Calibration by Optimization Without Using Derivatives

Markus Lazar[1], Fakultät für Ingenieurwissenschaften
University of Applied Sciences, Rosenheim, Germany
Florian Jarre[1], Mathematisches Institut,
University of Düsseldorf, Germany,

Abstract

Applications in engineering frequently require the adjustment of certain parameters. While the mathematical laws that determine these parameters often are well understood, due to time limitations in every day industrial life, it is typically not feasible to derive an explicit computational procedure for adjusting the parameters based on some given measurement data. This paper aims at showing that in such situations, direct optimization offers a very simple approach that can be of great help. More precisely, we present a numerical implementation for the local minimization of a smooth function $f : \mathbb{R}^n \to \mathbb{R}$ subject to upper and lower bounds without relying on the knowledge of the derivative of f. In contrast to other direct optimization approaches the algorithm assumes that the function evaluations are fairly cheap and that the rounding errors associated with the function evaluations are small. As an illustration, this algorithm is applied to approximate the solution of a calibration problem arising from an engineering application.

The algorithm uses a Quasi-Newton trust region approach adjusting the trust region radius with a line search. The line search is based on a spline function which minimizes a weighted least squares sum of the jumps in its third derivative. The approximate gradients used in the Quasi-Newton approach are computed by central finite differences. A new randomized basis approach is considered to generate finite difference approximations of the gradient which also allow for a curvature correction of the Hessian in addition to the Quasi-Newton update. These concepts are combined with an active set strategy.

The implementation is public domain; numerical experiments indicate that the algorithm is well suitable for the calibration problem of measuring instruments that prompted this research. Further preliminary numerical results suggest that an approximate local minimizer of a smooth non-convex function f depending on $n \leq 300$ variables can be computed with a number of iterations that grows moderately with n.

Key words: Calibration of measuring instruments, minimization without derivatives, direct search, quadratic model.

November 16, 2015

[1]With financial support of i-for-T GmbH, Germany

1 Introduction

1.1 An Engineering Application

1.1.1 Mathematical Compensation for the Calibration of Measuring Instruments

The Bureau International des Poids et Mesures defines: The calibration of a measuring instrument is an "operation that,[...] in a first step, establishes a relation between the quantity values [...] provided by measurement standards and corresponding indications [...] and, in a second step, uses this information to establish a relation for obtaining a measurement result from an indication"[4]. Thus, the calibration procedure must be preceded by the definition of a number of variables and of a formal relation between the measurand Y and measured quantity value X – the calibrating function. The variables (termed the calibration parameters k_i with $1 \leq i \leq l$) are to compensate for a variation in the manufacturing process of the measuring instruments which is always present.

The relation can formally be written as

$$\vec{Y} = f^{cal}(\vec{X}, k_1, \ldots, k_l) + r(\delta_{stat}) \tag{1}$$

where f^{cal} is the calibrating function and r is a residual term depending on some random disturbance variables δ_{stat}. During the calibration process as many as possible true quantity values y_i are adjusted and the associated measured quantity values x_j of the measuring instrument are being recorded. The choice of the parameter values k_i aims at minimizing the norm of $r(\delta_{stat})$.

1.1.2 Determination of the calibration parameters by an optimization problem

The determination of the calibration parameters $k_1, \ldots, k_l$ is formulated as an optimization problem where the 1-norm or the 2-norm of the residuals r is being minimized. In practice one often settles for rather simple calibrating functions – sometimes even linear functions – last, not least, in order to limit the effort of mathematical modeling. Also, the reference values of the measurand y_i very often are chosen in the calibration such that the resulting measured value x_j only depends on a single calibration parameter which can be determined by solving a single (scalar) equation. On the one side, however, there is pressure to allow for larger production tolerances reducing the production costs, and on the other side, there is the demand of the market for higher precision of the measuring instruments. These demands can only be met by more sophisticated mathematical modeling that goes beyond the simple strategies for adjusting the calibration parameters as described above.

1.1.3 Modeling of additional disturbance variables

And yet another effect contributes to the higher complexity of the mathematical modeling: Every time when it is not possible to determine the true value of the measurand $\vec{Y}$ directly by a normal or by a measuring instrument of higher precision, it must be determined by some other known quantities called actuating variables $\vec{S}$. In order to derive the measurand from the known actuating variables, another functional relation g enters the mathematical model. At this

point, again, systematic disturbance variables p_i are to be taken into account. These disturbance variables are caused by the inevitable deviation of the real calibration setup from an idealized model. Analogously to the calibration parameters, also these parameters $p_1, \ldots, p_m$ must be determined and taken into account in the determination of the calibration parameters.

$$\vec{Y} = g(\vec{S}, p_1, \ldots, p_m) + \tilde{r}(\delta_{stat}) \tag{2}$$

These parameters $p_1, \ldots, p_m$ either can be eliminated by high mechanical efforts – strict production tolerances or adjustment mechanisms – or they can be determined by measuring techniques once for every calibration setup. In both cases, the effectiveness of the chosen approach must be double checked in regular intervals since changes over time will directly influence the precision of the produced measuring instruments. Thus, it is preferable to include these parameters in the mathematical model of the calibration process as well.

$$\vec{Y} = \tilde{f}^{cal}(\vec{X}, \vec{S}, p_1, \ldots, p_m, k_1, \ldots, k_l) + \hat{r}(\delta_{stat}) \tag{3}$$

When collecting a sufficient number of data sets during the calibration process, the calibration parameters $k_1, \ldots, k_l$ as well as the disturbance variables $p_1, \ldots, p_m$ can be determined by minimization of the residuals $\hat{r}$. Small changes of the calibration setup are thus compensated for and do not influence the precision of the measuring instruments – at the expense of a much more complex mathematical model. For the determination of the unknown parameters p_i and k_j a nonlinear optimization problem is to be solved[2]. The modeling effort for such nonlinear optimization problems generally cannot be conveyed to the engineers in the industrial daily routine so that this method for the calibration of measuring instruments is restricted to few specific applications in spite of its conceptual merits. This motivates the present paper where a tool is being developed that allows a simple and effective solution of optimization problems for the engineers based on mere function evaluations without requiring additional efforts of setting up partial derivatives.

1.2 A Mathematical Optimization Problem

To conform the description of the minimization algorithm with the standard optimization literature, a change of notation will be necessary. The unknown parameters of Section 1.1 will be summarized in a vector x with a total of n components. These are the variables that shall be optimally adjusted by the optimization approach. If the calibrating function is as in relation (3) then the function to be minimized will be a smooth but nonlinear least squares function of the form

$$f(x) \equiv \|\vec{Y} - \tilde{f}^{cal}(\vec{X}, \vec{S}, p_1, \ldots, p_m, k_1, \ldots, k_l)\|_2^2$$

where x comprises all unknown parameters $p_1, \ldots, p_m, k_1, \ldots, k_l$ and $\| \, . \, \|_2$ denotes the Euclidean norm. For the minimization procedure the – often very

[2]In short distance photogrammetry this approach has been used successfully for a long time. Here, the calibration parameters describe the properties of the camera which change after each change of the lens. With the aid of a large number of measured values the parameters of the outer and the inner orientation of the camera are approximated on high performance computers to determine the measured variables. Another application is the compensation of the geometric deviations of machine tools and coordinate measuring equipment.

differing – physical meanings of the components of x can be ignored as long as the functional relations specified by the calibrating function $\tilde{f}^{cal}$ are preserved. The function f to be minimized depends on the measurements $\vec{X}$ and the actuating variables $\vec{S}$, but the precise dependence on these measurements shall not be analyzed. Instead, the function f shall be minimized without knowledge of its specific structure.

More generally, let some differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ be given and some mechanism for evaluating f at a given input $x \in \mathbb{R}^n$. Let $l \in (\mathbb{R} \cup \{-\infty\})^n$ be a lower bound and $u \in (\mathbb{R} \cup \{\infty\})^n$ be an upper bound on x. We assume that $l < u$. The goal is to find an approximate solution $\tilde{x}$ to the problem

$$\text{minimize} \quad f(x) \text{ where } l \leq x \leq u. \tag{4}$$

In this paper a new implementation of a "direct search method" not relying on derivative informations of f is presented.

1.3 Minimization without using derivatives

The implementation outlined below assumes that evaluations of f are fairly cheap. For expensive functions f, other approaches such as in [1, 5, 20] are to be preferred. It is also assumed that f is at least once differentiable, but that the derivative typically is not available. The algorithm generates approximations to the first and second derivative of f. Numerical examples in Section 4 indicate that the algorithm performs reasonably well when minimizing a convex function whose second derivative is not defined at the optimal solution and performs not so well when also the first derivative is not defined at the optimal solution.

While derivative-free optimization typically generates approximate solutions that are not highly accurate, the application discussed in Section 5 requires a fairly high accuracy of the approximate solution. It turns out that the central finite differences used in this paper lead to an approximate solution that satisfies the given demand on the final accuracy in spite of the fact that direct optimization is used.

The approach presented below is based on a trust region model and is thus related to the approaches in [8, 9, 10]. It also bears similarities with other approaches as described, for example in [26, 28]. It is suitable only for local minimization. The engineering applications targeted with this approach often come with a good initial estimate of the parameters to be adjusted, and do not have local "non-global" optimizers near the starting point. Thus, a local approach is sufficient for such applications.

For global minimization other approaches such as in [20] will be more suitable, see also [2, 7, 24, 28]. For global minimization, typically the computational effort explodes when the number of unknowns increases (an observation that is sometimes referred to as curse of dimensionality). The approach followed here only aims at local minimization where the dependence on the number of unknowns appears to be much weaker. In the numerical examples of Section 4 the algorithm performs reasonably well when the number n of variables is moderate (here, $n \leq 300$).

Within Matlab or octave the approach is very easy to use – all it takes is a Matlab program for evaluating f as well as a starting point (of the appropriate dimension) – and it is public domain [21]. No compilation or other adjustments are needed; downloading the file mwd_bnd.m will suffice.

1.4 Notation

The components of a vector $x \in \mathbb{R}^n$ are denoted by x_i; the canonical unit vectors in $\mathbb{R}^n$ are denoted by e^i for $1 \leq i \leq n$. When $x \in \mathbb{R}^n$ is some vector, $\|x\|$ denotes its Euclidean norm. Inequalities such as $l \leq x \leq u$ are understood componentwise; the box of vectors x satisfying such inequalities is denoted by $[l, u]$.

Given a vector x, the diagonal matrix with diagonal entries x_i is denoted by $Diag(x)$, and given a square matrix A, the vector with the diagonal entries of A is denoted by $diag(A)$.

The gradient of a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ at some point x will be denoted by a column vector $g(x) = \nabla f(x)$, and the Hessian (if it exists) by $H(x) = \nabla^2 f(x)$.

2 A line search based on least-squares-splines

Matlab/octave or scilab is an easy-to-use environment and well suitable for modeling functions such as $\tilde{f}^{cal}$ in Section 1.1. The standard Matlab routine "fminbnd" for minimizing

a function f of one variable on some interval $[l, u]$

uses bisection and quadratic interpolation. This routine is reasonably efficient in many cases. However, in certain cases fminbnd may return the largest function value as an approximate minimizer. This lack of reliability prompted the need for a new line search.

The new approach implemented for a line search based on bisection, golden mean search (see e.g. [23]), and quadratic interpolation used a slightly higher number of function evaluations than fminbnd which is based on the same algorithms (but ignoring the end points). To compensate for this higher computational effort—and since the line search is a widely used tool—a strategy to reduce the number of function evaluations based on a spline interpolation was implemented. Possibly due to the fact that spline interpolation typically is more accurate than quadratic interpolation (see e.g. [30], Section 2.4.3), the spline interpolation reduced the number of iterations compared to the quadratic interpolation, and for some examples this new approach also compares favorably to fminbnd in terms of the number of function evaluations. The new line search always returns a point with the lowest function value that is encountered during the algorithm, in particular less or equal to the values at l and u. It is based on a new concept of spline functions as explained next.

Let $n \geq 3$ and let $n + 1$ "support points" $l = x_0 < x_1 < \ldots < x_n = u$ be given with associated function values $f(x_i)$ for $0 \leq i \leq n$. It is well known (see e.g. [30]) that there exists a unique cubic spline function $s : [l, u] \to \mathbb{R}$ satisfying $s(x_i) = f(x_i)$ for $0 \leq i \leq n$ and any one of the following four conditions:

1. "Natural spline"
$$s''(x_0) = 0 \quad \text{and} \quad s''(x_n) = 0, \tag{5}$$
2. "end slope spline" (for some given values $f'(x_0), f'(x_n)$)
$$s'(x_0) = f'(x_0) \quad \text{and} \quad s'(x_n) = f'(x_n), \tag{6}$$

3. "periodic spline" (when $f(x_0) = f(x_n)$)
$$s'(x_0) = s'(x_n) \quad \text{and} \quad s''(x_0) = s''(x_n), \tag{7}$$
4. "not-a-knot spline"
$$s'''(x_1 - 0) = s'''(x_1 + 0) \quad \text{and} \quad s'''(x_{n-1} - 0) = s'''(x_{n-1} + 0). \tag{8}$$

The computation of the spline function typically is carried out in $O(n)$ arithmetic operations solving a linear system with a tridiagonal structure (possibly with a minor perturbation of the tridiagonal structure), see e.g.[30], Section 2.4, Algorithm 2.4.2.15.

Below, a somewhat simpler approach is proposed using $O(n)$ arithmetic operations as well, and allowing for a more flexible choice of additional conditions:

- First, construct an interpolating spline s_0 with the additional conditions
$$s_0'(x_0) = 0 \quad \text{and} \quad s_0''(x_0) = 0, \tag{9}$$
in place of one of the above four conditions, i.e.:

  For $i = 0, 1, \ldots, n-1$ do:

  1. Given $s_0'(x_i), s_0''(x_i)$, and $s_0(x_i) = f(x_i)$ define s_0 for $x \in [x_i, x_{i+1}]$ via
  $$s_0(x) := s_0(x_i) + s_0'(x_i)(x - x_i) + \frac{1}{2}s_0''(x_i)(x - x_i)^2 + \gamma_i(x - x_i)^3$$
  where
  $$\gamma_i := \frac{f(x_{i+1}) - \big(s_0(x_i) + s_0'(x_i)(x_{i+1} - x_i) + \frac{1}{2}s_0''(x_i)(x_{i+1} - x_i)^2\big)}{(x_{i+1} - x_i)^3}.$$
  2. Compute
  $$s_0'(x_{i+1}) = s_0'(x_i) + s_0''(x_i)(x_{i+1} - x_i) + 3\gamma_i(x_{i+1} - x_i)^2$$
  and
  $$s_0''(x_{i+1}) = s_0''(x_i) + 6\gamma_i(x_{i+1} - x_i).$$

  It is easy to see that s_0 generated above is indeed an interpolating cubic spline function that satisfies (9).

- Then (in the same fashion as above for s_0) generate spline functions s_1 and s_2 with $s_1(x_i) = 0$ and $s_2(x_i) = 0$ for $0 \le i \le n$ and
$$s_1'(x_0) = 1, \quad s_1''(x_0) = 0, \quad \text{and} \quad s_2'(x_0) = 0, \quad s_2''(x_0) = 1.$$
By construction, s_1 and s_2 are linearly independent. (In a numerical implementation, s_0, s_1, s_2 can be evaluated simultaneously to reduce the overall numerical effort.)

- Observe that any interpolating spline function s has the form
$$s(x) = s_0(x) + \alpha s_1(x) + \beta s_2(x)$$
for some fixed values $\alpha, \beta \in \mathbb{R}$.

The last observation motivates the following approach:

- Compute s_0, s_1, s_2 and solve a 2×2 system of linear equations for the coefficients α, β to satisfy any of the conditions (5) – (8).

2.1 Additional conditions

If $f'(x_0)$ and $f'(x_n)$ are not available, and if f is not known to be a periodic function, conditions (5) or (8) are often chosen to define an interpolating spline s.

Even if $\max_{1\le i\le n} x_{i-1} - x_i \to 0$, the choice (5) prevents that s'' will converge to f'' on $[l,u]$ unless $f''(x_0) = 0$ and $f''(x_n) = 0$. Thus, generalizations of the choice (8) will be considered in the sequel.

Given some spline function s define the "jump" of s''' at x_i by

$$\eta_s(x_i) := \lim_{x\to x_i,\ x<x_i} s'''(x) - \lim_{x\to x_i,\ x>x_i} s'''(x).$$

Alternative choices for determining s would then be

$$\text{minimize} \sum_{1\le i\le n-1} w_i \eta_s(x_i)^2 \tag{10}$$

or

$$\text{minimize} \max_{1\le i\le n-1} w_i |\eta_s(x_i)| \tag{11}$$

for some nonnegative weights w_i, e.g.

$$w_i := \frac{2}{x_{i-1} - x_{i+1}} \qquad (1 \le i \le n-1). \tag{12}$$

These weights account for the aim that large changes of s''' in between short intervals are penalized more than large changes between longer intervals. Note that both (10) and (11) solve (8) for the choice

$$w_1 = w_{n-1} = 1 \quad \text{and} \quad w_2 = w_3 = \ldots = w_{n-2} = 0 \tag{13}$$

(namely forcing $\eta_s(x_1) = \eta_s(x_{n-1}) = 0$). Numerical examples suggest that the choice (12) over the standard choice (13) tends to produce better approximations of analytic functions for $x \in (x_1, x_{n-1})$ but not necessarily near the end points. The line search in [21] is set up such that the minimizer of the spline is located in (x_1, x_{n-1}), and thus, the choice (10), (12) is used there.

Note that given s_0, s_1, s_2 as in (9), the solution of (10) is the solution of a 2×2 system of linear equations that can be set up (and solved) with $O(n)$ arithmetic operations. It turns out however, that the linear systems for solving (10), in spite of having only two unknowns, tend to be very ill-conditioned, in particular, when the support points x_i are not evenly distributed (but accumulate near a local minimizer of f). To reduce the rounding errors, the solution of the systems is carried out with orthogonal transformations, and given the solution, the interpolating spline is recomputed based on the values of s' and s'' at x_0.

The line search then is carried out as follows: By golden mean search an interval is identified containing a local minimizer. f is interpolated within this interval by a least-squares-spline (10). The minimizer of this spline is taken as next point at which f is being evaluated. Some tedious but straightforward safeguards ensure the convergence.

3 A trust region Quasi-Newton approach

In this section we describe an iterative algorithm for approximating a local minimizer of

$$\text{a smooth function } f : \mathbb{R}^n \to \mathbb{R} \text{ on some box } [l, u]$$

where $l \in (\mathbb{R} \cup \{-\infty\})^n$ and $u \in (\mathbb{R} \cup \{\infty\})^n$ are given data.

The algorithm relies on the availability of approximate gradients, the numerical evaluation of which is discussed in Section 3.3. The gradient information will also be used for a Quasi-Newton update of the Hessian. Generally, because the Quasi-Newton updates are numerically cheap ($O(n^2)$), they are applied in the context of numerically cheap search steps. Here, the situation is somewhat different: The evaluation of the approximate gradient is costly (typically $O(n^3)$ or higher) motivating a computationally more expensive search step to extract large progress based on the available gradient and Hessian information.

3.1 Initialization

The algorithm uses central finite difference approximations of the first derivative of f. The accuracy of the finite difference approximations and the performance of the algorithm below crucially depends on the accuracy with which f is being evaluated. The approximate error in the evaluation of f will be denoted by err.

To estimate err, the evaluation of the first gradient (at x^0 and with a randomized basis) is carried out with a one-sided finite difference for a step length $\epsilon^{2/3}$ where ϵ is the machine precision. Then, approximate $err \geq \epsilon$ by

$$err \approx \max_{1 \leq i \leq n} |\frac{1}{2}(f(x^0) + f(x^0 + 2\epsilon^{1/2}u^i)) - f(x^0 + \epsilon^{1/2}u^i)|,$$

where the first two terms in the max-term of the right-hand side are stored from the evaluation of the first gradient. In the absence of rounding errors in the evaluation of f, this estimate of err is of the order at most $\epsilon^{4/3}M$ where M is an upper bound for the norm of $\nabla^2 f$ near the point x^0. Hence, when $M \leq \epsilon^{-1/3}$ the above approximation will yield a lower bound for the evaluation error err of f.

The above estimate of err is used to define the step length for the finite difference approximations to the gradient and the Hessian of f, defining

$$\delta := 0.1\, err^{1/3}. \tag{14}$$

To motivate the definition (14) assume for the moment that the third derivative of f exists and is continuous. Assume further that $n = 1$ and denote the finite difference approximation of $f'(x)$ by g. Then, subtracting the Taylor expansions up to third order for the points $f(x \pm \delta)$ shows that estimating $f'(x)$ by central differences through the points $f(x \pm \delta)$ results in a discretization error of

$$|f'(x) - g| \leq \delta^2 \frac{|f'''(\xi)|}{6} + \frac{err}{\delta} = err^{2/3}(\frac{|f'''(\xi)|}{600} + 10).$$

The choice of the factor 0.1 in (14) is somewhat arbitrary; it implies that the linearization error dominates the error caused by the inaccuracy in the function values when $\|f'''(\xi)\| > 6000$. For $n > 1$ the partial derivatives of f are the first

derivatives of the function $t \mapsto f(x + te^i)$, so that the above estimate remains valid in the form

$$|g(x)_i - g_i| \leq err^{2/3}(\frac{|D^3 f(x + \xi e^i)[e^i, e^i, e^i]|}{600} + 10).$$

A similar consideration for $n = 1$ and for four times continuously differentiable f shows that the approximation of $f''(x)$ (or of β_i in Section 3.4) has an accuracy of at least $\frac{2|f''''(\xi)|}{300} err^{2/3} + 400 err^{1/3}$. Clearly, the update of β is meaningful only, as long as the norm of the correction $\Delta\beta$ in Section 3.4 is larger than this discretization error. Note further that the above term $400 err^{1/3}$ is correct only, when the error err in the evaluation of f is known; if a higher precision is assumed for the definition of δ than the one that is actually present, this term can explode. (A wrong estimate of err also influences the accuracy of the first derivative, but to a lesser extent.) For the limited number of numerical experiments carried out so far, the estimate of err near x^0 as detailed above was sufficient to allow for overall convergence.

3.1.1 Bounds

If f is not defined for $x \notin [l, u]$, the bounds l and u will be replaced by $l + \delta$ and $u - \delta$ for the main phase of the algorithm, and only in the last step, when no further gradient approximations are to be evaluated, the bounds l and u are set to their initial values. (This is the default of the algorithm in [21].)

Thus, the initial point and all iterates (except the last one) are assumed to satisfy the strengthened bounds so that central differences about the iterates are well defined. In particular, we assume that $l_i < u_i - 2\delta$ for all i; else, the variable x_i is fixed at $x_i = (l_i + u_i)/2$.

3.2 A trust region step with curvilinear search

Assume that a current iterate $x = x^k$ with $l \leq x \leq u$ is given along with estimates g for $g(x)$ and H for $H(x)$. Let $\delta > 0$ be the step length used for the finite difference approximation to the gradient defined in Section 3.1.

Active constraints at x are identified as the set

$$J := \{i \mid x_i \leq l_i + \delta,\ g_i > 0\} \cup \{i \mid x_i \geq u_i - \delta,\ g_i < 0\}.$$

We change x by replacing all x_i for $i \in J$ with the bounds l_i if $g_i > 0$ and with u_i if $g_i < 0$. We then keep x_j for $j \in J$ fixed and do a line search varying the values x_k for the remaining indices $k \in K := \{1, \ldots, n\} \backslash J$.

These variables are determined by a trust region subproblem: Let H_{KK} be the principal submatrix of H associated with K. Compute the eigenvalue decomposition $H_{KK} = VDV^T$ and set $d_{min} := \min_{k \in K} D_{k,k}$. For $t \in (0, 1)$ set

$$x_K(t) := \Pi_{[l_K, u_K]} \left(x_K - V \left(D + (\frac{1 - t}{t} - d_{min})I \right)^{-1} V^T g_K \right),$$

where $\Pi_{[l_K, u_K]}$ denotes the projection onto the box $[l_K, u_K]$,

$$\Pi_{[l,u]}(x) = \max\{l, \min\{u, x\}\}.$$

If $l_i = -\infty$ and $u_i = \infty$ for all i, then the points $x(t)$ are given by

$$x(t) = x - (H + (\frac{1-t}{t} - d_{min})I)^{-1} g, \tag{15}$$

where the parameterization $\hat{t} := \frac{1-t}{t} - d_{min}$ is chosen to restrict t to $(0, 1)$ and such that for small $t > 0$ it follows

$$x(t) - x \approx -tg,$$

i.e. $x(t)$ approximates the steepest descent path for small $t > 0$. (In the implementation in [21], a minor modification of the above definition of $x(t)$ is used to ensure that $x(t)$ is well defined on the closed interval $[0, 1]$.)

Note that the points $x(t)$ in (15) are optimal solutions of trust region subproblems

$$\text{minimize } \; g^T s + \frac{1}{2} s^T H s \;\; | \;\; s_J = x_J, \;\; \|x - s\| \leq \delta$$

for some trust region radius $\delta = \delta(t)$ depending on t. (This is true for $d_{min} \leq 0$; nevertheless, since H is only an approximation, we allow for positive values of d_{min} to enable longer steps – secured by a line search.) With the exception of the "hard case" addressed in Section 3.3 we also have $\delta(t) \to \infty$ when $t \to 1$.

The line search of Section 2 minimizing $f(x(t))$ for $t \in (0, 1)$ is then applied to find an approximate minimizer x^+ along the curve $x(t)$. In our examples, typically, about 10 to 20 function evaluations are needed for a line search.

Search steps of the form (15) have been considered before, for example, in [15], see also [6]. Here, a straightforward active set modification to allow for bound constraints is included. The line search allows for long steps before reevaluating the gradient, and thus, it is particularly rewarding for problems with $n \geq 10$ variables.

3.3 Gradient approximation and randomized bases

To estimate the gradient at some iterate $x = x^k$ the following randomized bases can be used: Randomly (and uniformly) generate an orthonormal basis $U = (u^1, \ldots, u^n)$ of $\mathbb{R}^n$ and set $dt = \delta + \epsilon^{2/3}\|x\|$ where ϵ is the machine precision and δ is the basic step length for the finite difference as defined in Section 3.1.

Let $fval_1$ be the vector with entries $fval_1(i) = \tilde{f}(x - dt\, u^i)$ and $fval_2$ be the vector with entries $fval_2(i) = \tilde{f}(x + dt\, u^i)$. Then,

$$g(x) \approx g := U(fval_2 - fval_1)/(2dt).$$

This approximation coincides with the standard central finite difference approximation when $U = I$ is the identity matrix. As will be seen next, the randomized bases allow a somewhat more balanced update for the Hessian than the choice $U = I$ corresponding to the standard finite difference star.

The computational cost of generating a random unitary matrix U is of order $O(n^3)$. If each evaluation of f for the finite difference approximation of the gradient takes at least $O(n^2)$ operations (this is the case, for example, for quadratic functions with a dense Hessian) then the cost of computing U is of at most the same order as the cost of the gradient approximation. Likewise, also the cost of the line search is not much higher than the cost of the gradient approximation.

For a randomly chosen basis, it is rather unlikely that the finite difference star will return an approximation g to the gradient that results in the "hard case" of the trust region subproblem – where g is exactly perpendicular to the eigenspace associated with the smallest eigenvalue of H. Thus, the curvilinear search in (15) is likely to satisfy $\|x(t)\| \to \infty$ for $t \to 1$.

3.4 Hessian updates

The gradient information of the line search step can be used in two complementing ways to update the approximation to the Hessian: First, a low rank update of the Hessian is implemented. This update can be corrected by a curvature correction and the combined update of the Hessian can be used in the next trust region line search step. Thus, the needs of the updates are twofold:

For the trust region line search step there is no need to assume that the approximate Hessian be positive definite. On the other hand, the trust region is based on the Euclidean norm – giving up on the idea of affine invariance. Thus, also the low rank update of the Hessian is carried out with respect to the Euclidean norm, preserving symmetry of the Hessian, but not necessarily positive definiteness. This leads to the choice of the PSB (Powell symmetric Broyden) update, [27][3]. This update is followed by a curvature correction which also relies on the Euclidean norm.

1. Let x denote the starting point of the curvilinear search in Section 3.3 and let x^+ be the approximate minimizer along this curve. Set $\Delta x := x^+ - x$. Another finite difference approximation g^+ to $g(x^+)$ is evaluated next. The estimate H for $H(x)$ is updated to an estimate H^+ for $H(x^+)$ as follows:

   If $\|\Delta x\|^2 > dt^2(1+\sqrt{\epsilon}\|x\|)$ the difference of the gradients g and g^+ at both points is used for a Quasi-Newton update based on the PSB formula. Let $\Delta g := g^+ - g$, $\Delta x = x^+ - x$, and

   $$\Delta H := \frac{(\Delta g - H\Delta x)\Delta x^T + \Delta x(\Delta g - H\Delta x)^T}{\Delta x^T \Delta x} - \frac{(\Delta g - H\Delta x)^T \Delta x}{(\Delta x^T \Delta x)^2} \Delta x \Delta x^T.$$

   The PSB update is then given by setting

   $$H := H + \Delta H.$$

2. This update can be corrected further by a least squares update of the Hessian based on central finite differences: Let β be the vector with the central difference approximations to the second derivative of f along the axes given by U, i.e.

   $$\beta_i = \frac{fval_1(i) + fval_2(i) - 2\tilde{f}(x^k)}{dt^2}$$

[3] In [27] this update is defined as the limit when iterating the Broyden-rank-1-update followed by a symmetrization. [27] also includes numerical examples and convergence properties. In addition, this update minimizes the Frobenius-norm of the correction subject to the Quasi-Newton condition and the symmetry condition, see e.g. [22], Theorem 6.6.10 and (6.6.18). The minimum norm property motivates the choice of this update for the Euclidean norm trust region problem.

where $fval_1$ and $fval_2$ are as in Section 3.3. If U was the identity matrix, then β_i would provide the central difference approximation to the diagonal elements $D^2 f(x^k)_{i,i}$. For general orthogonal U let $\Delta\beta := \beta - diag(U^T H U)$ denote the difference of the above central difference approximation and the corresponding values of the current matrix H. Then,

$$H^+ := H + U\ Diag(\Delta\beta)\ U^T;$$

provides a least squares correction for H.

By using randomized bases U, the updates are not restricted to diagonal elements only but correct all entries of the Hessian. (In the numerical examples in Section 4, the Hessian of f at the final iterate was approximated to about 3 significant digits even when using only the curvature correction without the PSB update.) This second update following the PSB update is not a low rank update and seems not to have been considered before. It will be called "curvature update" in the sequel.

Summarizing, a cost intensive but quite accurate approximation of the gradient by central differences and a combination of two different updates of the approximation to the Hessian matrix is intended to allow for a reliable solution of smooth nonlinear bound constrained optimization problems at a reasonable accuracy. In some preliminary numerical experiments it was indeed possible to observe with several examples that not only an approximate solution of the optimization problem was found, but also the Hessian approximation generated at the end of the algorithm often was quite accurate, in particular, when the number of iterations carried out by the algorithm was rather high.

4 Numerical results

4.1 Test problems

In this section we report on some preliminary numerical examples that were chosen to provide some intuition about the rate of convergence of the algorithm. The engineering application that triggered this paper is presented in the next section. For the interpretation of the numerical results we remind the reader that the algorithm MWD (Minimization Without (using) Derivatives, [21]) in this paper is intended for local minimization; no globalization strategy is included.

1. Some test functions of just two variables can be found at [3]. In order to avoid random effects that result from the approximation of different local minimizers, for a first test, some smooth test functions from [3] were chosen that only have few local minimizers namely: The Beale Function, the Goldstein-Price Function, and the McCormick Function. Their definitions are

$$\begin{aligned} f_B(x,y) &= (1.5 - x + xy)^2 + \left(2.25 - x + xy^2\right)^2, \\ f_{GP}(x,y) &= \left(1 + (x+y+1)^2 \left(19 - 14x + 3x^2 - 14y + 6xy + 3y^2\right)\right) \cdot \\ &\quad \left(30 + (2x-3y)^2 \left(18 - 32x + 12x^2 + 48y - 36xy + 27y^2\right)\right), \\ f_{MC}(x,y) &= \sin(x+y) + (x-y)^2 - \frac{3}{2}x + \frac{5}{2}y + 2.91322295498103777. \end{aligned}$$

The constant terms in above functions are modified such that the minimum value is zero. The starting points for MWD were $(0,0)^T$ for f_B, $\frac{1}{2}(1,-3)^T$ for f_{GP}, and $(1,-1)^T$ for f_{MC}.

2. The next set of test functions was intended to obtain some insight of the dependence of the approach on the dimension with regard to test problems of the same nature. The Styblinski-Tang Function and the Rosenbrock Function in [3] are smooth functions with $n \geq 2$ variables and (when $n \geq 3$) with more than one local minimizer. They are complemented below by the Chained Rosenbrock Function, which has just one minimizer. Their definitions are

$$\begin{aligned} f_{ST}(x) &= \frac{\sum_{i=1}^n x_i^4 - 16x_i^2 + 5x_i}{2}, \\ f_R(x) &= \sum_{i=2}^n (1 - x_{i-1})^2 + 100(x_i - x_{i-1}^2)^2, \\ f_{CR}(x) &= (x_1 - 1)^2 + 100 \sum_{i=2}^n (x_i - x_{i-1}^2)^2. \end{aligned}$$

f_{ST} is a separable function making its local minimization trivial if the starting point is a multiple of the all ones-vector. Thus, for this function the starting point was chosen with uniformly distributed components in $(-\frac{3}{2}, \frac{3}{2})^T$. To avoid convergence to the minimizer of f_R close to $(-1, 1, \ldots, 1)^T$, the starting point for f_R and also for f_{CR} was set to zero.

3. The effect of the various updates of the Hessian matrix was compared using the next test set consisting of the functions f_R, f_{CR} from above, and a modified Styblinski-Tang Function that has only one minimizer (and a region where the function is rather flat) and that has a dense Hessian. (Clearly, for a diagonal Hessian the randomized bases will not be meaningful.)

$$f_{STmod}(x) := \frac{\sum_{i=1}^n (Mx)_i^4 - 16(Mx)_i^2 + 35(Mx)_i}{2},$$

where $(Mx)_i$ is the i-th component of Mx and M is the matrix

$$M = \begin{bmatrix} n+1 & n-1 & n-2 & \ldots & 1 \\ 0 & n & n-2 & & 1 \\ 0 & 0 & n-1 & & 1 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 2 \end{bmatrix}.$$

Here the dimension was fixed to 10 and the starting point was set to zero for all examples.

4. Next, an augmented primal-dual function for a random linear program

$$\min\{c^T x \;\mid\; Ax = b,\; x \geq 0\}$$

is tested. This function is generated as follows: Randomly generate complementary vectors $\bar{x},\ \bar{s}$ whose nonzero entries are uniformly distributed

in $[0,1]$. Here, $\bar{x} \geq 0$ is constructed with m nonzero entries and $\bar{s} \geq 0$ having $n-m$ nonzero entries. Also define $\bar{y} \in \{-1,1\}$ randomly. Finally, generate a matrix $A \in \mathbb{R}^{n\times m}$ with a standard normal distribution. Set $b := A\bar{x}$, $c := A^T\bar{y} + \bar{s}$ and define the function $f : \mathbb{R}^{2n+m} \to \mathbb{R}$ by partitioning the vector $z \in \mathbb{R}^{2n+m}$ as $z = (x; y; s)$ and setting

$$f((x;y;s)) := \|Ax-b\|^2 + \|A^Ty+s-c\|^2 + (c^Tx-b^Ty)^2 + \|x_-\|^2 + \|s_-\|^2$$

where the i-th element of x_- is given by $\min\{0, x_i\}$. (Likewise for s_-)

Thus, f is minimized at the primal-dual solution $\bar{x}, \bar{y}, \bar{s}$ of the linear program and takes the value zero at this point. Note that f has a discontinuous second derivative; the algorithm MWD based on generating Hessian approximations is thus expected to converge less rapidly. It is clear that solving a linear program by minimizing the above function f is prohibitive; each function evaluation provides very little information about the linear program – just a single scalar number. Nevertheless, this type of function provides some nontrivial test problems with known optimal solutions.

5. Finally, a non-smooth penalty function for the same linear program as in the previous example is tested. Here,

$$f(x) := c^Tx + \sqrt{n}\,\|Ax-b\| + \sqrt{n}\|x_-\|_1 - c^T\bar{x}$$

or

$$f(x) := c^Tx + \sqrt{n}\,\|Ax-b\| - c^T\bar{x}.$$

Note that by the construction of the dual variables, this is an exact penalty function on $\mathbb{R}^n$ respectively on the domain $\{x \mid x \geq 0\}$ with minimizer $\bar{x}$ and $f(\bar{x}) = 0$.

For this function, the derivative is not defined at the optimal solution, nevertheless, MWD could generate a reduction of the function value, but stopped due to slow progress before reducing the function value by a factor of 10^{-4}.

4.2 Test results

The trust region algorithm has been implemented in Matlab and was tested with the examples above.

1. For the three 2-dimensional test problems, MWD generated to following results:

problem	$f(x^{fin})$	$\|g(x^{fin})\|$	iter.	#fn.eval.	#fn.eval. (l.s)
f_B	2.7e-26	1.4e-10	11	194	12.6
f_{GP}	5.7e-14	3.2e-7	8	164	15.5
f_{MC}	8.8e-16	8.1e-9	7	103	9.7

Here, x^{fin} is the output generated by MWD, $\|g(x^{fin})\|$ the norm of the finite difference approximation of the final gradient, #fn.eval. is the overall number of function evaluations, and #fn.eval. (l.s) is the average number of function evaluations during a line search.

2. For the three n-dimensional test problems in [3], MWD generated the following results: For each dimension, the norm $\|g(x^{fin})\|$ is listed, followed by the overall number of iterations and the average number of function evaluations during a line search. (Apart from the line search, exactly $2n + 1$ function evaluations were performed at each iteration.)

n	f_{ST}	f_R	f_{CR}
2	5.2e-7 / 9 / 12.2	1.7e-9 / 20 / 13.0	1.7e-9 / 20 / 13.0
4	7.3e-8 / 13 / 13.2	1.1e-6 / 47 / 12.3	1.7e-8 / 55 / 14.1
8	1.8e-6 / 19 / 14.2	8.6e-9 / 77 / 12.5	2.5e-8 / 310 / 16.7
16	3.6e-6 / 22 / 11.5	1.0e-9 / 135 / 12.1	4.7e-5 / 1.0e4 / 13.5
32	2.2e-6 / 23 / 11.5	3.2e-9 / 239 / 11.9	6.0e-6 / 1.0e4 / 14.4
64	3.2e-6 / 25 / 12.3	2.9e-9 / 433 / 11.9	1.5e-4 / 1.0e4 / 13.1
128	1.4e-5 / 28 / 16.5	1.1e-8 / 835 / 11.8	1.5e-3 / 1.0e4 / 12.8
256	2.2e-5 / 28 / 16.5	5.5e-8 / 1612 / 11.8	1.9e-3 / 1.0e4 / 11.5

In particular, for the function f_{CR}, function values close to zero do not imply that the argument is close to the optimal solution; this function has a very flat valley, along which the iterates of descent methods typically converge rather slowly to the optimal solution.

3. The comparison of the different updates returned the following results where for each test run the values $f(x^{fin}) - f(x^{opt})$ and the number of iterations are listed:

Update	f_{STmod}	f_R	f_{CR}
zero Hessian	3.4e-4 / 461	3.7e-5 / 7192	4.8e-3 /10000
curv. only, U=I	4.4e-5 / 123	1.1e-5 / 4159	3.8e-3 / 10000
curv. only, U=rand$_{best}$	0 / 49	9.5e-23/ 95	8.9e-18/ 1719
curv. only, U=rand$_{worst}$	1.1e-11 / 73	3.0e-15 / 129	1.7e-14/ 1853
PSB only	4.5e-13 / 23	1e-12 / 93	6.4e-16 / 738
PSB & curv., U=I	1.9e-8 / 34	1.1e-15 / 51	6.2e-15 / 739
PSB & curv., U=rand$_{best}$	0 / 28	3.0e-21 / 56	1.6e-19 / 786
PSB & curv., U=rand$_{worst}$	2.3e-11 / 39	6.4e-16 / 67	1.8e-16 / 824

Above, U denotes the basis for the finite difference approximation. The row "zero Hessian" refers to the steepest descent method with (nearly) exact line search, the row "PSB only" refers to a trust region Quasi-Newton-approach with PSB update and with a trust region radius determined by a line search. For the randomized bases, 100 test runs were performed each, and the best and the worst results over the 100 test runs are reported. While the curvature update by itself (without PSB update) results in a significant speedup compared to the steepest descent algorithm, in particular, when the concept of randomized bases is used, the combination of both updates does not necessarily improve over the plain PSB update. In particular, for the example f_{STmod} the error for a function evaluation was higher by three decimal digits than for the other two examples, making the curvature update of the Hessian less effective (see the discussion in Section 3.1). The plain PSB update is set as default value in [21].

4. Below, column 1 lists the dimensions $N = 2n + m$ of the input for an augmented primal-dual function f for a linear program, column 2 lists the number of iterations, and column 3 lists the final function values divided by the initial values (taking into account that the optimal value is zero). The starting point was chosen from an independent uniform distribution, all entries in $[0, 1]$.

N	iterations	reduction of obj. function
10	78	4.4e-14
20	142	3.7e-13
40	106	2.3e-10
80	137	6.3e-10
160	316	4.1e-8
320	1544	2.1e-8

5. The last example of a non-smooth penalty function for linear programs (violating the smoothness assumptions of the algorithm) generated the following results.

	unconstrained		$x \geq 0$	
n	iterations	reduction, obj. fn.	iterations	reduction, obj. fn.
4	28	0.0022	23	0.0052
8	41	0.00085	50	0.0000001
16	57	0.0011	86	0.00073
32	205	0.0018	127	0.0022
64	279	0.0037	171	0.0040
128	259	0.0012	253	0.0011
256	933	0.00086	287	0.00055

Here, the algorithm did not use an excessive number of iterations such as for the Chained Rosenbrock function above, but terminated due to slow progress after some number of successful initial iterations. (The dimension $n = 128$ above refers to the same LP as $N = 2n+m = 320$ in the example of the augmented primal-dual function above.)

4.3 Other direct solvers

A comparison of currently available direct solvers (optimization solvers not using derivatives) can be found, for example, in [29], documenting a recently revived interest in this subject. The aims of the codes compared in [29] differ from the aims followed in the present paper; in particular, the codes in [29] consider noisy function evaluations, nonsmoothness, globalization strategies in the presence of local, non-global minimizers, or the desire of generating a quick approximation to a minimizer depending on only few (but expensive) function evaluations.

It is evident that the more general frameworks of the codes compared in [29] come at the prize of a somewhat weaker performance with respect to the criteria followed in this paper: high accuracy for problems depending on a moderate (but not necessarily tiny) number of parameters. Therefore, the following comparison concentrating just on the aspects relevant for the present paper is not intended

as evaluation of the quality of any of the codes. It is only to relate the final accuracy and the number of variables in the present code with other codes, and it is limited to a restricted class of problems, namely smooth local minimization.

For a more complete comparison we refer to [29]. ([29] does not include the code of the present paper which was written after [29] was published, but it is clear that the code of the present paper cannot compare favorably to other codes in the general setting of [29]; the code of present paper is not designed for such settings.)

Below, a brief comparison is given with those Matlab codes in [29] that are freely available, namely GLOBAL [11, 12], SID-PSM [13, 14], cmaes [18, 19], snobfit [20], and PSwarmM [31]. For the comparison, two functions without local, non-global minimizers are chosen, functions that can be formulated in terms of an arbitrary number of variables, namely the chained Rosenbrock function f_{CR} and the modified Styblinski-Tang function f_{STmod}. The superscript for the function is used to denote the number of variables, i.e. f^{10}_{STmod}, for example, stands for the modified Styblinski-Tang function depending on 10 variables. For each of the codes, the time in seconds, the number of function evaluations, and the final function value are listed. (The times refer to a desktop with eight Intel(R) Core(TM) i7-4770 CPU (3.40GHz).)

solver	f^2_{CR}	f^{10}_{CR}	f^{50}_{CR}
GLOBAL	0.078 / 3220 / 4.9e-9	0.79 / 30947 / 1.8e-3	0.67 / 20007 / 3.3
SID-PSM	0.024 / 120 / 0	0.19 / 392 / 0	20.4 / 1752 / 0
cmaes	1.2 / 1063 / 1.9e-17	9.5 / 64462 / 1.0e-14	604 / 3102177 / 1.2e-9
snobfit	3.1 / 504 / 1.0e-6	153 / 5760 / 9.8e-2	234 / 2912 / 1.7e+3
PSwarm	0.34 / 2012 / 2.9e-4	1.9 / 10007 / 7.1e-2	47.2 / 50091 / 2.6e-2
MWD	0.08 / 352 / 2.6e-21	4.2 / 29502 / 7.5e-15	26.4 / 350217 / 1.1e-7

solver	f^2_{STmod}	f^{10}_{STmod}	f^{50}_{STmod}
GLOBAL	0.033 / 466 / 3.9e-10	1.1 / 18850 / 2.92	1.47 / 19514 / 1.1e+6
SID-PSM	0.050 / 147 / 1.1e-13	5.2 / 3084 / 2.1e-6	19300 / 2797849 / 9.2e-5
cmaes	1.2 / 515 / 1.1e-13	2.5/ 5218 / 4.5e-13	19.2 / 69939 / 5.5e-12
snobfit	3.2 / 504 / 8.1e-8	8.0 / 880 / 1.2e+3	257 / 3024 / 1.6e+7
PSwarm	0.36 / 2001 / 1.7e-9	1.9 / 10013 / 1.7e-7	20 / 50012 / 0.26
MWD	0.046 / 164 / 1.1e-13	0.14 / 879 / 2.3e-13	1.1 / 11832 / 3.6e-12

In all cases, the domain was the box $[-2, 2]^n$ with initial vector 0 (if an initial vector is required – and for PSwarm, also a second initial value, the vector $(-1, 1, \ldots, 1)^T$ was provided. The same initial points were also used for the modified Styblinski-Tang function, where the optimizer lies near the point $(0, \ldots, 0, -0.5, -1.6)^T$.) For snobfit and PSwarm, the parameters were modified from the default to avoid an early termination; in snobfit, the parameter nstop was increased from 5 to 50, and in PSwarm, the upper function evaluation limit was increased to $1000n$. For SID-PSM, the default for the run with f^{50}_{STmod} was reset to "No search step and default poll order". To interpret the results of snobfit, e.g. for dimension 50, we note that the average function value of f_{CR} on $[-2, 2]^{50}$ is about 2e+4 and the average function value of f_{STmod} on $[-2, 2]^{50}$ is about 1e+10. This implies that values like 1.7e+3 or 1.6e+7, correspond to a reduction of the function value by a factor of 10 to 1000 over the average function value. Comparing such reduction – obtained with a global search and with rather few function evaluations – with a local solver such as MWD is

misleading. Similar reservations apply to a comparison of GLOBAL and MWD Thus, we do not report any further comparison.

As the stopping criteria are difficult to align, the interpretation of the above tables is not conclusive. It seems though that cmaes returns results that are comparable with those of MWD, also for moderate dimensions.

4.4 Testing the worst case behavior for the cg-method

Consider the conjugate gradient method (cg-method) for solving $Ax = b$ with a symmetric positive definite matrix A starting with $x^0 := 0$. Denote the solution by $x^* := A^{-1}b$, set $\gamma := (x^*)^T Ax^*$, and let the A-norm $\| \ . \ \|_A$ be given by $\| \ u \ \|_A^2 := u^T Au$. It is well known that the k-th conjugate gradient iterate x^k minimizes the function

$$f: \ x \mapsto x^T Ax - 2b^T x + \gamma \equiv \|x - x^*\|_A^2$$

on the Krylov-subspace $K_k := \text{span}\{b, Ab, \ldots, A^{k-1}b\}$ see e.g. [17], Lemma 10.2.1, Theorem 10.2.2.

Assume that $0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ are the eigenvalues of A and $\kappa := \lambda_n/\lambda_1$ is its condition number. Then, the improvement of the k-th conjugate gradient iterate x^k over the initial iterate $x^0 = 0$ in terms of the function value f is bounded by

$$\left(\frac{f(x^k)}{\gamma}\right)^{1/2} \leq 2\left[\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k + \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k\right]^{-1}. \tag{16}$$

It is known that this bound is sharp, i.e. for any k and $\kappa > 1$ there exist A, b for which this bound is attained, see e.g. [25].

Below, the knowledge of the bound is used to construct test problems with a known optimal value. More precisely, for fixed k, the left hand side of (16) is considered as a function of A and b, and the left hand side of (16) is maximized by varying A and b. For the numerical examples, it is assumed (without loss of generality) that A is a diagonal matrix so that the left hand side of (16) is a function of λ and b. For the test λ_1 is fixed to $\lambda_1 := 1$ and $\lambda_n := \kappa$. Moreover, without loss of generality, one may fix $b_1 := 1$. Thus, $f(x^k)/\gamma$ depends on $2n - 3$ parameters $\lambda_2 \ldots \lambda_{n-1} \in [1, \kappa]$ and $b_2, \ldots, b_n$; in particular, also x^k depends on these parameters. Formally, the unknowns $\lambda_2, \ldots, \lambda_{n-1}, b_2, \ldots, b_n$ are summarized in some vector z, and a function $\hat{f}: z \mapsto \hat{f}(z) := f(x^k)/\gamma$ is minimized via MWD, where the value $f(x^k)$ is evaluated by executing k steps of the cg-method starting with $x^0 = 0$.

As a starting point z^0 for all examples the values $\lambda_i \equiv 1 + \frac{(i-1)(\kappa-1)}{n-1}$ and $b_i \equiv 1$ are chosen.

Below, the left hand side of (16) i.e. the reduction of the A-norm after k cg-steps for the initial data λ, b is listed (called red_0), the upper bound for this reduction given by the right hand side of (16) (called red_{max}) followed by the distance $dist := red_{max} - red_{MWD}$ where $red_{MWD} \leq red_{max}$ denotes the reduction of the A-norm after k cg-steps for the data λ, b generated by MWD. In addition, the number of iterations and the number of function evaluations used in MWD are listed. The last column gives the time in seconds for the run of MWD on a MacBookPro from 2012 (Intel(R) Core(TM) i5 CPU, M 540, 2.53GHz). The first table refers to $n = 10$, $\kappa = 10$, and values $k = 1, \ldots, 10$.

k	red_0	red_{max}	$dist$	# it.	# f-eval.	time
1	0.6158	0.8182	1.7e-13	11	700	0.70
2	0.3827	0.5031	0	22	1237	0.54
3	0.2257	0.2750	0	22	1242	0.54
4	0.1231	0.1449	0	18	1015	0.47
5	0.0607	0.0756	1.2e-14	25	1426	0.70
6	0.0264	0.0393	1.3e-14	30	1748	0.91
7	0.0098	0.0204	4.0e-15	28	1619	0.87
8	0.0029	0.0106	6.3e-14	29	1694	0.95
9	0.0006	0.0055	8.0e-15	45	2688	1.60

We note that for $k < n - 1$ the solution found by MWD is locally not unique. It contains zero components b_i of the right hand side b (for which the associated eigenvalues $\lambda_i \in [1, \kappa]$ do not influence the value $f(x^k)/\gamma$) or it contains multiple eigenvalues λ_i (for which the components of the right hand side b can be arbitrarily manipulated as long as their Euclidean norm remains the same; e.g. when $\lambda_1 = \lambda_2$ and $b_1 = 3, b_2 = 4$ then $b_1 := 5, b_2 := 0$ yield the same value $f(x^k)/\gamma$). Hence, the maximizers are degenerate, a situation for which standard nonlinear optimization approaches may have slow local convergence. In this respect the high accuracy of the solutions generated by MWD and the moderate number of iterations are unexpected.

In the next table, k is fixed to $k = 20$, κ is set to $\kappa = 100$, and, as above, the times and accuracies are compared, but now for different values of n.

n	red_0	red_{max}	$dist$	# it.	# f-eval.	time
25	0.0002	0.0361	6.0e-15	68	8205	7.34
50	0.0095	0.0361	1.4e-14	79	17486	14.48
100	0.0235	0.0361	3.6e-12	194	8.2213	67.92
200	0.0250	0.0361	4.5e-12	222	182701	150.23

Observe that the row $n = 200$ refers to 397 variables that are to be adjusted. For this problem size, MWD uses a rather long computation time and the result is somewhat less accurate.

While the example in this sub-section is rather special in exploiting several properties of the cg-method in order to come up with the bound red_{max}, it is conceivable that for some other class of problems, the performance of some other simple algorithms depending on certain parameters can be tested and optimized via MWD as well.

5 Application, Inclination Sensor

The MWD algorithm outlined above has been applied in the manufacturing of industrial inclination sensors. This application shall be detailed next.

5.1 Principle of Function of an Inclination Sensor

The core of the newly developed inclination sensor for industrial applications with high accuracy requirements is a MEMS (microelectromechanical system) that measures the acceleration of a check mass in all three spatial directions.

The electronics of the sensor converts the gravitational movement of the mass into electric signals which, in a first approximation, depend linearly on the acceleration. The integrated firmware computes the inclination (with respect to the gravitational field) of the sensor based on the three values of acceleration. The metal casing protects the MEMS and the electronics of environmental influences, it is used for the attachment of the sensor to the object to be measured, and it also embodies the coordinate system of the sensor to which the computed output angles refer to.

5.2 Calibration Parameters

5.2.1 Signal conversion

In the specific example presented next, a calibration of the sensor is necessary since the variance of the performance of the electronic components is too large to guarantee a precise conversion of the acceleration to a signal. In order to compensate for this effect – as the most simple nonlinear model – a quadratic approach was chosen,

$$U_i = k_{0,i} + k_{1,i} a_i + k_{2,i} a_i^2 \tag{17}$$

where U_i is the value measured along the i-th measurement axis, a_i is the true value of the acceleration along the i-th measurement axis, and $k_{j,i}$ are calibration parameters that describe the behavior of an individual sensor for $0 \leq j \leq 2$ and $1 \leq i \leq 3$. Thus, there are a total on nine quantities $k_{i,j}$ to be determined (cf. function f^{cal} in (1)).

5.2.2 Orientation of the coordinate systems

In addition, the tolerance chain when mounting the MEMS into the casing must be considered. The soldering of the MEMS to the board, screwing the board into the casing and shape deviations of the casing lead to a non tolerably large variance of the MEMS axes' orientation towards the inclination sensor's coordinate system. Accordingly, the orientation of the axes about all three spatial directions must be determined for each inclination sensor during the calibration process. This leads to three further calibration parameters.

5.2.3 Calibration procedure

As detailed above, during the calibrating procedure true measured variables are being adjusted and then it is observed which measured values appear. Based on this information the functional relation between the measured value and the measured variable is determined so that later with the "real" measurement the associated measured variable can be computed for each measured value.

In the specific example, a calibrating station with 2 pivoted supports - one mounted on the other - is used where the axes are adjusted in a way that they are exactly perpendicular, both, to each other as well as to the direction of gravity. Each axis is equipped with an angle measuring system, the precision of which is better by more than one order of magnitude than the desired accuracy of the inclination sensor to be calibrated; these angles can thus be considered as independent freely adjustable actuating variables $\vec{S}$ as described in 1.1.3. Since turning the first axis of the calibration station automatically changes the

orientation of the second axis, the angles of the calibration station's axes (control variables) represent Euler angles according to the "x-y-z"-convention([16]). On the other hand, the values of the sensor signals correspond to the direction cosines of the sensor coordinate axes relative to a fix world coordinate system aligned to gravity direction. Therefore, a conversion of the expression of the sensor's orientation from Euler angles to direction cosines (or vice versa) is necessary. This conversion corresponds to the function g in (2). Thus, in this application both f^{cal} and g are nonlinear functions and also the overall mathematical model $\tilde{f}^{cal}$ (cf. (3)) is nonlinear so that a direct solution via a linear system is not possible.

5.2.4 Determination of the calibration parameters

The calibration parameters can now be taken from the result of an optimization algorithm comparing the measured variables and the transformed measured values. More precisely, the residuals of (3) are to be minimized – an optimization problem that would pose a challenge to most developers. Here, the MWD approach [21] was used allowing a great reduction in modeling and implementation efforts. The developer only had to set up the function $f(X, k_i)$ along with meaningful initial values and bounds for the calibration parameters. As output, the optimal set of calibration parameters minimizing the residual error and best describing the performance of the sensor was returned very quickly.

5.2.5 Flexible improvement of the mathematical model

It turned out, however, that even the optimal calibrating values did not render a satisfactory result. While the accuracy of the sensor aimed for was to be better than $\pm 0.5°$ the remaining measurement deviations still were about $\pm 2.5°$ (see Fig. 5.1 'without compensation"). Since evidently these were systematic deviations, additional disturbance variables had to be present that had not yet been taken into account into the mathematical model. Subsequently, therefore, additional thinkable disturbance variables were included in the mathematical model, followed by the solution via MWD and then, based on the size of the resulting residuals, it was decided whether the additional disturbance variable allowed a sufficiently good description of the observed phenomenon.

This approach highlights another practical advantage of the MWD algorithm. In a first step it allowed a quick formulation and solution of an initial calibration model. The solution of the calibration problem via MWD then revealed that the current model was insufficient. In a second step, various extended models were set up and solved via MWD. Here, it helped that rather than having to set up several different optimization problems it was sufficient to adapt the functional relation between measured variable and measured values as well as to enlarge the set of calibration parameters. Thus it was possible in a fairly short time to identify the cause for the measurement deviations: While it was assumed initially that the 3 measuring axes of the MEMS are aligned quite well orthogonally to each other it was surprising to find out that the modeling via a distorted coordinate system significantly reduced the remaining residual error. Thus, by the definition of an additional two calibration parameters, the accuracy of the sensor could be improved significantly (see Fig. 5.1 "with compensation"). Moreover, as a result of the above described findings, the angular

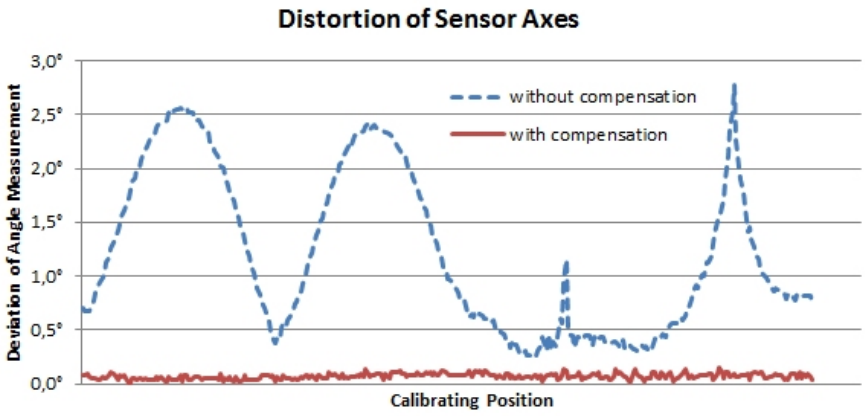

Fig. 5.1: Distortion of the Sensor Axes.

tolerances of the MEMS were further investigated and it turned out that the production process defining the sensor axe's orientation is not sufficiently under control so that deviations of several degree may well occur. Thus, the analysis of the mathematical model unveiled flaws in the purchased parts and its flexible adaption made a simple and fast suppression of the flaw's effects possible - all thanks to an easy-to-use tool for direct minimization such as MWD.

5.3 Transfer to industrial practice

After showing that inclination sensors with satisfactory performance can be produced by the above procedure, the method was included in industrial practice. Each of the produced sensors undergoes a calibration procedure where a set of about 100 measured data (including the associated actuating variables) is recorded. Using the MWD algorithm the calibration parameters for each sensor are being computed and stored in its electronics. Currently a production capacity of 10000 pieces per year is planned.

Thus, due to the contribution of the mathematical optimization, it was possible to develop a highly accurate measuring device at a fairly low cost. An accuracy comparison of this sensor with competing products demonstrates the success of this approach. For the test, all sensors were mounted on an inclined plane with known inclination and were then turned around the normal to this plane. From the measurement result in each position, the Euler angles according to the "x-y-z-convention" [16] are calculated. While the angle ϕ of the first rotation around the z-axis (which - by definition of the sensor-coordinate system - is parallel to the gravitation vector) can not be measured by an inclination sensor, the second Euler angle θ describes the rotation around the x-axis and the third Euler angle ψ the rotation around the new - now tilted - z-axis. Thus θ depicts the angle of the plane's inclination, whereas ψ expresses the sensor's orientation towards the direction of the inclination. Independent from the turning position ψ, the second Euler angle θ computed from the acceleration values of the three axes is expected to always be constant corresponding to the inclination angle of the plane. A variance of angle θ can thus be interpreted as a

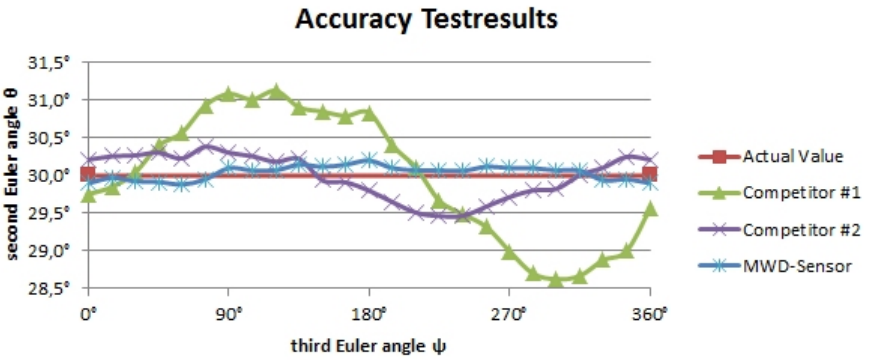

Fig. 5.2: Accuracy of the Test Results.

measuring uncertainty (cf. Fig. 5.2). Since only a small number of sensors were compared, this evaluation does not bear any statistic significance. Nevertheless, the general qualification of the method becomes apparent.

5.4 Summary and perspective

Based on a practical example it was shown how to optimize the measurement accuracy of an inclination sensor with the aid of mathematical compensation. By applying the MWD algorithm the optimization problem was solved automatically so that the developer could concentrate on determining the relationship between measured variables and measured values. Only by the extension of the model to compensate for the unexpectedly large influence of a disturbance variable that was not taken into consideration at first, it was possible to successfully complete the project. Without a simple and reliable direct optimization tool, such as MWD in this example, it would not have been possible to obtain the desired result in such a short time.

In Section 1.1.3 it was explained that by the definition of additional parameters p_i also the inaccuracies of the calibration setting can be modeled mathematically. These parameters can also be determined by the solution of an optimization problem analogously to the determination of the calibration parameters k_j. In the example described here, this possibility has not yet been used since there was no prior experience about the implementability of this approach. After the advantages have now been demonstrated rather clearly, the potential of this approach shall be explored further. It might be possible, for example, to reduce the cost of further calibrating stations by relinquishing tight geometric tolerances of the adjustment of the axes. Instead, a deviation from the orthogonality of the axes could be compensated for mathematically by two additional parameters.

Acknowledgment

The authors would like to thank Andrew Conn, Roland Freund, and Arnold Neumaier for helpful criticism and to an unknown referee for the comments that helped to improve this paper.

References

[1] E. J. Anderson and M. C. Ferris: A direct search algorithm for optimization of expensive functions by surrogates, SIAM J. Optim. 11, 837–857 (2001).

[2] C. Audet: A Survey on Direct Search Methods for Blackbox Optimization and Their Applications, P.M. Pardalos, T.M. Rassias eds, Mathematics Without Boundaries, Springer, 31–56 (2014).

[3] D. Bingham: Virtual Library of Simulation Experiments: Test Functions and Datasets, Optimization Test Problems. http://www.sfu.ca/~ssurjano/optimization.html (2005).

[4] BIPB - JCGM 200:2012, International vocabulary of metrology - Basic and general concepts and associated terms (VIM), 28; Third Edition, (2012).

[5] A. J. Booker, J. E. Dennis, Jr., P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset: A rigorous framework for optimization of expensive functions by surrogates. Structural Optim. 17, 1–13 (1999).

[6] C. A. Botsaris and D. H. Jacobson: A Newton-type curvilinear search method for optimization, Journal of Mathematical Analysis and Applications **54** (1), 217–229 (1976).

[7] A. R. Conn, K. Scheinberg, and L. N. Vicente: Introduction to Derivative-Free Optimization, MPS-SIAM Series on Optimization, SIAM, Philadelphia (2009).

[8] A. R. Conn, K. Scheinberg, and L. N. Vicente: Global convergence of general derivative-free trust-region algorithms to first- and second-order critical points, SIAM J. Optim., 20, 387-415 (2009).

[9] A. R. Conn, K. Scheinberg, and Ph. L. Toint: On the convergence of derivative-free methods for unconstrained optimization, in Approximation Theory and Optimization, Tributes to M. J. D. Powell, M. D. Buhmann and A. Iserles, eds., Cambridge University Press, Cambridge, 83-108 (1997).

[10] A. R. Conn, K. Scheinberg, and Ph. L. Toint, Recent progress in unconstrained nonlinear optimization without derivatives, Math. Program., 79, 397-414 (1997).

[11] T. Csendes: Nonlinear parameter estimation by global optimization - efficiency and reliability. Acta Cybernetica 8, 361–370 (1988).

[12] T. Csendes, L. Pal, J. Oscar, H. Sendin, and J.R. Banga: The GLOBAL Optimization Method Revisited, Report, Institute of Informatics, University of Szeged, Hungary (2008).

[13] A. L. Custodio and L. N. Vicente: Using sampling and simplex derivatives in pattern search methods, SIAM Journal on Optimization, 18: 537-555 (2007).

[14] A. L. Custodio, H. Rocha, and L. N. Vicente: Incorporating minimum Frobenius norm models in direct search, Computational Optimization and Applications, 46: 265–278 (2010).

[15] J. E. Dennis, Jr., N. Echebest, M. T. Guardarucci, J. M. Martinez, H. D. Scolnik, and C. Vacchino: A Curvilinear Search Using Tridiagonal Secant Updates for Unconstrained Optimization, SIAM J. on Opt., **1** (3), 333–357 (1991).

[16] H. Goldstein, C. Poole, and J. Safko: Classical mechanics, 150 –154; Third Edition, Addison-Wesley (2002).

[17] G.H. Golub and C.F. Van Loan: Matrix Computations, second edition, The Johns Hopkins University Press, Baltimore - London, (1993).

[18] N. Hansen: The CMA Evolution Strategy: A Comparing Review. In J.A. Lozano, P. Larraga, I. Inza and E. Bengoetxea (eds.). Towards a new evolutionary computation. Advances in estimation of distribution algorithms. pp. 75–102, Springer (2006).

[19] N. Hansen, A.S.P. Niederberger, L. Guzzella and P. Koumoutsakos: A Method for Handling Uncertainty in Evolutionary Optimization with an Application to Feedback Control of Combustion. IEEE Transactions on Evolutionary Computation, 13(1), 180–197 (2009).

[20] W. Huyer and A. Neumaier: Snobfit - Stable Noisy Optimization by Branch and Fit, ACM Trans. Math. Software 35, Article 9 (2008).

[21] F. Jarre: MWD, smooth Minimization Without using Derivatives, a Matlab collection. http://www.opt.uni-duesseldorf.de/en/forschung-fs.html (2015).

[22] F. Jarre and J. Stoer: Optimierung, Springer Verlag Berlin, Heidelberg, New York (2004).

[23] J. Kiefer: Sequential Minimax Search for a Maximum Proceedings of the American Mathematical Society Vol. 4, No. 3, 502–506 (1953).

[24] R.M. Lewis, V. Torczon, M.W. Trosset: Direct search methods: then and now, Journal of Computational and Applied Mathematics, Volume 124, Issues 1-2, 191–207 (2000).

[25] R.C. Li: On Meinardus' examples for the conjugate gradient method, Mathematics of Computation, Vol. 77, Nr. 261, 335–352 (2008).

[26] C. Elster and A. Neumaier: A grid algorithm for bound constrained optimization of noisy functions, IMA J. Numer. Anal. **15**, 585–608 (1995).

[27] M.D.J. Powell: A new algorithm for unconstrained optimization, in Nonlinear Programming, J.B. Rosen, O.L. Mangasarian, and K. Ritter, eds., Academic Press, New York, 31-65 (1970).

[28] M.D.J. Powell: Direct search algorithms for optimization calculations, Acta Numerica **7**, 287–336 (1998).

[29] L.M. Rios and N.V. Sahinidis: Derivative-free optimization: a review of algorithms and comparison of software implementations, J. Glob. Optim. 56: 1247–1293 (2013).

[30] J. Stoer, R. Bulirsch: "Introduction to Numerical Analysis", Third Edition, Texts in Applied Mathematics 12, Springer (2002).

[31] A.I.F. Vaz and L.N.Vicente: A particle swarm pattern search method for bound constrained global optimization, Journal of Global Optimization, 39: 197–219 (2007).