Optim Lett (2016) 10:383–399
DOI 10.1007/s11590-014-0830-y

ORIGINAL PAPER

A sequential quadratic programming algorithm for equality-constrained optimization without derivatives

Anke Tröltzsch

Received: 30 November 2013 / Accepted: 7 November 2014 / Published online: 25 November 2014

Abstract In this paper, we present a new model-based trust-region derivative-free optimization algorithm which can handle nonlinear equality constraints by applying a sequential quadratic programming (SQP) approach. The SQP methodology is one of the best known and most efficient frameworks to solve equality-constrained optimization problems in gradient-based optimization [see e.g. Lalee et al. (SIAM J Optim 8:682–706, 1998), Schittkowski (Optim Lett 5:283–296, 2011), Schittkowski and Yuan (Wiley encyclopedia of operations research and management science, Wiley, New York, 2010)]. Our derivative-free optimization (DFO) algorithm constructs local polynomial interpolation-based models of the objective and constraint functions and computes steps by solving QP sub-problems inside a region using the standard trust-region methodology. As it is crucial for such model-based methods to maintain a good geometry of the set of interpolation points, our algorithm exploits a self-correcting property of the interpolation set geometry. To deal with the trust-region constraint which is intrinsic to the approach of self-correcting geometry, the method of Byrd and Omojokun is applied. Moreover, we will show how the implementation of such a method can be enhanced to outperform well-known DFO packages on smooth equality-constrained optimization problems. Numerical experiments are carried out on a set of test problems from the CUTEst library and on a simulation-based engineering design problem.

Keywords Derivative-free optimization · Nonlinear optimization · Trust region · Equality constraints · SQP · Numerical experiments

A. Tröltzsch (✉)
German Aerospace Center (DLR), Linder Höhe, 51147 Cologne, Germany
e-mail: anke.troeltzsch@dlr.de

1 Introduction

We consider the equality-constrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to} \quad h(x) = 0, \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $h : \mathbb{R}^n \to \mathbb{R}^m$ are nonlinear twice continuously differentiable functions but where the explicit derivatives are for some reason unavailable.

This paper uses the following notation. The Lagrangian function associated with (1) is denoted by

$$L(x, \lambda) = f(x) + \lambda^T h(x), \tag{2}$$

where λ is the vector of Lagrange multipliers. The gradient of the Lagrangian function is denoted by $g(x)$, and $A(x)$ denotes the $n \times m$ matrix of constraint gradients. We will assume that $A(x_k)$ has full column rank for all x_k. The matrix $H(x)$ denotes the Hessian of the Lagrangian function at x or an approximation to it.

At each iteration of an SQP trust-region method, a quadratic programming (QP) sub-problem of the form

$$\min_{s \in \mathbb{R}^n} g_k^T s + \frac{1}{2} s^T H_k s \qquad \text{subject to} \qquad A_k^T s + h_k = 0, \tag{3}$$

is solved inside the trust region

$$\mathscr{B}_2(x_k, \Delta_k) = \{x \in \mathbb{R}^n \mid \|x - x_k\|_2 \le \Delta_k\}. \tag{4}$$

However, as is well known, restricting the size of the step by a trust-region constraint may prevent the solver from satisfying the linearized constraints in (3). Therefore, Byrd [6] and Omojokun [23] suggested to decompose the step s_k and first compute a *restoration step* r_k (also called normal or vertical step) that lies well inside the trust region and that satisfies the linearized constraints as much as possible. The second step t_k is called *tangential* (or horizontal) and is complementary to r_k. This part of the step aims to reduce the quadratic model of the objective function in (3) while maintaining gains in feasibility obtained through the restoration step.

In the given derivative-free context, the trust-region sub-problem (3) can be established by using models of the objective and constraint functions and the model's gradients. These models will be determined by interpolating known function values at a given set $\mathscr{Y}_k$ of *interpolation points*. The set $\mathscr{Y}_k$, known as the *interpolation set*, contains in our case at least $n + 1$ points (to build a fully determined linear model) and at most $\frac{1}{2}(n + 1)(n + 2)$ points (for a fully determined quadratic model).

Trust-region methods have long been considered for solving unconstrained and constrained optimization problems, and we refer the reader to [8] for an extensive coverage of this topic.

In the literature, other applied frameworks to handle general constraints in model-based derivative-free optimization include augmented Lagrangian method [15], sequential penalty method [10,21], filter-SQP method [7], inexact restoration method [5], filter-trust-region method [17] and sequential linear programming method [26].

The paper is organized as follows. We present the basic framework of our SQP trust-region DFO algorithm in Sect. 2. After recalling elements of polynomial interpolation theory, we discuss the main algorithmic details in Sect. 3. In Sect. 4, details on our particular implementation to enhance robustness and performance of our algorithm are described. Section 5 reports numerical results with the new algorithm and compares it to COBYLA [25,26], DFO [9,10] and NOMAD [3,4], three well-known packages developed for the purpose of solving derivative-free constrained optimization problems and for which the software implementations are publicly available. Comparisons are presented on an equality-constrained subset of the CUTEst test problem library [19] and on a simulation-based engineering design application. We finally give some conclusions and perspectives in Sect. 6.

2 The SQP trust-region DFO algorithm

An outline of the algorithm is given in Algorithm 1 below where the acronym ECDFO stands for "Equality-constrained derivative-free optimization". Its details are discussed in the next section.

Algorithm 1 ECDFO

```
Step 0: Initialization. A trust-region radius $\Delta_0$, a vector of initial Lagrange multipliers $\lambda_0$ and an accuracy threshold $\epsilon$ are given. An interpolation set $\mathcal{Y}_0$, which contains $x_0$, is also given. Parameters $\eta, \gamma_1, \gamma_2, \gamma_3, \theta^r, \theta^t$ are defined. Build the initial interpolation models $m_0^L$ and $m_0^h$ of the Lagrangian and constraint functions. Set $k = 0$.

Step 1: Criticality test.
If $\|\nabla m_k^L(x_k, \lambda_k)\|_2 \le \epsilon$ and $\|h(x)\|_2 \le \epsilon$ and the models are sufficiently accurate, return $x_k$.

Step 2: Compute a trial point and evaluate the objective function.
Step 2.1: Compute the restoration step $r_k$ by minimizing the constraint violation in $\theta^r \mathcal{B}_2(x_k, \Delta_k)$.
Step 2.2: Compute the tangential step $t_k$ by minimizing the model of the Lagrangian $m_k^L$ inside the null space of the gradients of the constraints in $\theta^t \mathcal{B}_2(x_k + r_k, \Delta_k)$.
Step 2.3: Compute $x_k^+ = x_k + r_k + t_k$.
Step 2.4: Evaluate $f$ and $h_i$ at $x_k^+$.
Step 2.5: Compute $L(x_k^+, \lambda_k^+)$ after updating the Lagrange multipliers.

Step 3: Compute merit function value.
Update the penalty parameter $\mu_k$ and compute the merit function value at $x_k^+$. Compute the ratio of achieved versus predicted reduction in the merit function.

Step 4: Define the next iterate and update the trust-region radius.
Take a decision on how to possibly incorporate the current trial point $x_k^+$ into the set $\mathcal{Y}_{k+1}$, define $x_{k+1}$ and determine $\Delta_{k+1}$.

Step 5: Update the model.
If $\mathcal{Y}_{k+1} \neq \mathcal{Y}_k$, compute the interpolation models $m_{k+1}^L$ and $m_{k+1}^h$ around $x_{k+1}$ using $\mathcal{Y}_{k+1}$. Increment $k$ by one and go to Step 1.
```

3 Algorithmic details

3.1 Polynomial interpolation

As derivatives are not available in the given problem setting, g_k, H_k and A_k are approximated by building interpolation models of the Lagrangian function and of the constraint functions, respectively. The coefficients of g_k and H_k, here represented by the vector α, are determined for each constraint from the interpolation conditions

$$\left(m^L(y^i) =\right) \sum_{j=1}^{q} \alpha_j \phi_j(y^j) = L(y^i), i = 1, \ldots, p. \tag{5}$$

To build models of all constraint functions $h^l, l = 1, \ldots, m$, the coefficients β^l have to be determined from the conditions

$$\left(m^{h^l}(y^i) =\right) \sum_{j=1}^{q} \beta^l_j \phi_j(y^j) = h(y^i), i = 1, \ldots, p. \tag{6}$$

The points $y^1, \ldots, y^p$ considered in the interpolation conditions (5) and (6) form the interpolation set $\mathcal{Y}_k$ and the set of functions $\phi = \{\phi_1, \ldots, \phi_q\}$ is a polynomial basis which could be for instance the basis of monomials or the bases of Lagrange or Newton fundamental polynomials. The $p \times q$ matrix $M_k(\phi, \mathcal{Y}_k)$ which contains the polynomial bases $\phi_1, \ldots, \phi_q$ evaluated at the points $y_k^1, \ldots, y_k^p$ is called *interpolation matrix*.

Considering models of variable degree, it is natural to think of models evolving from linear to quadratic as minimization progresses. Thus, the number of interpolation conditions p that are imposed on a model $m(x)$ varies in the interval $[n + 1; \frac{1}{2}(n + 1)(n + 2)]$. We decided to consider always the subset of the first $p \leq q$ polynomial functions of ϕ. As is suggested in [13], we will use the notion of *sub-basis* of the basis ϕ.

3.2 Geometry improvement

For the models to be well-defined, the interpolation points must be poised [11,26], meaning that this set of points must be sufficiently well distributed over the full-dimensional variable space to stay safely away from degeneracy. As this is a crucial detail of model-based DFO algorithms, several strategies have been proposed in the past (e.g. [13,27]). The most common way to avoid a degenerate set of interpolation points is to perform at each iteration a geometry improving step. Such a step consists of a test whether the geometry of the interpolation points is good enough for proceeding or it has to be improved. In the latter case, an auxiliary procedure has to be applied to exchange one or more points from the current interpolation set $\mathcal{Y}_k$ which in turn requires additional function evaluations.

As this is in general not desirable, a new algorithm has been recently designed to substantially reduce the need of such geometry improving steps by exploiting a self-correcting property of the interpolation set geometry [29]. It has been shown that this approach guarantees either decrease in the merit function value or improvement in the geometry of the interpolation set. In particular, in unsuccessful iterations, the algorithm always tries first to improve the geometry by replacing an appropriate point from the set and only if such a point can not be found, the trust-region radius is decreased.

3.3 Step computation

To compute a new trial point in iteration k of an SQP trust-region algorithm, the sub-problem

$$\begin{aligned} &\min_{s\in\mathbb{R}^n} g_k^T s + \tfrac{1}{2} s^T H_k s, \\ &\text{subject to } \; A_k^T s + h_k = 0, \\ &\quad\text{and } \; \|s\|_2 \le \Delta_k \end{aligned} \tag{7}$$

has to be solved.

The possible difficulty of solving (7) is that the linearized constraints and the trust-region constraint may have no common feasible point. To circumvent this difficulty, we apply the approach suggested by Byrd [6] and Omojokun [23] which decomposes the computation of the search step s_k into two sub-steps

$$s_k = r_k + t_k, \tag{8}$$

where r_k and t_k are orthogonal. The vector r_k is the restoration step and $t_k = Z_k u_k$ is the tangential step, where Z_k is a basis for the null space of A_k^T. For computing the first component r_k, the following trust-region model sub-problem

$$\begin{aligned} &\min_{r\in\mathbb{R}^n} \|A(x_k)r + h(x_k)\|_2 \\ &\text{subject to } \; \|r\|_2 \le \theta^r \Delta_k, \end{aligned} \tag{9}$$

where $\theta^r \in (0, 1)$ is a constant, has to be solved. In (9), the constraint violation is minimized inside the trust region, i.e., the step r_k satisfies the linearized constraints in (7) as much as possible. The algorithm is designed so that the full step s_k does not need to move any closer to the feasible space than r_k does.

The second component of s_k is obtained by solving for u_k the following sub-problem

$$\begin{aligned} &\min_{u\in\mathbb{R}^{n-m}} (g_k + H_k r_k)^T Z_k u + \tfrac{1}{2} u^T Z_k^T H_k Z_k u \\ &\text{subject to } \; \|Z_k u\|_2 \le \theta^t \Delta_k, \end{aligned} \tag{10}$$

where $\theta^t \in (0, 1)$ is a constant. This step component aims for optimality without corrupting the gains which where made in the restoration step. In Fig. 1, we can see an illustration of the decomposed step computation, where in the depicted problem, it is possible for the restoration step to reach a feasible point and the tangential step

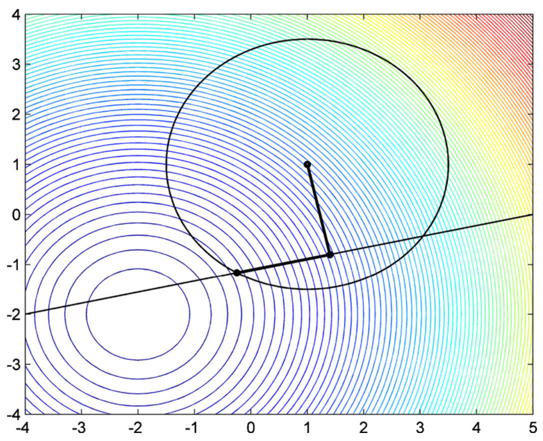


Fig. 1 Solving the trust-region sub-problem in two steps

does optimize the quadratic problem (10) by moving along the linearized constraints until the trust-region boundary is hit.

3.4 Merit function and penalty parameter

After having computed the new trial point $x_k^+ = x_k + s_k$ in Step 2 of Algorithm 1, one has to check in Step 3 whether the re-composed step s_k makes sufficient progress toward the solution of problem (1). To do so, a merit function has to be invoked. Byrd and Omojokun suggest to use the merit function

$$\psi(x,\mu) = f(x) + \mu\|h(x)\|_2, \tag{11}$$

where μ is a non-negative parameter, usually called the penalty parameter. This parameter has to be updated at each iteration of the algorithm (see e.g. [8] for more details).

The trial step s_k is accepted if sufficient decrease in the merit function is produced. To measure this decrease, the actual reduction $ared$ in the function (11) and the predicted reduction $pred$ in the model merit function

$$\bar{\psi}_k(s,\mu) = s^T g_k + \frac{1}{2} s^T H_k s + \mu\|A_k^T s + h_k\|_2 \tag{12}$$

is computed. If the actual reduction $ared$ is sufficiently positive in the sense that $ared \geq \eta pred$, where $\eta \in (0,1)$ is a constant, iteration k is called successful. Otherwise, the iteration is called unsuccessful and s_k is rejected.

3.5 The updating procedure

In Step 4 of the algorithm, it has to be decided whether and how to incorporate the point x_k^+ into the interpolation set $\mathscr{Y}_{k+1}$. As already mentioned, this step is important in a model-based derivative-free optimization algorithm as the quality of the interpolation model, and therefore the quality of the next step computation, is directly dependent on the well-poisedness of the interpolation set. Now we want to briefly summarize the procedure.

As long as the quadratic interpolation models m_k^L and m_k^h are not fully determined by the number of points in $\mathcal{Y}_k$ (i.e. if $p < \frac{1}{2}(n+1)(n+2)$), the new trial point is added to the interpolation set when possible, i.e. when appending the trial point does not deteriorate the poisedness of $\mathcal{Y}_{k+1}$ too much. If the iteration was successful, the iterate $x_{k+1} = x_k^+$ and the trust-region radius $\Delta_{k+1} = \min(\max(\gamma_3 \|s_k\|_2; \Delta_k); \Delta_{max})$ are updated, where $\gamma_3 \geq 1$. Otherwise, $x_{k+1} = x_k$ and $\Delta_{k+1} = \Delta_k$ is set.

When the quadratic models are fully determined by the available interpolation points, the new point is tried to be incorporated by replacing another point in the interpolation set. If the iteration was successful, one point y_k^r is replaced by x_k^+ such as to improve the geometry of the interpolation set as much as possible, e.g. such as to find

$$y_k^r = \arg \max_{y_k^j \in \mathcal{Y}_k} |\ell_{k,j}(x_k^+)|, \tag{13}$$

where $\ell_{k,j}(x_k^+)$ is the Lagrange polynomial associated with the replaced point and evaluated at the new point.

In unsuccessful iterations, the trial point is also attempted to be included in the interpolation set to improve its geometry. Here is where the approach of the self-correcting geometry comes into play and a particular sequence has to be followed. First, a far point from $\mathcal{Y}_k \setminus \{x_k\}$ is attempted to be replaced by the new point x_k^+, i.e. a point which lies outside of the current trust region and for which

$$y_k^r = \arg \max_{y_k^j \in \mathcal{Y}_k \setminus \{x_k\}} \|y_k^j - x_k^+\|_2^2. \tag{14}$$

In this way, the locality of the interpolation set is improved. But, if all points lie already inside the trust region, a point y_k^r is tried to be replaced as in (13) for which the poisedness of $\mathcal{Y}_k$ can be improved. Then $\mathcal{Y}_{k+1} = \mathcal{Y}_k \setminus \{y_k^r\} \cup \{x_k^+\}$ is updated.

In case the new trial point could finally not be included into the interpolation set under the above conditions, it implies that the interpolation set must be reasonably poised, as otherwise we could have improved it during the above procedure. As a consequence, $x_{k+1} = x_k$, $\mathcal{Y}_{k+1} = \mathcal{Y}_k$ is set and the trust-region radius is reduced such that $\Delta_{k+1} \in [\gamma_1 \Delta_k; \gamma_2 \Delta_k]$, where $0 < \gamma_1 \leq \gamma_2 < 1$. More details on the self-correcting geometry approach can be found in [29].

4 Practical implementation issues

The implementation of the described algorithmic ingredients is a DFO SQP code which applies a self-correcting geometry strategy to maintain a sufficient quality of the interpolation models. This combination of an SQP framework with quadratic interpolation gives rise to a robust code for derivative-free optimization.

In this section, we want to give some details on our particular implementation of the algorithm, especially on how robustness and performance of such an implementation can be enhanced in practice.

4.1 Controlling poisedness

In the theory of the self-correcting geometry approach [29], it is possible to choose how often a geometry improving step is performed to control well-poisedness of the interpolation set. Our experiments have shown that for the sake of saving function evaluations, it is sufficient to perform such a step only when convergence of the algorithm has to be granted at the (prospective) end of an optimization run. But as computations are always limited to finite precision, the condition of the system matrix has to be controlled regularly.

In [12] was shown that a relation between the condition number of $\hat{M} = M(\bar{\phi}, \hat{\mathscr{Y}})$ and the measure of Λ-poisedness can be established when considering the basis of monomials $\bar{\phi}$ and $\hat{\mathscr{Y}}$, a shifted and scaled version of $\mathscr{Y}$. Thus, given a sample set $\mathscr{Y} = \{y^1, y^2, \ldots, y^p\}$, a shift of coordinates is first performed to center the interpolation set $\mathscr{Y}$ at the origin. The region $\mathscr{B}$ is then fixed to be $\mathscr{B}_2(0, \Delta(\mathscr{Y}))$ and the radius $\Delta = \Delta(\mathscr{Y}) = \max_{2 \le i \le p} \|y^i - y^1\|_2$, where y^1 denotes the current best iterate which is usually the center of the interpolation, is used to scale the set. The resulting scaled interpolation set $\hat{\mathscr{Y}}$ is then contained in a ball of radius one centered at the origin. In our implementation, the condition number of $\hat{M}$, being the measure cheaper to compute, is used to control the condition of the system matrix.

As described above, in certain situations, e.g. when adding new points to the interpolation set before quadratic degree is reached and no proper exchange of points is possible, or due to numerical finite precision issues, the poisedness of $\mathscr{Y}_k$ may deteriorate. To be sure to have a reliable algorithm, the condition number of the shifted and scaled system matrix $M(\bar{\phi}, \hat{\mathscr{Y}})$ is computed at each iteration before building the polynomial model. The threshold κ_{illcond} for declaring $M(\bar{\phi}, \hat{\mathscr{Y}})$ as badly conditioned is user-defined in the software. In the case $\text{cond}(\hat{M}) > \kappa_{\text{illcond}}$, the singular value decomposition of this matrix is determined and all singular values smaller than a threshold δ are replaced by this threshold. Then the corresponding system which computes the model is solved.

4.2 Representation of the Lagrange polynomials

As $\mathscr{Y}$ varies, the code computes a QR factorization

$$M(\phi, \mathscr{Y})^T = Q_{\mathscr{Y}} R_{\mathscr{Y}}$$

of the matrix of the system M (or of its shifted version $\hat{M}$ if appropriate), where the basis ϕ is that of the monomials. If the vector ℓ_j contains the coefficients of the Lagrange polynomial $\ell_j(x)$ associated with $\mathscr{Y}$, their definition implies that they have to satisfy $M(\phi, \mathscr{Y})\ell_j = e_j$ and hence may be retrieved from the formula $\ell_j = Q_{\mathscr{Y}} R_{\mathscr{Y}}^{-T} e_j$.

4.3 Trust-region radius update

It is common strategy in many trust-region based derivative-free algorithms, and also in the self-correcting geometry approach in [29], to shrink the trust region radius in unsuccessful iterations only after having reached a very well-poised interpolation set. In this case, one can be sure that a bad geometry of the set of points is not responsible for the failure of the iterate but the trust-region radius is too big.

Inspired by the work of Fasano, Morales and Nocedal [18], we observed that a more standard trust-region management as it is used in gradient-based optimization can be very advantageous. In fact, our experiments have shown that a decrease of the trust-region radius by factor 2 at every unsuccessful iteration does in average noticeably enhance the performance of our implementation of ECDFO.

To not corrupt the derivative-free theory, constant shrinking of the trust-region radius Δ at unsuccessful iterations is only performed until a threshold Δ_{min} is reached. Our experience show that $\Delta_{min} = 10^{-3} * \varepsilon$, where ε is the stopping criterion of the algorithm, could be a reasonable choice.

4.4 Trust-region in the sub-problem

To solve the two sub-problems from (9) and (10), we use standard Matlab® routines. But concerning the size of the trust-region in the decomposed sub-problem, we noticed that the use of $\theta_r = \theta_t = 1$ could be a good choice. This choice is covered by the trust-region theory (see e.g. [8]) and guarantees at every step the possibility of moving towards the feasible region (if infeasible) and towards the optimum (if applicable). Other implementations of the Byrd Omojokun approach suggest to use $\theta_r + \theta_t \leq 1$ what restricts the tangential step s_t to a small fraction of the trust-region radius Δ_k in the infeasible case when the restoration step s_r is taking up a more or less considerable part of the available radius Δ_k.

5 Numerical experiments

In the presented experiments, our Matlab-implementation of ECDFO is compared with the three publicly available software packages Constrained Optimization BY Linear Approximation (COBYLA) [25,26] in the double-precision version, Derivative-Free Optimization (DFO) [9,10] and Nonlinear Optimization by Mesh Adaptive Direct Search v.3.6.2 (NOMAD) [3,4].

COBYLA, developed by M.J.D. Powell, is a very well-known FORTRAN package which is part of many publicly available optimization libraries as for instance DAKOTA [1], pyOpt [24] and Scipy and has been ported to other programming languages like Java and C#. COBYLA is applying a trust-region algorithm and building linear approximations of objective and constraint functions to solve nonlinear optimization problems.

The DFO package is a FORTRAN implementation, by Scheinberg, of the trust-region-based derivative-free optimization algorithm. Quadratic interpolation is used to model the objective function and constraints. The trust-region sub-problem is solved by the IPOPT package [33], a nonlinear optimization package that uses an interior point approach.

NOMAD is a C++ implementation of the Mesh Adaptive Direct Search (MADS) algorithm developed by M. Abramson, C. Audet, G. Couture, J. Dennis, S. Le Digabel and C. Tribes and designed for global optimization of blackbox functions subject to nonlinear constraints. NOMAD is amongst others part of the OPTI Toolbox [14] which provides Matlab-interfaces to a selection of open source optimization packages. We use NOMAD with the default option of applying quadratic models to aid finding appropriate search directions. The only non-default option is to lower the required accuracy on the feasibility of the constraints to be $h_{\min} = 10^{-4}$ instead of being zero.

As COBYLA and NOMAD can only handle inequality constraints, the interface to the respective code converts each equality constraint to a pair of inequality constraints.

5.1 Experiments on analytical test problems

The CUTEst testing environment [19] is used in our experiments. We take all equality-constrained test problems provided by the CUTEst collection with a size of at most 30 variables and involving at most 30 equality constraints. Problems where one of the considered codes converges to a different minimum are not considered in the test set as this circumstance makes a direct comparison meaningless. Furthermore, we do not consider problems containing fixed variables because COBYLA and DFO do not provide the required prerequisites in their implementations. This yields a set of 76 test problems which are listed, together with the results of the testing, in Appendix A.

As the considered software packages use all different stopping criteria, an independent criterion needs to be applied for the comparison. For this reason, we use the optimal objective function value computed by the IPOPT package [33] (using analytic first and second derivatives) as a reference for our experiments. We take the number of function evaluations needed until a prescribed number of correct significant figures in the objective function value and the constraint violation was attained. In addition, a run is considered as a fail if more than 15,000 function evaluations are needed to solve the respective test problem.

We report our results using performance profiles [16]. Such profiles compare the number of function evaluations needed by each solver to achieve the desired accuracy in the objective function value and in the constraint values. We use 4 significant figures in $f(x^*)$ and $h(x^*)$ as the level of accuracy. In the following figures, P designates the percentage of problems which are solved within a factor τ of the best solver. Please note that τ has a logarithmic scale in the presented figure.

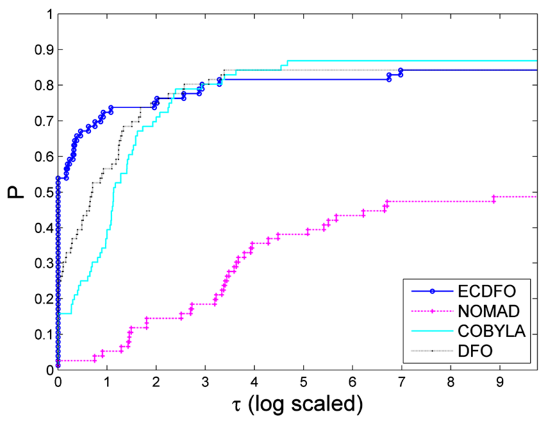


Fig. 2 Comparison of ECDFO, COBYLA, DFO and NOMAD in terms of nbr of function evaluations on 76 equality-constrained test problems of the CUTEst library

The performance profile reported in Fig. 2 shows that for the required accuracy, ECDFO solves 55 % of the test problems faster than the other solvers, whereas DFO solves 25 % of the problems faster than the other, COBYLA solves 16 % of the problems fastest and NOMAD solves only 3 % of the 76 test problems faster than the other three. In terms of robustness, the model based methods outperform the direct-search based approach mostly due to a faster convergence speed. In fact, ECDFO and DFO are finally able to solve 64 and COBYLA solves even 66 of the 76 test problems, NOMAD solves only 37 of the test problems in the given budget of 15,000 function evaluations. We think that direct search methods may have a disadvantage when solving our set of test problems as has been shown in the extensive comparison in [28] for unconstrained derivative-free optimization. Whereas COBYLA may suffer from the use of linear approximations in contrary to DFO and ECDFO applying quadratic approximations which seem to provide faster descent. But at the end, COBYLA solves the most problems in the given evaluation budget.

In the case of more or less required accuracy in the final objective and constraint function values, the results are similar to the presented case of four required digits.

5.2 Experiments on a simulation-based application

Finally, we would like to know whether our new algorithm ECDFO is also suitable to solve simulation-based engineering design problems. Our test application consists of the minimization of the wave drag coefficient of a conical body of revolution. The outer shape of the body is represented by 20 distinct radial points (see Fig. 3). The full body of revolution is formed by rotating the profile around the center-line. The optimization problem involves one equality constraint which is to keep the volume of the initial body constant. This application is (engineering-wise) relatively simple and is often used by engineers to verify aerodynamic simulation tools [22] because optimal body shapes of revolution have been also derived analytically in the past. For the case where the body has a pointed nose and ends in a cylindrical portion

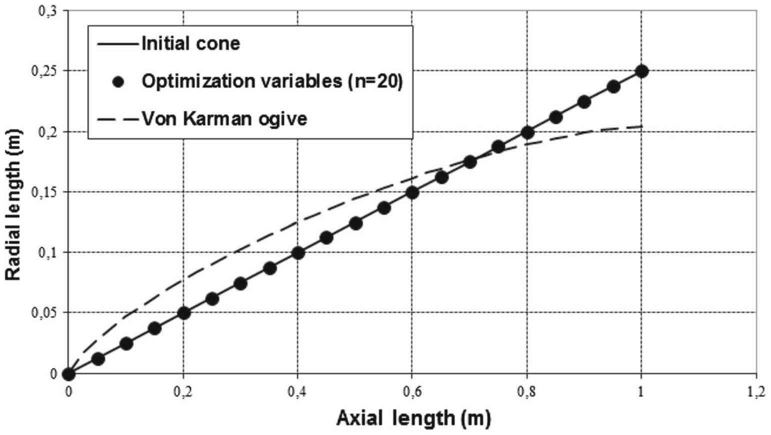
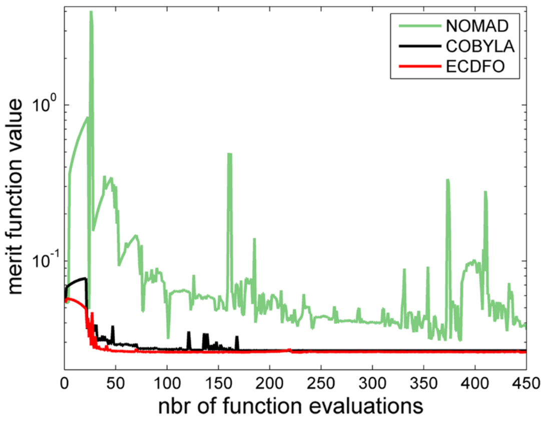


Fig. 3 Aerodynamic shape optimization of a rotational body

Fig. 4 Convergence histories on the "von Karman ogive" test case



(as in our case), the solution to this problem represents the so-called "von Karman ogive" [2]. This circumstance provides the opportunity for engineers (and also for us) to compare simulated to analytic results. In our experiments, the DLR simulation tool HOT Second Order Shock Expansion method (HOTSOSE) [32] is used to evaluate the objective and constraint function values of this aerodynamic design problem.

The numerical results on shaping the "von Karman ogive" by minimizing the wave drag coefficient can be seen in Fig. 4. Each of the number of function evaluations, depicted on the lower axis, include one evaluation of the objective function and one evaluation of the constraint function since the two values are computed in the same run of the utilized simulation code.

In this experiment, we considered the solvers ECDFO, COBYLA and NOMAD. Unfortunately, the software DFO could not be included due to system incompatibilities. All three tested solvers converge eventually to the same minimum. After the given budget of 450 evaluations, the acquired accuracy of the computed solution of ECDFO is

$$\|x^*_{\mathrm{ECDFO}} - x^*_{\mathrm{analytic}}\|_\infty = 4.8 \cdot 10^{-4},$$

COBYLA attained an accuracy of 1.5×10^{-3} and NOMAD of 1.2×10^{-2}, all three compared to the analytic solution of the "von Karman ogive" shape. The solution of COBYLA is slightly less accurate, probably due to the constraint handling by means of two inequalities instead of one equality. In terms of performance, ECDFO shows a slight advantage in the presented test case as it provides a faster decrease in the merit function value right from the beginning. NOMAD shows the typical behaviour of a global optimization method and may thus be more advantageous in highly multimodal applications.

6 Concluding remarks

We presented an SQP trust-region algorithm for equality-constrained optimization without derivatives. As common in model-based DFO, special care must be taken to maintain a valid geometry of the set of interpolation points which is carried out in our algorithm by applying a self-correcting strategy. In the implementation, we added a safeguard strategy as a complement to the self-correcting idea to ensure poisedness in finite precision arithmetic. Furthermore, some practical details are given which we believe did enhance the performance of our algorithm. The numerical experiments have shown that such a code may be efficient and robust for solving analytic problems as well as simulation-based engineering design applications.

Our next step will be to incorporate inequality constraints in the problem formulation which includes the handling of simple bounds on the variables. For the respective implementation, an active-set approach or an interior point method could be considered.

Appendix A: Test problem statistics and detailed results

Table 1 depicts the equality-constrained test problems taken from the CUTEst testing environment. It shows the name of the problem and gives specific details on the number of variables, the number of constraints and the type of the objective and constraint functions. As used in the CUTEst library, 'N' stands for 'no objective function' where a feasibility problem has to be solved, 'L' stands for 'linear function', 'Q' means 'quadratic function', 'S' stands for 'sum of squares' and 'O' stand for any 'other' type of function which could for instance involve exponential terms. If problems involve constraints of different type, the most non-linear type is given in the table. Furthermore, the table shows the number of function evaluations needed by each solver to attain four significant figures of the reference objective function value f^* from IPOPT, and to satisfy $\|h\|_\infty \leq 10^{-4}$.

Table 1 Results on equality-constrained CUTEst test problems

Name	n	m	Type (f)	Type (h)	ECDFO	COBYLA	DFO	NOMAD
10FOLDTR	10	10	N	O	1,315	172	434	1,602
ARGTRIG	10	10	N	O	23	48	72	349
AUG2D	24	9	Q	L	59	303	617	Failed
AUG2DC	24	9	Q	L	58	289	581	Failed
BOOTH	2	2	N	L	5	9	4	14
BROWNALE	10	10	N	O	95	45	263	1,006
BT1	2	1	Q	Q	2,247	21	Failed	Failed
BT10	2	2	L	O	7	21	13	521
BT11	5	3	O	O	Failed	194	89	Failed
BT12	5	3	Q	Q	40	210	43	Failed
BT2	3	1	Q	Q	Failed	233	Failed	4 542
BT3	5	3	S	L	16	168	29	Failed
BT4	3	2	Q	Q	22	37	3	36
BT5	3	2	Q	Q	20	43	13	Failed
BT6	5	2	O	O	138	109	Failed	Failed
BT7	5	3	O	Q	63	136	Failed	Failed
BT8	5	2	Q	Q	28	52	34	13,126
BT9	4	2	L	O	27	59	44	74
BYRDSPHR	3	2	L	Q	Failed	51	21	Failed
CHANDHEU	10	10	N	O	21	60	78	Failed
CLUSTER	2	2	N	O	9	17	8	53
COOLHANS	9	9	N	Q	52	Failed	99	Failed
CUBENE	2	2	N	O	8	11	19	102
CYCLIC3	6	6	N	O	Failed	Failed	Failed	Failed
DIXCHLNG	10	5	S	O	504	Failed	Failed	Failed
EIGENA2	6	3	Q	Q	20	53	64	223
EIGENACO	6	3	S	Q	28	55	116	47
EIGENAU	6	6	N	O	8	13	38	111
EIGENB2	6	3	Q	Q	99	72	75	Failed
EIGENBCO	6	3	S	Q	67	102	190	Failed
FLT	2	2	Q	O	Failed	Failed	Failed	10
GENHS28	10	8	Q	L	19	Failed	113	Failed
GOTTFR	2	2	N	Q	10	16	8	52
HATFLDF	3	3	N	O	Failed	37	30	Failed
HATFLDG	25	25	N	O	79	167	663	193
HEART6	6	6	N	O	Failed	Failed	Failed	Failed
HEART8	8	8	N	O	325	8,295	397	Failed
HIMMELBA	2	2	N	L	5	15	7	62
HIMMELBC	2	2	N	Q	8	14	8	15
HIMMELBE	3	3	N	Q	7	19	17	74

Table 1 continued

Name	n	m	Type(f)	Type(h)	ECDFO	COBYLA	DFO	NOMAD
HS100LNP	7	2	O	O	112	148	263	Failed
HS111LNP	10	3	O	O	Failed	665	Failed	Failed
HS26	3	1	O	O	87	46	55	Failed
HS27	3	1	S	Q	315	208	84	78
HS28	3	1	S	L	10	48	14	340
HS39	4	2	L	O	27	59	44	74
HS40	4	3	O	O	21	51	33	Failed
HS42	4	2	S	Q	16	Failed	24	728
HS46	5	2	O	O	71	156	152	Failed
HS47	5	3	O	O	8,571	82	68	Failed
HS48	5	2	S	L	14	59	33	1,406
HS49	5	2	O	L	82	830	108	3,502
HS50	5	3	O	L	78	153	Failed	Failed
HS51	5	3	Q	L	14	43	33	712
HS52	5	3	Q	L	23	92	Failed	Failed
HS56	7	4	O	O	Failed	174	185	Failed
HS6	2	1	Q	Q	59	29	10	Failed
HS61	3	2	Q	Q	Failed	Failed	12	Failed
HS7	2	1	O	O	27	29	16	45
HS77	5	2	O	O	65	87	160	Failed
HS78	5	3	O	O	47	48	38	Failed
HS79	5	3	O	O	38	46	60	Failed
HS8	2	2	N	Q	9	16	8	83
HS9	2	1	O	L	10	16	13	57
HYDCAR6	29	29	N	O	Failed	Failed	644	Failed
HYPCIR	2	2	N	Q	8	11	7	76
MARATOS	2	1	Q	Q	8	17	9	833
MWRIGHT	5	3	O	Q	60	51	57	Failed
ORTHREGB	27	6	Q	Q	869	475	1,523	Failed
POWELLBS	2	2	N	O	624	64	145	Failed
POWELLSQ	2	2	N	O	23	109	37	62
RECIPE	3	3	N	O	14	37	20	49
RSNBRNE	2	2	N	O	39	76	10	155
S316-322	2	1	Q	Q	Failed	Failed	Failed	Failed
SINVALNE	2	2	N	O	43	126	33	302
ZANGWIL3	3	3	N	L	9	210	14	91

References

1. Adams, B.M., Bauman, L.E., Bohnhoff, W.J., Dalbey, K.R., Ebeida, M.S., Eddy, J.P., Eldred, M.S., Hough, P.D., Hu, K.T., Jakeman, J.D., Swiler, L.P., Vigil, D.M.: Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 5.4 user's manual. Sandia Technical Report SAND2010-2183 (2009, updated 2013) (2013)
2. Ashley, H., Landahl, M.: Aerodynamics of Wings and Bodies. Dover Books on Aeronautical Engineering Series. Addison-Wesley Publishing Company, New York (1965)
3. Audet, C., Dennis, J.J.E.: Mesh adaptive direct search algorithms for constrained optimization. SIAM J. Optim. **17**, 188–217 (2006)
4. Audet, C., Dennis, J.J.E.: A progressive barrier for derivative-free nonlinear programming. SIAM J. Optim. **20**, 445–472 (2009)
5. Bueno, L., Friedlander, A., Martinez, J., Sobral, F.: Inexact restoration method for derivative-free optimization with smooth constraints. SIAM J. Optim. **23**(2), 1189–1213 (2013)
6. Byrd, R.H.: Robust trust region methods for constrained optimization. In: Third SIAM Conference on Optimization, Houston, Texas (1987)
7. Colson, B.: Trust-region algorithms for derivative-free optimization and nonlinear bilevel programming. Ph.D Thesis, The University of Namur, Rempart de la Vierge 8, 5000 Namur, Belgium (2003)
8. Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust-Region Methods. MPS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia (2000)
9. Conn, A.R., Scheinberg, K., Toint, P.L.: On the convergence of derivative-free methods for unconstrained optimization. In: Iserles, A., Buhmann, M. (eds.) Approximation Theory and Optimization: Tributes to M. J. D. Powell, pp. 83–108. Cambridge, England (1997)
10. Conn, A.R., Scheinberg, K., Toint, P.L.: A derivative free optimization algorithm in practice. In: Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, September 2–4 (1998)
11. Conn, A.R., Scheinberg, K., Vicente, L.N.: Geometry of interpolation sets in derivative free optimization. Mathe. Program. Ser. A, B **111**, 141–172 (2008)
12. Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-Free Optimization. MPS-SIAM Series on Optimization. SIAM, Philadelphia (2008)
13. Conn, A.R., Scheinberg, K., Vicente, L.N.: Global convergence of general derivative-free trust-region algorithms to first- and second-order critical points. SIAM J. Optim. **20**(1), 387–415 (2009)
14. Currie, J., Wilson, D.: OPTI: Lowering the barrier between open source optimizers and the industrial MATLAB user. In: Sahinidis, N., Pinto, J. (eds.) Foundations of Computer-Aided Process Operations. Savannah (2012)
15. Diniz-Ehrhardt, M.A., Martinez, J.M., Pedroso, L.G.: Derivative-free methods for nonlinear programming with general lower-level constraints. Comput. Appl. Math. **30**, 19–52 (2011)
16. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–203 (2002)
17. Driessen, L.: Simulation-based optimization for product and process design. Ph.D Thesis, Tilburg University, Tilburg, Netherlands (2006)
18. Fasano, G., Morales, J.L., Nocedal, J.: On the geometry phase in model-based algorithms for derivative-free optimization. Optim. Methods Softw. **24**(1), 145–154 (2009)
19. Gould, N.I.M., Orban, D., Toint, P.L.: CUTEst: a constrained and unconstrained testing environment with safe threads. Comput. Optimiz. Appl. (to appear)
20. Lalee, M., Nocedal, J., Plantenga, T.: On the implementation of an algorithm for large-scale equality constrained optimization. SIAM J. Optim. **8**, 682–706 (1998)
21. Liuzzi, G., Lucidi, S., Sciandrone, M.: Sequential penalty derivative-free methods for nonlinear constrained optimization. SIAM J. Optim. **20**(5), 2614–2635 (2010)
22. Neeb, D., Schwanekamp, T., Gülhan, A.: Preliminary aerodynamic shape optimization of the spaceliner by means of engineering methods. In: 17th AIAA International Space Planes and Hypersonic Systems and Technologies Conference, American Institute of Aeronautics and Astronautics (2011)
23. Omojokun, E.O.: Trust region algorithms for optimization with nonlinear equality and inequality constraints. Ph.D. Thesis, University of Colorado, Boulder (1989)
24. Perez, R., Jansen, P., Martins, J.: pyopt: A python-based object-oriented framework for nonlinear constrained optimization. Struct. Multidiscip. Optim. **45**(1), 101–118 (2012)

25. Powell, M.: A direct search optimization method that models the objective and constraint functions by linear interpolation. In: Gomez, S., Hennart, J.P. (eds.) Advances in Optimization and Numerical Analysis, Mathematics and Its Applications, vol. 275, pp. 51–67. Springer, Netherlands (1994)
26. Powell, M.: Direct search algorithms for optimization calculations. Acta Numerica **7**, 287–336 (1998)
27. Powell, M.J.D.: Developments of NEWUOA for minimization without derivatives. IMA J. Numer. Anal. **28**(4), 649–664 (2008)
28. Rios, L.M., Sahinidis, N.V.: Derivative-free optimization: a review of algorithms and comparison of software implementations. J. Glob. Optim. **56**(3), 1247–1293 (2013)
29. Scheinberg, K., Toint, P.L.: Self-correcting geometry in model-based algorithms for derivative-free unconstrained optimization. SIAM J. Optim. **20**(6), 3512–3532 (2010)
30. Schittkowski, K.: A robust implementation of a sequential quadratic programming algorithm with successive error restoration. Optim. Lett. **5**(2), 283–296 (2011). doi:10.1007/s11590-010-0207-9
31. Schittkowski, K., Yuan, Y.X.: Sequential quadratic programming methods. In: Cochran, J.J., Cox, L.A., Keskinocak, P., Kharoufeh, J.P., Smith, J.C. (eds.) Wiley Encyclopedia of Operations Research and Management Science. Wiley, New York (2010)
32. Streit, T., Martin, S., Eggers, T.: Approximate heat transfer methods for hypersonic flow in comparison with results provided by numerical Navier–Stokes solution. DLR-Forschungsbericht, 94–36 (1994)
33. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006)