preferred.

Care should also be taken, when using polynomial function forms with a higher number of terms, to check that overfitting does not occur and the calculated function is convex.

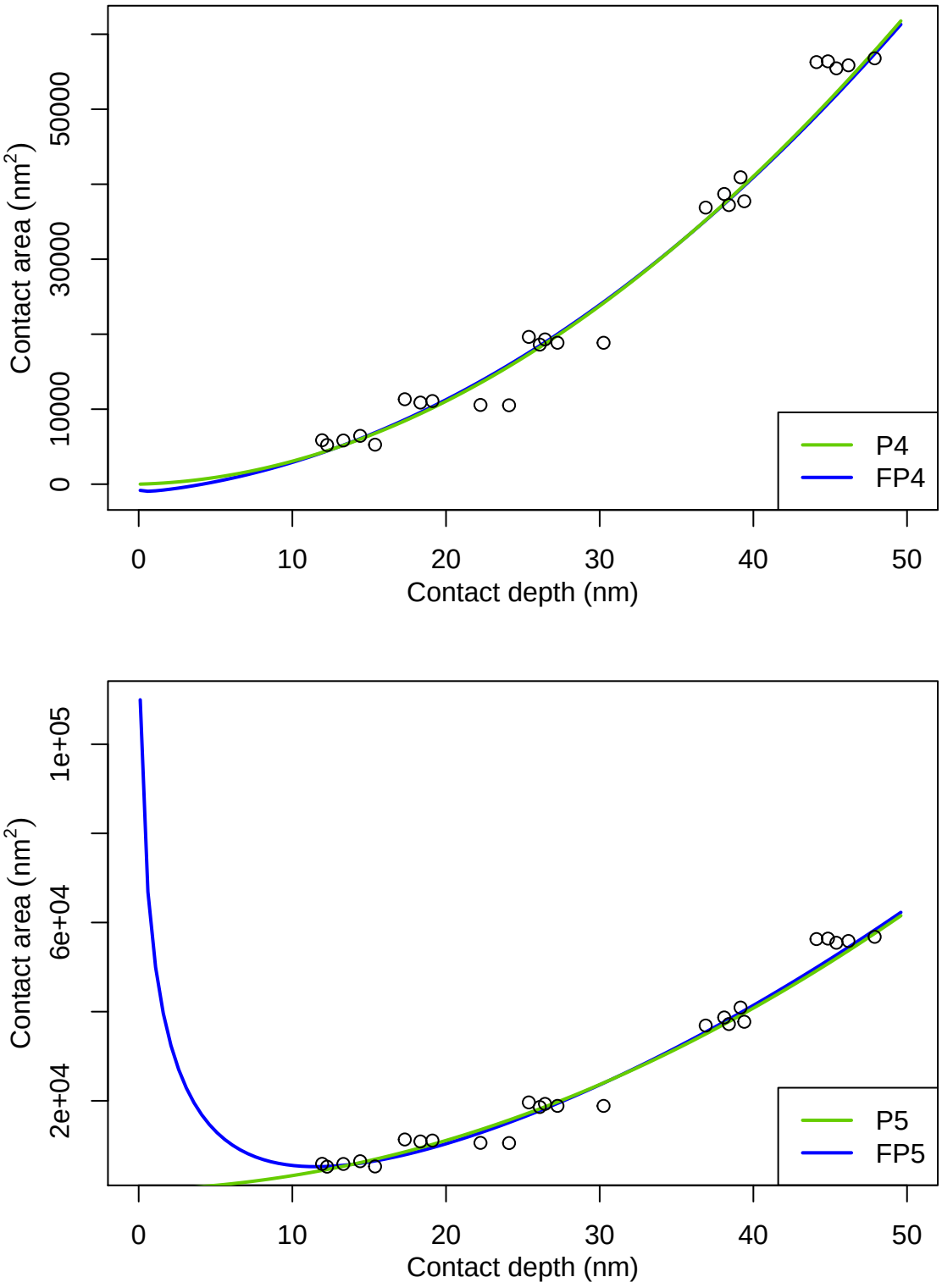


Figure 8.6: Estimation of contact area function at small contact depths using a polynomial (green line) and fractional polynomial (blue line) with 4 terms (top) and 5 terms (bottom), clustered data.

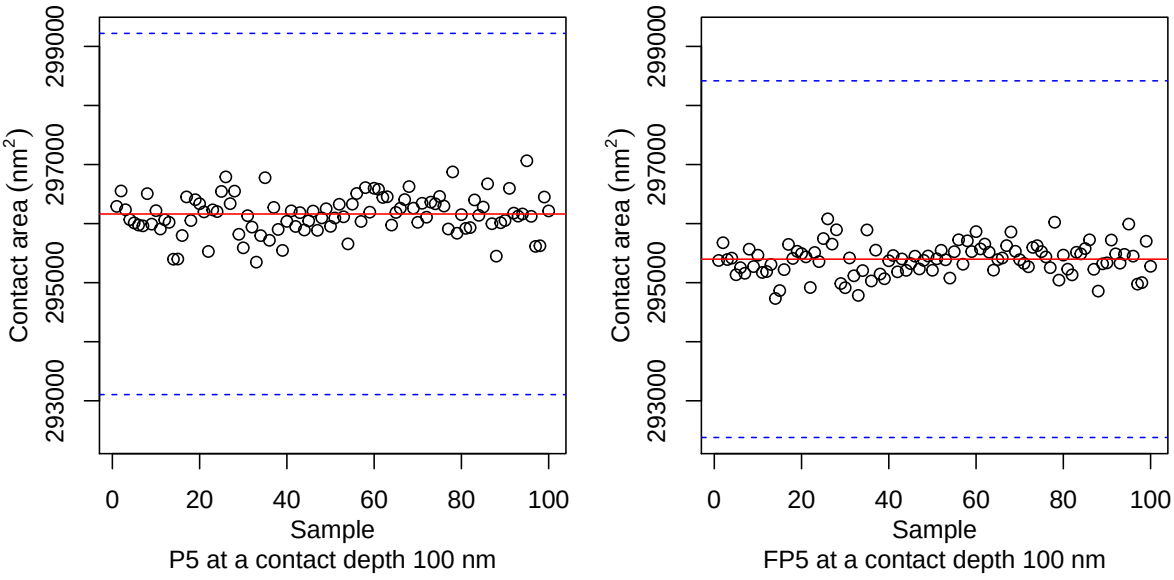


Figure 8.7: Estimations of contact area at a contact depth 100 nm using resampling procedure for polynomial and fractional polynomial with 5 terms, spread data. The red line indicates the whole set estimate with 95% confidence interval (blue lines).

8.2.2 The behaviour of different area function forms

The method's robustness was tested using the following resampling procedure: Random 5 pairs of contact depth and the projected area was excluded from the dataset, and the estimation of the contact area function from this truncated data was computed using OEFPIL. This procedure was repeated 100 times, and the individual estimates were compared with the estimate obtained using the complete dataset. For clustered data, to avoid the exclusion of one complete cluster, the procedure was modified. First, five clusters were randomly selected, and then one random value from each cluster was removed from the dataset. The results are displayed in Figures 8.7 and 8.8. For both considered types of data, the behaviour of various (fractional) polynomials differs with concrete dataset and selected contact depth, but deviations between particular estimates are, with rare exceptions, within the confidence interval calculated from the complete dataset. The results also confirmed greater deviations between different contact area forms in lower depths.

Note: Although the estimate for individual depths often varies, the scale of the y-axis is so large that these changes do not greatly affect the final shape of the curve (except for small depths).

A difference between P and FP can also be observed in their computational properties: fitting the area function with OEFPIL generally requires more iterations when using FP compared to P form of the area function (the exact numbers depend on the number of terms and required convergence threshold).

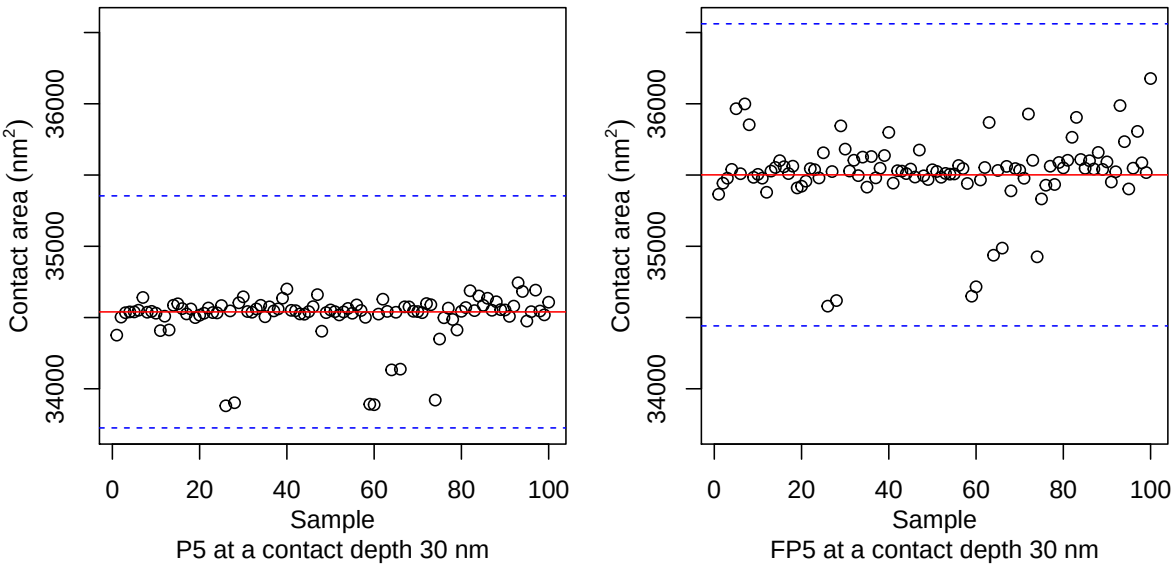


Figure 8.8: Estimations of contact area at a contact depth 30 nm using resampling procedure for polynomial and fractional polynomial with 5 terms, spread data. The red line indicates the whole set estimate with 95% confidence interval (blue lines).

8.2.3 Influence of correlations present in area calibration data

One of the key achievements of this project is the implementation of calculation procedures that enable to take account of the correlations between h_c and A_p for each point as well as for A_p between different points. The effect of neglecting all the correlations involved is demonstrated on an instance of clustered data (Tables 8.1 and 8.2, visualization in Figures 8.10 and 8.12) and spread data (Tables 8.3 and 8.4; also visualized in Figures 8.9 and 8.11).

While the differences in the evaluated area function values can be considered insignificant, the effect of the neglection on the calculated uncertainty is notable: from overestimating by approx. 10–40 % at lower contact depths to underestimating by more than 25 % (less precise instrument, clustered data) or 70 % (more precise instrument, spread data) at higher contact depths.

Table 8.1: Clustered data, contact depths 20 and 50 nm.

		A_p	σ	$A_{p,diag}$	σ_{diag}
$h_c = 20\,\mathrm{nm}$	P3	9547.59	235.88	9765.67	320.53
	P4	9330.18	267.06	9621.55	374.22
	P5	9187.04	282.69	9429.08	402.73
	P6	9143.64	298.17	9365.26	418.46
	FP3	9225.96	275.91	9455.13	390.05
	FP4	9235.06	301.62	9429.25	407.99
	FP5	9421.42	366.35	9525.17	463.75
	FP6	8395.27	399.47	8873.31	538.27
$h_c = 50\,\mathrm{nm}$	P3	58018.58	469.44	58281.04	492.85
	P4	58480.99	538.76	58525.15	592.91
	P5	59001.98	591.76	59006.75	673.09
	P6	59189.14	598.71	59116.66	684.23
	FP3	58758.05	521.29	58805.36	566.06
	FP4	59035.81	548.95	58962.49	602.26
	FP5	58743.45	599.00	58797.46	685.97
	FP6	59215.06	594.90	59082.76	678.75

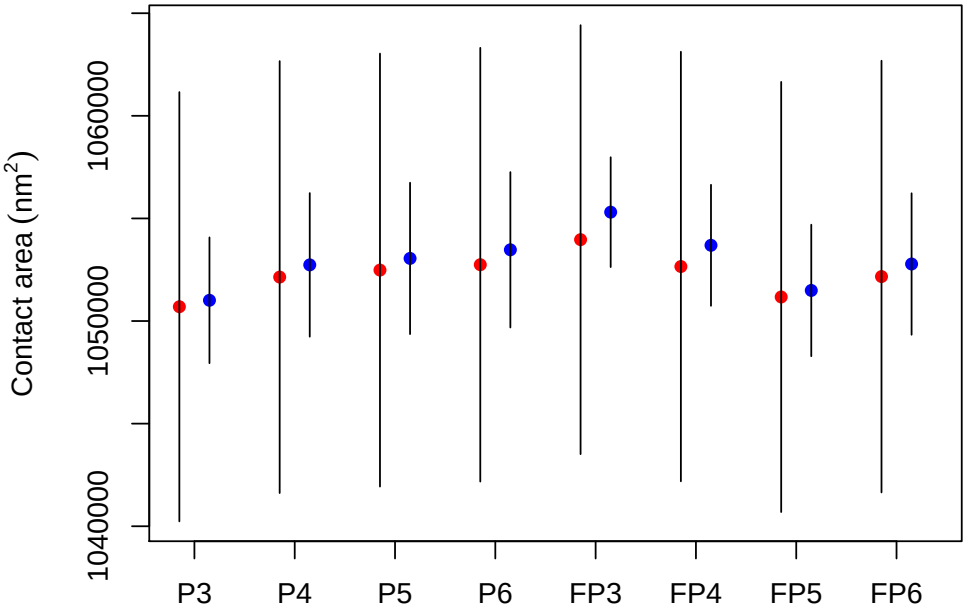


Figure 8.9: Comparing contact area function fitting without (blue points) and with (red points) correlations and 95% confidence intervals of estimate, spread data.

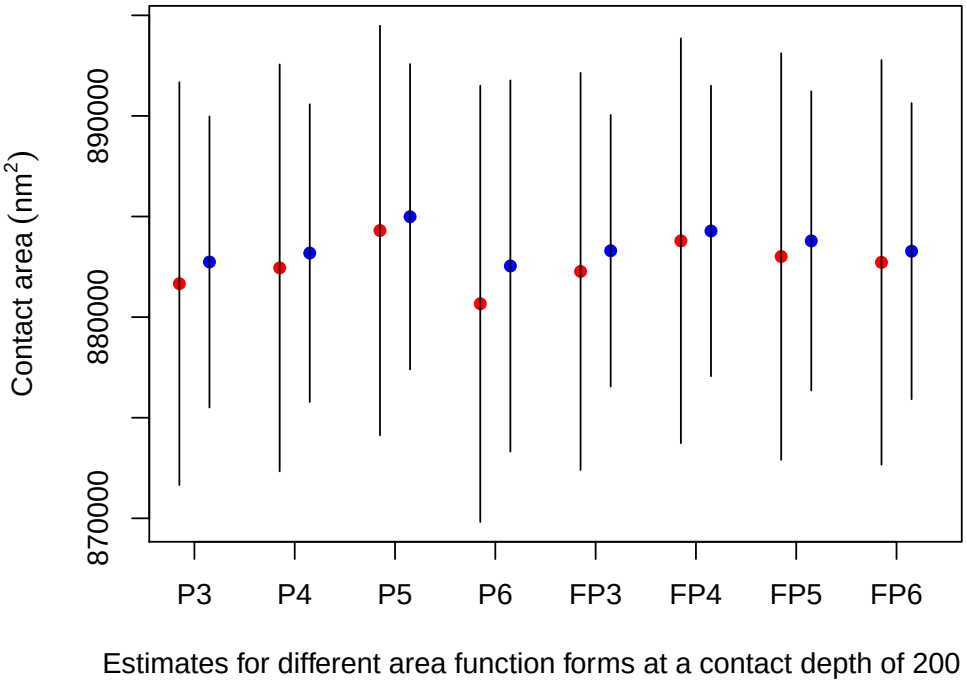


Figure 8.10: Comparing contact area function fitting without (blue points) and with (red points) correlations and 95% confidence intervals of estimate, clustered data.

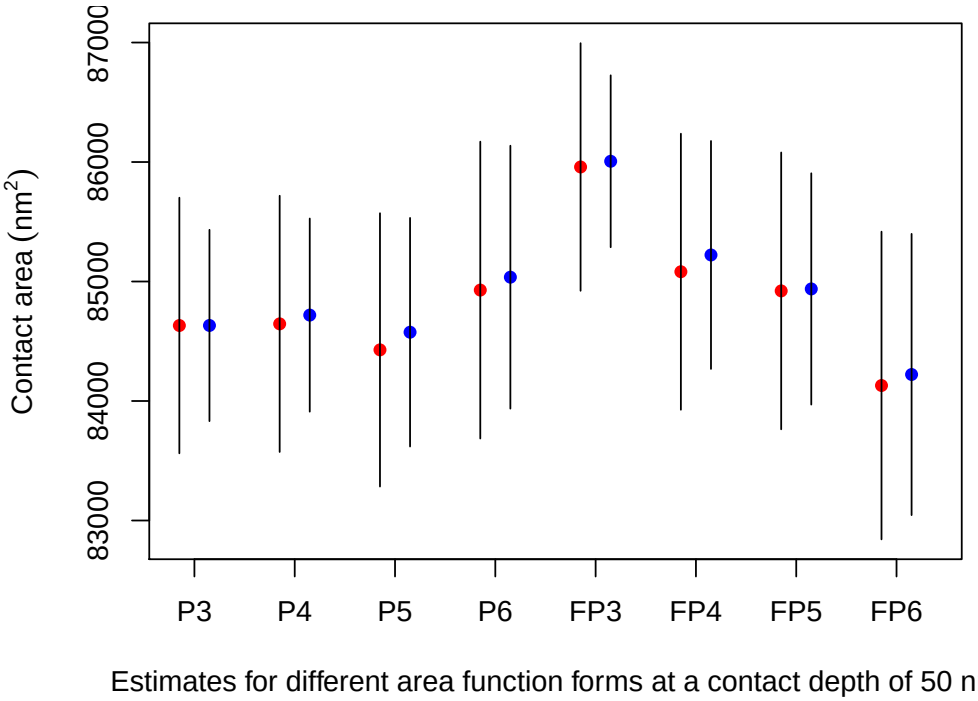


Figure 8.11: Comparing contact area function fitting without (blue points) and with (red points) correlations and 95% confidence intervals of estimate, spread data.

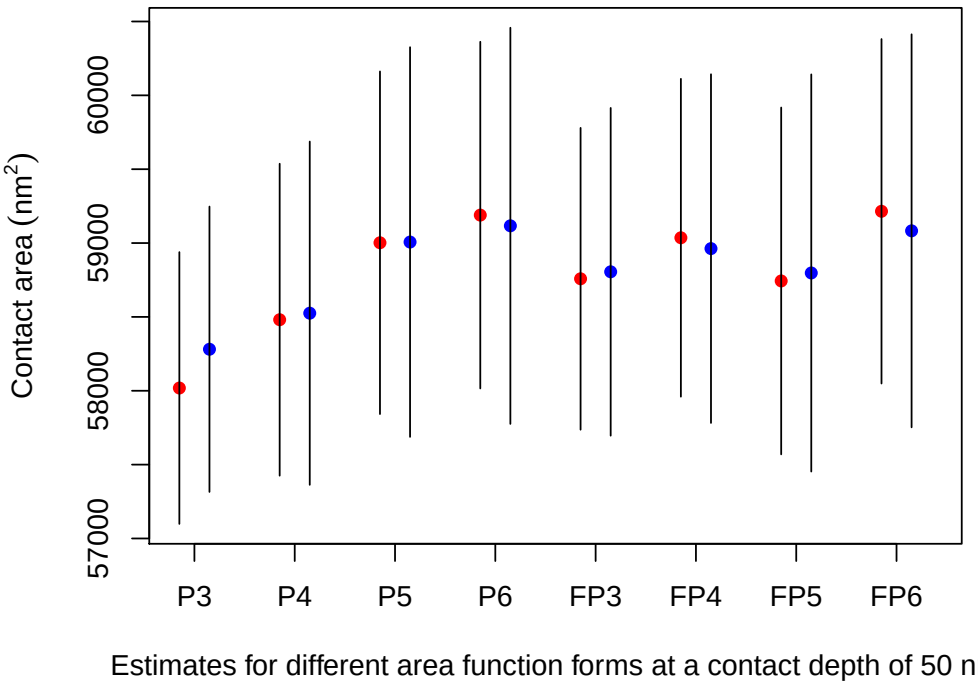


Figure 8.12: Comparing contact area function fitting without (blue points) and with (red points) correlations and 95% confidence intervals of estimate, clustered data.

Table 8.2: Clustered data, contact depths 100 and 200 nm.

		A_p	σ	$A_{p,diag}$	σ_{diag}
$h_c = 100\,\text{nm}$	P3	227394.78	1547.77	227398.30	1428.03
	P4	227625.03	1557.20	227513.33	1440.09
	P5	225432.34	1839.15	225442.02	1963.15
	P6	224496.71	1907.03	224870.05	2067.57
	FP3	227137.20	1412.58	227073.70	1138.29
	FP4	225891.80	1612.50	226261.72	1542.24
	FP5	226176.20	1637.94	226464.16	1606.81
	FP6	224634.37	1687.86	225240.55	1697.74
$h_c = 200\,\text{nm}$	P3	881661.89	5107.08	882741.56	3690.85
	P4	882446.69	5159.31	883182.76	3774.09
	P5	884304.01	5193.54	884989.79	3871.23
	P6	880663.31	5532.21	882540.23	4707.31
	FP3	882271.58	5034.93	883298.78	3443.88
	FP4	883791.78	5133.19	884282.04	3683.45
	FP5	883012.54	5156.76	883788.93	3793.09
	FP6	882721.51	5132.01	883275.89	3754.02

Table 8.3: Spread data, contact depths 20 and 50 nm.

		A_p	σ	$A_{p,diag}$	σ_{diag}
$h_c = 20\,\text{nm}$	P3	18047.23	264.60	17974.50	299.71
	P4	17329.03	358.42	17391.26	407.87
	P5	17565.86	416.98	17526.75	471.71
	P6	17245.71	458.10	17247.60	514.04
	FP3	16148.59	420.54	16359.78	473.73
	FP4	16810.95	414.99	16856.07	468.63
	FP5	17490.12	538.85	17560.65	605.92
	FP6	18374.70	696.33	18387.30	775.92
$h_c = 50\,\text{nm}$	P3	84631.85	545.80	84632.17	408.52
	P4	84645.63	547.03	84719.06	412.44
	P5	84427.93	583.59	84575.75	488.12
	P6	84928.30	633.82	85036.26	561.36
	FP3	85958.48	528.68	86006.34	367.07
	FP4	85081.91	589.38	85222.21	486.74
	FP5	84921.51	591.04	84937.90	493.90
	FP6	84129.61	656.99	84221.90	600.46

Table 8.4: Spread data, contact depths 100 and 200 nm.

		A_p	σ	$A_{p,diag}$	σ_{diag}
$h_c = 100\,\text{nm}$	P3	295190.70	1524.03	295551.77	523.72
	P4	295998.29	1551.86	296181.30	602.96
	P5	296162.60	1560.07	296322.25	654.27
	P6	295520.45	1587.09	295715.55	745.63
	FP3	295561.46	1531.38	295566.54	531.55
	FP4	295903.98	1535.79	296021.07	560.51
	FP5	295397.94	1540.38	295635.43	575.81
	FP6	296216.59	1573.66	296419.28	687.46
$h_c = 200\,\text{nm}$	P3	1050697.35	5337.18	1051010.87	1563.86
	P4	1052142.27	5369.55	1052736.54	1785.43
	P5	1052483.00	5382.75	1053051.73	1882.36
	P6	1052744.54	5392.53	1053471.77	1931.93
	FP3	1053965.85	5332.77	1055305.77	1365.70
	FP4	1052656.53	5340.54	1053690.62	1505.04
	FP5	1051172.61	5347.53	1051490.63	1635.10
	FP6	1052166.49	5366.92	1052778.54	1759.37

Chapter 9

Example data evaluation

In this section we shall demonstrate the application of the newly developed methods on a real measurement data set. A set of 49 measurements has been measured on a single sample, with the indentations placed a 7×7 square grid. These data have been evaluated by the procedures developed within this project. A typical loading curve is shown in Figure 9.1. The input uncertainties have been estimated as

$$\begin{array}{rcrl} u(h_{\text{noise}}) & = & 0.5 & \text{nm}, \\ u(F_{\text{noise}}) & = & 1 & \mu\text{N}, \\ u(h_{\text{contact}}) & = & 1 & \text{nm}, \\ u(F_{\text{contact}}) & = & 2 & \mu\text{N}, \\ u(E_{\text{i}}) & = & 3 & \text{GPa}, \\ u(\nu_{\text{i}}) & = & 0.01, & \\ u(\nu) & = & 0.05. & \end{array} \tag{9.1}$$

Different fitting methods Comparison is possible only with a diagonal covariance matrix since correlations are not included in NLS nor ODR. Note, that no estimate for the uncertainty of the parameters is available for NLS. While the estimates of the parameters differ only very slightly between ODR and OEFPIL it can be stated that the uncertainties are significantly underestimated by ODR. On the other hand, OEFPIL is more sensitive to the data and fails also in cases where ODR succeeds. An example of the resulting values from the different methods is in tabel 9.1.

Table 9.1: Comparison of the resulting parameter estimates using different fitting methods. In order to show the agreement between ODR and OEFPIL results we list the values with more digits than necessary.

	NLS	ODR	OEFPIL
α	0.000972785	0.000793976 ± 0.000138218	0.000793983 ± 0.000462999
h_{p}	128.495	127.942 ± 0.481088	127.942 ± 1.61239
m	1.99512	2.04405 ± 0.0416521	2.04404 ± 0.139599

Covariance matrix including contact point uncertainty The inclusion of the contact uncertainties uh_{contact} and uF_{contact} into the covariance matrix leads to a very small change in the estimate of the parameters α, m and h_{p}, at least for reasonable values of contact uncertainties, see Table 9.2.

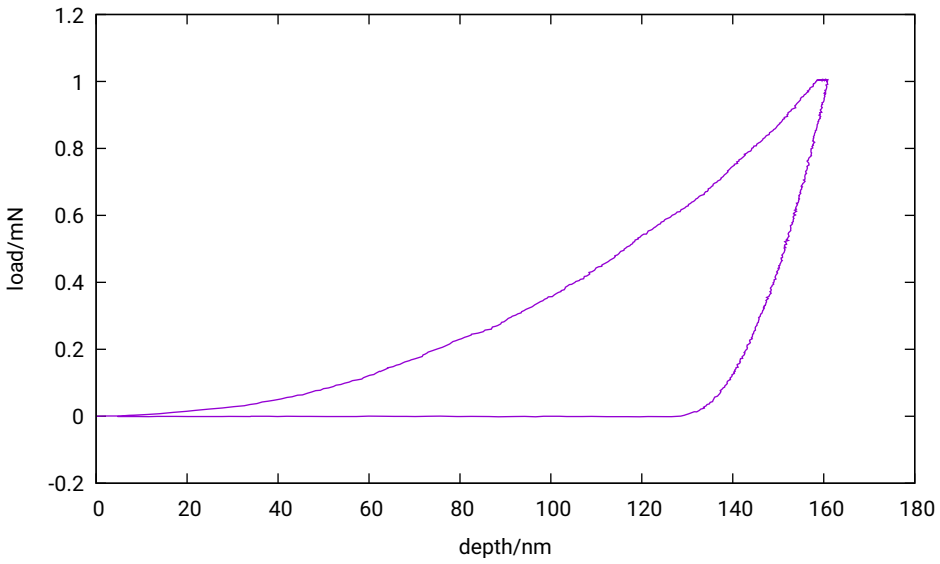


Figure 9.1: Typical loading curve for indentation measurement.

Table 9.2: Comparison of the resulting parameter estimates using either a full covariance matrix or only its diagonal form. For clarity we list the results with more digits.

	Full covariance matrix	Diagonal covariance matrix
$\alpha/(\mathrm{mN\ nm}^{-m})$	0.000794091 ± 0.000464219	0.000793983 ± 0.000462999
h_p/nm	127.943 ± 1.908	127.942 ± 1.612
m	2.04401 ± 0.13988	2.04404 ± 0.13960

Comparison of Monte Carlo and LHS As mentioned previously, LHS has been implemented in Niget in two forms, whether or not the the parameters of the power-law fit α, h_p and m are generated as pseudorandom numbers or whether they are found from individual fits of randomly generated unloading curves. Two drawbacks have been found for the full version of LHS. Firstly, in many cases the covariance matrix of all generated variables is not positive definite. This may indicate an inadequate estimate of our uncertainties involved. Secondly, it requires too much memory when data for hundreds on random variables have to be generated.

A sample output is shown in Figure 9.2. The approximate computation times on a desktop computers were 2 min 54 s (MC), 3 min 55 s (LHS full full covariance matrix), 2 min 15 s (LHS full diagonal covariance matrix) and 0.2 s (LHS short). Here the number of samples 1000 is clearly insufficient. Generation of larger samples will require a reorganization of the code. However, it seems that the multivariate normal distribution for the fitting parameters is an acceptable approximation for the experimental distribution as obtained from Monte Carlo. The fact that the performance of LHS for multivariate distributions need not be superior to Monte Carlo has been explored in literature, e.g. [24]. Nevertheless, the LHS library remains a very useful tool for generating data with complicated correlation structure or distribution functions.

Uncertainty budgets The uncertainty budget in Table 9.3 shows the contributions of various sources of uncertainty, with the total estimates $H_\mathrm{IT} = (1955 \pm 31)$ MPa, $E_\mathrm{IT} = (75.2 \pm 3.1)$ GPa and $E_\mathrm{r} = (77.09 \pm 1.70)$ GPa.

It can be noted that the relevance of the various sources differs for indentation hardness, indentation modulus and reduced contact modulus. First, the large contribution coming from the uncertainty in Poisson's ratio of the sample should be noted. This value is rarely available

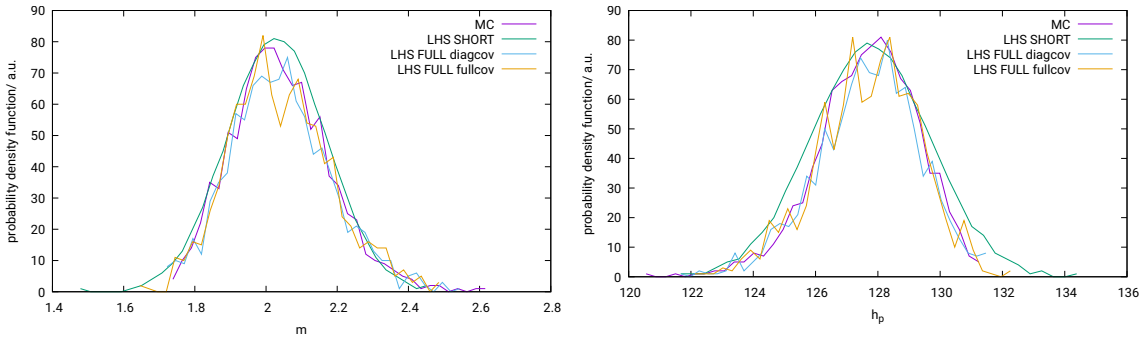


Figure 9.2: The probability density functions for the fitting parameters m and h_p for different evaluation methods.

Table 9.3: Uncertainty budget.

Source	$u(H_\mathrm{IT})$/MPa	$u(E_\mathrm{IT})$/GPa	$u(E_\mathrm{r})$/GPa
Uncertainty of contact	24.9	0.514	0.491
Noise	14.7	1.69	1.62
AF coefficients	11.1	0.229	0.219
ν	0	2.48	0
ν_i	0	0.0076	0
E_i	0	0.0143	0
Total	31.0	3.05	1.703

and is usually taken from literature, without any estimates of its uncertainty. The reduced contact modulus does not suffer from this disadvantage but cannot be compared with results from literature obtained by other methods. Secondly, the indentation hardness is more sensitive to the uncertainty of the contact. This is because it depends on the maximum load whereas the moduli depend on the slope which is not directly affected by the choice of contact.

Repeated measurement In nanoindentation, the common practice is to estimate the uncertainties of hardness and modulus as the standard deviation of the mean. This is in agreement with standard [12]. Uncertainties of a single indentation are not taken into account. This approach leads to the values $H_\mathrm{IT} = (2214.59 \pm 25.657)$ MPa, and $E_\mathrm{IT} = (78.5051 \pm 0.629126)$ GPa. These values are smaller than those obtained by a rigorous analysis of each indentation itself, manifesting thus the limitations of this approach. In Figures 9.4 and 9.3 we can see the fairly large variability of the data. Especially in case of the hardness, values given by different indentations do not agree even within uncertainties.

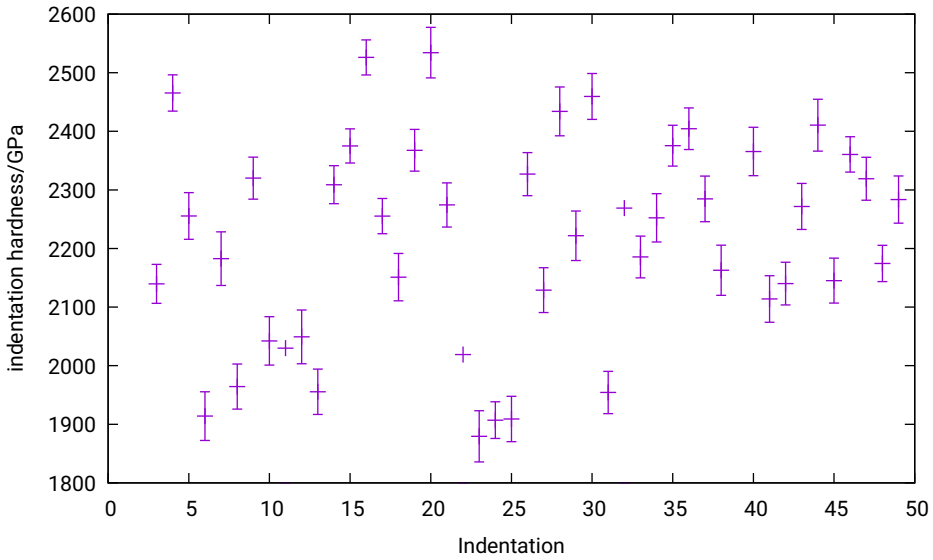


Figure 9.3: Indentation hardness for individual indentations.

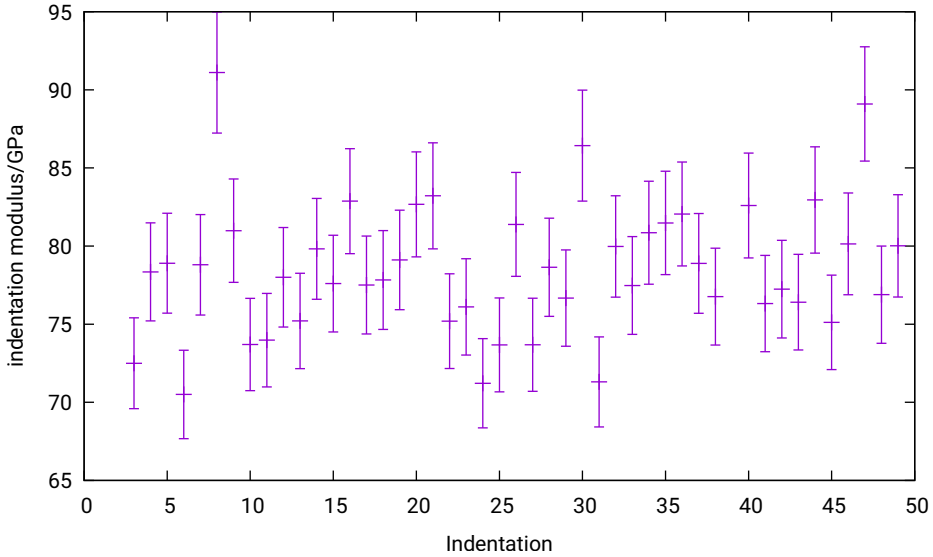


Figure 9.4: Indentation modulus for individual indentations.

Conclusion

All the goals set up in the project proposal have been achieved, and declared outcomes of the project have been produced:

TJ02000203-V1 User software for evaluating data from instrumented indentation measurements The newly developed method for nonlinear data regression, improved methods for uncertainty propagation, and tools for tip area function calibration have been included in Niget and released in version 0.5.9 of the software, available at `http://nanometrologie/niget`.

TJ02000203-V2 Research report describing the methods developed and the implementation of them This report. The general mathematical and statistical methods are dealt with in Part III. Part IV then describes the application of these methods in evaluation of measurements by instrumented indentation technique, and presents novel results that have been obtained using the methods and newly developed tools.

TJ02000203-V3 Software for data evaluation using nonlinear orthogonal regression in R software A new method for nonlinear regression, which allows to account for an arbitrary covariance matrix for the input data, has been developed, implemented, and tested. The R package `OEFPIL` has been written, reviewed, and is freely available at `https://cran.r-project.org/package=OEFPIL`.

C library for nonlinear regression An additional outcome has been produced: a free software library in C for function fitting, implementing the OEFPIL method developed in TJ02000203-V3, has been created, and is freely available at `https://gitlab.com/cmi6014/oefpil`.

Using the newly developed methods and software tools, various data sets from instrumented indentation measurements have been evaluated, which has provided a novel insight to the measured data. Parts of the standardized data evaluation procedures related to function fitting and uncertainty estimation have now gained a more solid mathematical background.

The applicability of the general mathematical and statistical methods developed and explored in this project is obviously not limited to the standardized instrumented indentation procedure for elasticity modulus and hardness determination, on which base they were developed in this project. Presented results and freely available tools developed for indentation data processing aim to impact fundamental metrology, related material research, and operators in industry. Moreover, the nonlinear regression tools developed within this project can be used separately also in other areas where similar complex measurement data evaluation is needed, such as metrology of chemical or electrical quantities, or in the areas of research and data processing outside the metrology community.

The mathematical methods themselves also provide a field for further research and applications, like investigation of nonlinear regression for non-normally distributed data, or higher-order uncertainty propagation of correlated variables in the multivariate setting.

The software tools developed in this project are also planned to be further developed, be extended with new features, and reflect feedback from the users.

Bibliography

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide, third edition*. Society for Industrial and Applied Mathematics, 1999.

[2] D. S. Bernstein. *Scalar, Vector, and Matrix Mathematics: Theory, Facts, and Formulas – Revised and Expanded Edition*. Princeton University Press, 2018.

[3] P. T. Boggs, R. H. Byrd, and R. B. Schnabel. A stable and efficient algorithm for nonlinear orthogonal distance regression. *SIAM Journal on Scientific and Statistical Computing*, 8(6):1052–1078, 1987. doi:10.1137/0908085.

[4] P. T. Boggs, J. R. Donaldson, R. H. Byrd, and R. B. Schnabel. Algorithm 676: ODRPACK: Software for weighted orthogonal distance regression. *ACM Transactions on Mathematical Software*, 15(4):348–364, 1989. doi:10.1145/76909.76913.

[5] A. Charvátová Cambell, P. Grolich, and R. Šlesinger. Niget: Nanoindentation general evaluation tool. *SoftwareX*, 9:248–254, 2019. doi:10.1016/j.softx.2019.03.001.

[6] N. D. Cox. Tolerance analysis by computer. *Journal of Quality Technology*, 11(2):80–87, 1979. doi:10.1080/00224065.1979.11980884.

[7] A. C. Fischer-Cripps. *Nanoindentation*. Mechanical Engineering Series. Springer, 2011.

[8] U. H. G. Aldrich-Smith, N. M. Jennett. Direct measurement of nanoindentation area function by metrological AFM. *Z. Metallkd.*, 96:1267–1271, 2005.

[9] G. H. Golub and C. F. Van Loan. *Matrix Computations, Fourth edition*. Johns Hopkins Studies in the Mathematical Sciences. The Johns Hopkins University Press, 2012.

[10] J. L. Hay and G. M. Pharr. *Mechanical Testing and Evaluation*, volume 8, chapter Instrumented Indentation Testing. ASM International, 2000. doi:10.31399/asm.hb.v08.a0003273.

[11] ISO/IEC. Propagation of distributions using a Monte Carlo method. ISO/IEC 98–3:2008/Suppl 1:2008, International Organization for Standardization, Geneva, Switzerland, 2008.

[12] ISO/IEC. Uncertainty of measurement – part 3: Guide to the expression of uncertainty in measurement (GUM:1995). ISO/IEC 98–3:2008, International Organization for Standardization, Geneva, Switzerland, 2008.

[13] ISO/IEC. 14577-1:2015 Metallic materials – Instrumented indentation test for hardness and materials parameters – Part 1: Test method. ISO/IEC, International Organization for Standardization, Geneva, Switzerland, 2015.

[14] ISO/IEC. 14577-2:2015 Metallic materials – Instrumented indentation test for hardness and materials parameters – Part 2: Verification and calibration of testing machines. ISO/IEC, International Organization for Standardization, Geneva, Switzerland, 2015.

[15] S. Kossman, T. Coorevits, A. Iost, and D. Chicot. A new approach of the oliver and pharr model to fit the unloading curve from instrumented indentation testing. *Journal of Materials Research*, 32:2230–2240, 2017.

[16] M. Krystek and M. Anton. A least-squares algorithm for fitting data points with mutually correlated coordinates to a straight line. *Measurement science and technology*, 22(3):035101, 2011.

[17] L. Kubáček. *Foundations of Estimation Theory*, volume 9 of *Fundamental Studies in Engineering*. Elsevier, 1988.

[18] L. Kubáček and L. Kubáčková. *Statistika a metrologie*. Palacký University Olomouc, 2000.

[19] L. Kubáček and E. Tesaříková. Linear error propagation law and nonlinear functions. *Acta Universitatis Palackianae Olomucensis, Facultas Rerum Naturalium, Mathematica*, 49(2):69–82, 2010.

[20] R. Köning, G. Wimmer, and V. Witkovský. Ellipse fitting by nonlinear constraints to demodulate quadrature homodyne interferometer signals and to determine the statistical uncertainty of the interferometric phase. *Measurement Science and Technology*, 25(11), 2014. doi:10.1088/0957-0233/25/11/115001.

[21] M. Lecuna, F. Pennecchi, A. Malengo, and P. G. Spazzini. Calibration curve computing (CCC) software v2.0: a new release of the INRIM regression tool. *Measurement Science and Technology*, 31, 2020. doi:10.1088/1361-6501/ab7d6e.

[22] A. Malengo and F. Pennecchi. A weighted total least-squares algorithm for any fitting model with correlated variables. *Metrologia*, 50:654–662, 2013. doi:10.1088/0026-1394/50/6/654.

[23] G. Mana and F. Pennecchi. Uncertainty propagation in non-linear measurement equations. *Metrologia*, 44:246–251, 2007. doi:10.1088/0026-1394/44/3/012.

[24] R. Manteufel. Evaluating the convergence of latin hypercube sampling. In *41st Structures, Structural Dynamics, and Materials Conference and Exhibit*, page 1636, 2000.

[25] M. D. McKay, W. J. Conover, and R. J. Beckman. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[26] D. Nečas and P. Klapetek. Gwyddion: an open-source software for SPM data analysis. *Central European Journal of Physics*, 10(1):181–188, 2012. doi:10.2478/s11534-011-0096-2.

[27] W. C. Oliver and G. M. Pharr. An improved technique for determining hardness and elastic-modulus using load and displacement sensing indentation experiments. *J. Mater. Res.*, 7:1564–1583, 1992.

[28] A. B. Owen. Orthogonal arrays for computer experiments, integration and visualization. *Statistica Sinica*, 2(2):439–452, 1992. URL: http://www.jstor.org/stable/24304869.

[29] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020. URL: https://www.R-project.org/.

[30] A.-N. Spiess. *onls: Orthogonal Nonlinear Least-Squares Regression*, 2015. R package version 0.1-1. URL: https://CRAN.R-project.org/package=onls.

[31] A.-N. Spiess. *propagate: Propagation of Uncertainty*, 2018. R package version 1.0-6. URL: https://CRAN.R-project.org/package=propagate.

[32] L. P. Swiler and G. D. Wyss. *A user's guide to Sandia's Latin Hypercube Sampling Software: LHS UNIX library/standalone version*, 7 2004. URL: https://www.osti.gov/biblio/919175, doi:10.2172/919175.

[33] H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. URL: https://ggplot2.tidyverse.org.

[34] H. Wickham and G. Grolemund. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media, Inc., 2017.

[35] V. Witkovský and G. Wimmer. Generalized polynomial comparative calibration: Parameter estimation and applications. In S. Y. Yurish, editor, *Advances in Measurements and Instrumentation: Reviews*, volume 1, pages 15–52. International Frequency Sensor Association Publishing, 2019.

[36] J. W. Zwolak, P. T. Boggs, and L. T. Watson. Algorithm 869: ODRPACK95: A weighted orthogonal distance regression code with bound constraints. *ACM Trans. Math. Softw.*, 33(4), Aug. 2007. doi:10.1145/1268776.1268782.

[37] S. Zámečník, V. Šindlář, Z. Geršlová, and G. Wimmer. *OEFPIL: Optimal Estimation of Function Parameters by Iterated Linearization*, 2021. R package version 0.1.0. URL: https://CRAN.R-project.org/package=OEFPIL.