SQPDFO - a Trust-Region Based Algorithm for Generally-Constrained Derivative-Free Optimization

A. Tröltzsch,[1] C. Ilic,[2] and M. Siggel[3]

[1)] *German Aerospace Center (DLR), Institute of Software Technology, Köln, Germany.*
(Corresponding Author: anke.troeltzsch@dlr.de)
[2)] *German Aerospace Center (DLR), Institute of Aerodynamics and Flow Technology, Braunschweig, Germany.*
(E-Mail: caslav.ilic@dlr.de)
[3)] *German Aerospace Center (DLR), Institute of Software Technology, Köln, Germany.*
(E-Mail: martin.siggel@dlr.de)

Abstract. Derivative-free optimization is a specific branch of mathematical optimization where first and higher order derivatives of the objective function of the optimization problem are not available, too expensive to compute or too inexact to be used. Such problems do arise in many application areas, e.g. in engineering design optimization, wastewater treatment and quantum chemical processes. As only function value information and no derivative information is available, SQPDFO applies different sampling techniques to build local interpolation models of the objective and constraint functions and uses a self-correcting error technique (Scheinberg and Toint [1]) which guarantees the quality of these models and their derivatives during the optimization process. Throughout the optimization process, first and second order derivatives of these models are used. SQPDFO can handle nonlinear and linear equality and inequality constraints and simple bounds on the variables. It is based on the SQP (Sequential Quadratic Programming) method which solves a sequence of optimization subproblems, each of which optimizes a quadratic program of the original optimization problem. Inequality constraints are carefully handled by slack variables which are not included in the local models to not unnecessarily increase the size of the interpolation matrix. We will present numerical results on a large set of academic test problems from the well-known optimization library CUTEst showing the good performance of the implementation of SQPDFO. Furthermore, we extended the code to run in parallel and several function evaluations can be used in each iteration. This was especially useful when applying SQPDFO to a multidisciplinary DLR research shape design problem of an entire airplane. As one complete function evaluation of the top-level optimization problem can take up to 56 hours, a code which needs a minimum number of iterations is crucial. We will show how SQPDFO is able to find good solutions within a very small number of iterations.

1. INTRODUCTION

Derivative-free optimization algorithms are widely used in practice for several reasons: the explicit evaluation of the derivatives may be impossible, very time-consuming or very inexact.

In this paper, we consider the constrained continuous nonlinear optimization problem

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x) \\ \text{s.t. } &ce(x) = 0, \\ \text{s.t. } &ci(x) \leq 0, \\ \text{s.t. } &l \leq x \leq u, \end{aligned} \tag{1}$$

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

The algorithm SQPDFO (Sequential-Quadratic-Programming Derivative-Free Optimization) applies a model-based trust-region SQP algorithm and is a successor of the algorithm ECDFO [2]. ECDFO has shown very competitive on equality-constrained optimization problems (see [2]). In SQPDFO, the algorithm ECDFO has been extended to handle also bound constraints and more general inequality constraints. Bounds are handled by an active set approach (see [3] for the details). Inequality constraints are handled by adding slack variables. In this approach, one slack variable is included for each inequality constraint like the following.

$$\begin{aligned} ce_{mce+i} = ci_i + x_{sl_i} = 0, \quad i = 1,\ldots,mci \\ \text{s.t. } x_{sl_i} \geq 0, \quad i = 1,\ldots,mci \end{aligned}$$

Each inequality constraint ci_i with added slack variable x_{sl_i} is counted to the problem as an additional equality constraint ce_{mce+i}. Each slack variable x_{sl_i} is counted to the problem as an additional bound-constrained variable. The

implementation of the slack variables in the algorithm is such that they are not included in the interpolation models to not increase the size of the interpolation matrix and the computation time of the used linear algebra inside the algorithm.

We use the following notation in this paper. The Lagrangian function associated with (1) is denoted by

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

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.

Trust-region methods have long been considered for solving unconstrained and constrained optimization problems very successfully. The book [4] contains an extensive coverage of this topic.

In the literature, other applied frameworks to handle general equality and inequality constraints in model-based derivative-free optimization include Augmented Lagrangian method [5], sequential penalty method [6, 7], filter-SQP method [8], inexact restoration method [9], filter-trust-region method [10] and sequential linear programming method [11].

This paper is organized as follows. We present the basic framework of the algorithm SQPDFO and the main algorithmic details in Section 2. Section 3 reports numerical results with the new algorithm on a set of academic test problems from the well-known optimization library CUTEst. In section 4, we will present our multidisciplinary DLR research shape design optimization problem of an entire airplane. Section 5 will give some conclusions of our work.

2. THE ALGORITHM

The outline of the algorithm

The algorithm SQPDFO (Sequential-Quadratic-Programming Derivative-Free Optimization) applies a model-based trust-region SQP algorithm. An outline of SQPDFO is given in Algorithm 1. Its details are discussed below.

Algorithm 1: SQPDFO

Step 1: Initialization.

Step 2: Construction of the interpolation models.

Step 3: Criticality test.

Step 4: Computation of a new trial point and evaluation of the objective function and the constraints.

Step 5: Computation of the merit function value.

Step 6: Defining the next iterate. Go to Step 2.

Step 1: Initialization

Before the optimization process can be started, some algorithmic parameters have to be set. An initial trust-region radius Δ_0 and the accuracy thresholds ε_g for the model gradient and ε_c for the constraints, to check convergence, can be given by the user. The default values are $\Delta_0 = 1$ and $\varepsilon_g = \varepsilon_c = 10^{-4}$. A vector of initial Lagrange multipliers λ_0 can be also given by the user. If is is not given, the Lagrange multipliers are initialized by the algorithm. A starting point x_0 and the lower and upper bounds on the variables, if any, have to be given by the user. An initial interpolation set Y_0 is sampled by the algorithm around the given starting point x_0 using the given distance Δ_0. The iteration counter $k = 0$ is set by the algorithm.

Step 2: Construction of the interpolation models

As it is assumed that no first or second order information of the objective and constraint functions is available, this information is approximated by the algorithm. To do so, local interpolation models of the objective and constraint functions are built. Then the model gradient and Hessian information are used to compute the next trial step.

In each iteration, the interpolation models m_f of the objective function, m_{ce_i} and m_{ci_i} of all constraint functions and m_L of the Lagrangian function must be built. In our context, the models will be determined by interpolating known objective and constraint function values at a given set Y of *interpolation points*, meaning that

$$\begin{aligned} m_f(y) &= f(y), \\ m_{ce_i}(y) &= ce_i(y), \forall i = 1, ..., mce, \\ m_{ci_i}(y) &= ci_i(y), \forall i = 1, ..., mci, \\ \text{for all } \ & y \in Y. \end{aligned}$$

All inequality constraints are reformulated to equality constraints by applying slack variables as it is described in Section 1. The model of the Lagrangian function is built as

$$m_L(x, \lambda) = m_f(x) + \lambda^T m_{ce}(x), \tag{3}$$

where λ is the vector of Lagrange multipliers.

For the models to be well-defined, the interpolation points must be *poised* [11, 12], meaning that this set of points must be sufficiently well distributed over the full-dimensional variable space to stay safely away from degeneracy. The quality of *poisedness* of Y can be measured using the following notion [13]. Let a set $B \in \mathbf{R}^n$ be given and let ϕ be a basis in the polynomial space P_n^d. A poised set $Y = \{y^1, y^2, ..., y^p\}$ is said to be Λ-poised in B for some $\Lambda > 0$ if and only if for the basis of Lagrange polynomials associated with Y

$$\Lambda \geq \max_{1 \leq i \leq p} \max_{x \in B} |\ell_i(x)|.$$

This upper bound on the absolute value of the Lagrange polynomials in a region B can be interpreted as a measure of the distance to a nonpoised set or equivalently to a singular system matrix [13] (page 49).

For more information on polynomial interpolation in derivative-free optimization, see e.g. [3, 13].

Step 3: Criticality test

As the optimization problem is internally reduced to a bound-constrained equality-constrained optimization problem, we only have to check the value of the projected gradient and the values of the constraint values all to be small enough.

Hence, if in Step 3 of the algorithm, the projected model gradient and the constraint values are all small enough, meaning that

$$||P_F(x_k - \nabla m_k) - x_k||_\infty \leq \varepsilon_g \tag{4}$$

and

$$||ce(x_k)||_\infty \leq \varepsilon_c, \tag{5}$$

the optimization process is stopped successfully and the respective optimization variables x_k are returned as the solution of the optimization problem to be solved.

It has been shown in [3, 14] that the use of the projected model gradient as a stopping criterion is justified if the quality of the model gradient ∇m_k is sufficiently good. The result of the proof in [3] shows that

$$||P_F(x_k - \nabla f) - x_k||_\infty \leq ||P_F(x_k - \nabla m_k) - x_k||_\infty + ||\nabla f - \nabla m_k||_\infty. \tag{6}$$

From [13] we know that the quality of the model gradient is bounded by

$$||\nabla f - \nabla m_k||_\infty \leq \Delta_Y + \Lambda \tag{7}$$

where Δ_Y is the distance from the current iterate to the furthest point in the actual interpolation set Y_k and Λ is the poisedness, the maximum of all maximized Lagrange polynomials. This measure can be approximated by the condition number of the interpolation matrix in the case where the interpolation set Y_k is shifted to the origin and normalized to be in the range between 0 and 1.

Hence, in the criticality test in Step 3 of Algorithm 1, we test that the projected model gradient is smaller than some given ε_g, that Δ_Y is small and that $\Lambda \leq 1.5$. If the first condition is fulfilled but not the latter two, a new poised interpolation set Y_k^+ in a smaller region, e.g. $\Delta_Y \leq \varepsilon_g$, is computed.

Step 4: Computation of a new trial point

To compute a new trial point inside an SQP trust-region method, a quadratic programming (QP) sub-problem of the form

$$\min_{s\in\mathbb{R}^n} g_k^T s + \frac{1}{2} s^T H_k s \qquad \text{subject to} \quad A_k^T s + ce_k = 0, \tag{8}$$

is solved inside the trust region

$$\mathscr{B}_2(x_k,\Delta_k) = \{x \in \mathbb{R}^n \mid \|x - x_k\|_2 \leq \Delta_k\}. \tag{9}$$

However, as is well known, restricting the size of the step s_k by a trust-region constraint Δ_k may prevent the solver from satisfying the linearized constraints in (8). Therefore, Byrd [15] and Omojokun [16] 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 constraint violation is minimized by solving the following trust-region model sub-problem

$$\begin{aligned} &\min_{r\in\mathbb{R}^n} \|A(x_k)r + ce(x_k)\|_2 \\ &\text{subject to } \|r\|_2 \leq \theta^r \Delta_k, \end{aligned} \tag{10}$$

where $\theta^r \in (0,1)$ is a constant.

The second step t_k is called *tangential* (or horizontal) *step* and is complementary to r_k. The component t_k is obtained by solving for u_k the following sub-problem

$$\begin{aligned} &\min_{u\in\mathbb{R}^{n-m}} (g_k + H_k r_k)^T Z_k u + \tfrac{1}{2} u^T Z_k^T H_k Z_k u \\ &\text{subject to } \|Z_k u\|_2 \leq \theta^t \Delta_k, \end{aligned} \tag{11}$$

where $\theta^t \in (0,1)$ is a constant. This part of the step aims to reduce the quadratic model of the objective function in (8) while maintaining the gains in feasibility obtained through the restoration step by operating in the nullspace of the Jacobian.

Finally, the new trial point $x_k^+ = x_k + s_k$ with $s_k = r_k + t_k$ is computed.

Step 5: Computation of the merit function value

In Step 5, the objective and constraint functions must be evaluated at the new trial point x_k^+. Then, one has to check whether the step s_k makes sufficient progress towards the solution of the original optimization problem in (1). The proper way is to establish a merit function which contains the actual objective and constraint function values. Byrd and Omojokun suggest to use the merit function

$$\psi(x,\mu) = f(x) + \mu \|ce(x)\|_2, \tag{12}$$

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. [4] for more details).

The trial step s_k is accepted if sufficient decrease in the merit function is produced. To measure this decrease, in trust-region methods, the achieved reduction *ared* in the merit function (12) and the predicted reduction *pred* in the

model merit function

$$\bar{\psi}_k(s,\mu) = s^T g_k + \frac{1}{2} s^T H_k s + \mu \|A_k^T s + ce_k\|_2 \tag{13}$$

is computed. If the achieved 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.

Step 6: Definition of the next iterate

To define the next iterate, the self-correcting geometry approach of Scheinberg and Toint [1] is used. In this step, it has to be decided whether and how to incorporate the point x_k^+ into the interpolation set Y_{k+1}. In the default case, optimization starts with $n+1$ iteration points and a linear model is built. As soon as function values are evaluated at new trial points, these points are tried to be incorporated in the interpolation models and underdetermined quadratic interpolation models are built. As long as the quadratic interpolation models m_f, m_{ce} and m_{ci} are not fully determined by the number of points in 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 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

$$y_k^r = \arg\max_{y_k^j \in \mathscr{Y}_k} |\ell_{k,j}(x_k^+)|, \tag{14}$$

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 $Y_k \setminus \{x_k\}$ 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

$$y_k^r = \arg\max_{y_k^j \in \mathscr{Y}_k \setminus \{x_k\}} \|y_k^j - x_k^+\|_2^2. \tag{15}$$

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 (14) for which the poisedness of Y_k can be improved. Then $Y_{k+1} = Y_k \setminus \{y_k^r\} \cup \{x_k^+\}$ 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, $x_{k+1} = x_k$ and $Y_{k+1} = Y_k$ is set and the trust-region radius is reduced such that $\Delta_{k+1} = \gamma_1 \Delta_k$, where $0 < \gamma_1 < 1$. More details on the self-correcting geometry approach can be found in [1].

Finally, $k = k+1$ is set by the algorithm and the next iteration starts with Step 2 again.

3. NUMERICAL EXPERIMENTS ON ANALYTICAL TEST PROBLEMS

In this section, we compare our solver to the well-known solver COBYLA (Constrained Optimization BY Linear Approximation) which was developed by M. J. D. Powell [11, 17]. It implements a sequential trust-region algorithm that tries to maintain a regularly shaped simplex throughout the iterations. The code is known to perform very well and is included in many freely available libraries like SciPy, pyOpt, NLopt, PDFO and others. For our comparison, we used the Python-interface to COBYLA in SciPy.

The software implementation of SQPDFO

SQPDFO is implemented as a Python 3 package and its code is hosted as open source software on GitHub (https://github.com/DLR-SC/sqpdfo). It is licensed using the permissive three-clause BSD license, which allows a free use of the software. SQPDFO is based to a large extend on the BCDFO algorithm [14], which was originally implemented in MATLAB and now has been rewritten in Python and has been re-licenced to a BSD license.

The SQPDFO Python package can be installed similarly to other Python packages by executing the `setup.py` or by using `pip install`.

The optimization algorithm is started by calling `sqpdfo.optimize`. The function parameters contain the objective function, the initial parameters as well as upper and lower bounds for the parameters. Constraints are added via optional arguments by passing the number of equality and inequality constraints as well as the constraint function. More details on how to use the software can be found in the example repository on https://github.com/DLR-SC/sqpdfo/tree/master/examples.

The CUTEst testing library

In our numerical experiments, the CUTEst testing environment [18] is used. This library contains more than 1000 unconstrained, bound-constrained, equality, inequality or generally constrained optimization problems of different size, type and complexity. The problems are written in SIF (Standard Input Format). The library provides interfaces to be used from Fortran, C, Matlab and Python code. In our comparison, we take all inequality- and equality-constrained test problems provided by the CUTEst library with a size of at most 30 variables and involving at most 30 constraints.

After running COBYLA and SQPDFO on these 476 test problems, 61 of them had to be excluded from our comparison because the two solvers failed both on these problems. Furthermore, problems where the two optimizers converge to different minima are not considered in the test set as this circumstance makes a direct comparison meaningless. This yields a set of 373 test problems. From these 373 problems, 241 involve only equality constraints, 106 involve only inequality constraints and 26 problems involve both, equalities and inequalities.

The performance profile

We report our results using performance profiles [19]. 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. As a reference, we use the optimal function values computed by the solver IPOPT [20, 21] (using first and second derivatives). As IPOPT uses derivative information, it is not included in our comparison.

In the presented figures below, 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.

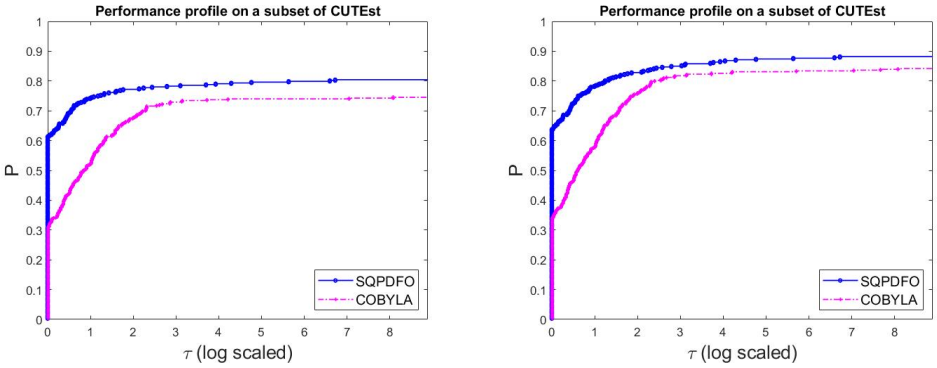
Numerical results

For both solvers, COBYLA and SQPDFO, the stopping criteria of the algorithms were set to 10^{-4} and a function evaluation limit (counting only objective function evaluations) of 15000 was imposed. In our comparison, we use two different levels of accuracy: 2 and 4 digits in $f(x^*)$, $ce(x^*)$ and $ci(x^*)$, compared to the reference values.

The performance profile reported in Figure 1(a) shows that (for the required accuracy of 4 digits) SQPDFO solves 61% of the test problems faster than COBYLA and COBYLA solves 31% of the 373 test problems faster than SQPDFO. The remaining 8% could be solved by the two solvers but not to the required accuracy of 4 digits in the objective and constraint functions. Finally, SQPDFO solves 81% of the problems and COBYLA solves 74% of the test set to the required accuracy of 4 digits.

For low accuracy (in Figure 1(b)), COBYLA manages to solve 84% and SQPDFO solved 88% of the test problems to the required accuracy of 2 digits. Here, SQPDFO solves 64% of the test cases faster than COBYLA and COBYLA solves 34% of the problems fastest. The remaining 2% could not be solved by both solvers to the required accuracy.

The overall conclusion is that SQPDFO is a little more robust and twice as fast as COBYLA and it seems to make a visible difference to use quadratic instead of linear approximations in constrained derivative-free optimization.



(a) 4 digits of accuracy in final function values (b) 2 digits of accuracy in final function values

FIGURE 1. Experiments on a subset of CUTEst

4. THE ENGINEERING DESIGN OPTIMIZATION PROBLEM

The optimization problem

We applied SQPDFO to a high-fidelity, computationally intensive *overall aircraft design* problem. Such a problem is composed of many distinct engineering disciplines, each of which is tasked with a design of an aircraft subsystem. In one possible optimization formulation, an overall aircraft design problem can be decomposed into several disciplinary suboptimizations, controlling disciplinary local variables, connected through an overall system optimization, controlling the global variables. Local variables are understood as those which have a large influence on discipline's suboptimization and a smaller influence on other disciplines' suboptimizations. Global variables are those which have large influence on all disciplines. The reason for making the decomposition is to enable effective treatment of large number (on the order of hundreds to thousands) of variables in conjunction with even larger number of constraints (from tens of thousands to tens of millions).

In this setting, SQPDFO was applied at the level of global variables. For each change in the global variables, all disciplines perform their own suboptimizations and the overall objective function and global constraint values are assembled and provided as result of evaluation to SQPDFO. This was deemed a good fit for the problem, because the separate disciplines, although often internally using gradient-based optimizers on their local variables, were unable to provide derivatives for the global variables. Another important reason is the numerical noise inherent in such a large-scale problem, which is well handled by SQPDFO's explicit treatment of interpolation poisedness.

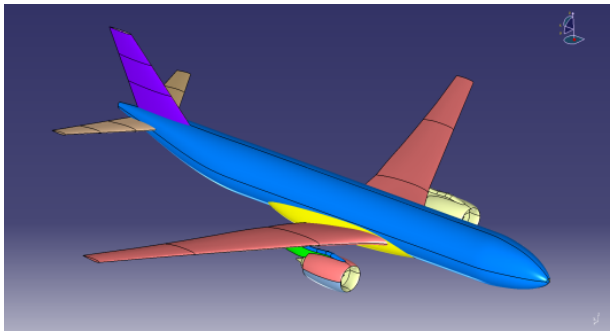


FIGURE 2. Parametric CAD model of a long-range, twin-engine airliner.

The detailed description of the aircraft design problem and engineering discussion of the results can be found in a previous publication by the authors [22]. Here only a short summary of the problem setup and results pertinent to performance of SQPDFO are shown, as follows. A large twin-engine, wide-body, long-range transport aircraft was to be optimized to minimize the mass of fuel expended on a prescribed mission. (For a concrete feeling: a mix of

300 passengers and 10 tons of cargo were to be flown over a 10,500 km distance, which is approximately the flight route between Frankfurt and Singapore.) A geometry model of such aircraft in a CAD system is given by Figure 2. SQPDFO controls two variables, the wing aspect ratio and wing sweep. Several geometry variations are shown on Figure 3. There were 10 global constraints handled by SQPDFO, such as take-off field length, landing approach speed, ground stability, and so on.

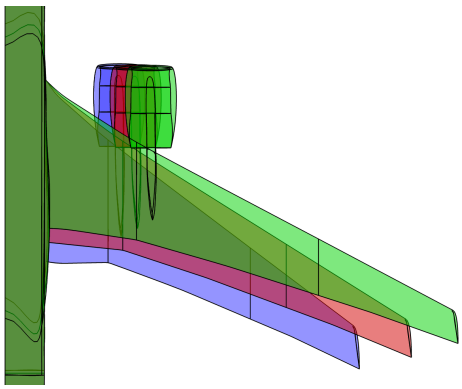


FIGURE 3. Wing shape variations with changes of aspect ratio and wing sweep.

Three aircraft engineering disciplines were included that perform either suboptimization or evaluation:

- *Aircraft synthesis* evaluates the expended fuel mass and accumulates masses needed by other disciplines (such as mass on take-off or in the middle of the flight). It is a quick evaluation method based on explicit formulas and semi-empirical data, no optimization was performed.

- *Aerodynamic airfoil design* adapts the local shape of wing sections to best match the current global shape of the wing, as represented by the local objective of minimal aerodynamic drag. It is an adjoint gradient-based aeroelastic optimization process, using RANS flow and linear elastic structure, implemented within the FlowSimulator framework [23]. In the present problem, 126 variables (airfoil control points) and 3 constraints (steady flight equations) were used in the suboptimization.

- *Design loads and structural sizing* computes the relevant critical forces (loads) to which the aircraft may be exposed in flight and optimizes the internal structure to be able to withstand them, while having the minimal mass. It is an aeroelastic structural design process based on NASTRAN, employing panel aerodynamics and gradient-based mass optimization [24]. There are 392 variables (structural element thicknesses) and over 800,000 constraints (discretized strength and buckling failure criteria, for each critical load case).

The parallel extension of SQPDFO

The computation time of one complete function evaluation from the perspective of SQPDFO, which was one full set of disciplinary suboptimizations from the perspective of disciplines, was 56 hours on a high-performance computing (HPC) cluster. During one evaluation a variable number of computing nodes was used, depending on the precise execution order of disciplines, with up to 4 nodes at peak moments. Each node consisted of two AMD Epyc 7601 32-core processors, resulting in a maximum of 256 cores in use. For reference, the whole cluster had 2,280 nodes with 145,920 cores, theoretical peak performance (Rpeak) of 2568 TFLOPS and maximum measured performance (Rmax) of 1746 TFLOPS.

For this reason, the implementation of SQPDFO was extended to provide several points in the n-dimensional space in parallel, so that they can be evaluated in parallel, when the cluster occupancy allows for it. Obviously, this possibility can be used when building the initial interpolation model. Here, $n+1$ points, including the starting point, have to be evaluated to build the initial model. Thus, the code was modified to provide those points at once in the case where HPC facilities are available. To use the availability of possible parallel function evaluations even more, we have extended the code to provide a user-defined number of points in each iteration (inside the trust region), such that their function values can be evaluated at the same time. After each such iteration, the algorithm decides whether

and how it can incorporate several of the points in the next interpolation set to build a new interpolation model. One can imagine that the possibility of a faster decrease is given when applying such an approach. Above all, convergence theory of the algorithm holds as the trust-region framework allows for such a *double-step* and still guarantees global convergence to a local point (see e.g. [4]).

Optimization results

Due to the long run time of one function evaluation, optimization was performed in two stages, a low-fidelity stage and a high-fidelity stage.

In the first stage, the actual computationally intensive disciplines were replaced by approximate reduced-order models (ROMs). The data for the ROMs was produced by the conceptual aircraft design tool OpenAD [25], and the ROMs themselves were built using the SMARTy toolbox [26]. The evaluation time of the approximate model is on the order of seconds, and it was used to tune the parameters of SQPDFO such as to minimize the number of serial function evaluations and thus the overall run time of the optimization process. In addition, since each set of parallel evaluations counts as a single serial evaluation from the viewpoint of total optimization run time, one decision was to recompute fully the interpolation set after each trust-region iteration, in order to maintain the best possible poisedness. Furthermore, this low-fidelity stage was also used to reduce the set of global constraints to possibly remove those which were far away from being active at the optimum.

The convergence history of the first-stage optimization, with the SQPDFO parameters as finally tuned, is given by Figure 4. Both, the convergence of the objective function (expended fuel mass) and of the feasibility (a norm of constraint satisfaction) are presented. It can be seen that the optimization had already converged after seven iterations. On a closer inspection, we could see that SQPDFO was proceeding further only to more tightly fix the feasibility. Furthermore, the approximate model allowed the design to move much more from the initial point than would be expected (by engineering judgment) in the high-fidelity model. Only 2 out of the 10 constraints became active, and that happened only at the extreme design change, near the approximate model's optimum. Therefore, and to save computing time for the optimization of the high-fidelity model, all global constraints were dropped in the second stage.

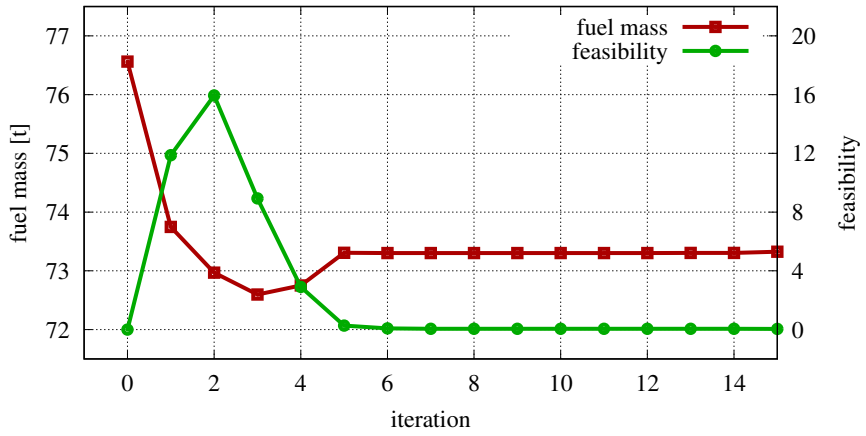


FIGURE 4. Convergence history of the low-fidelity model (ROMs)

The second stage was the optimization of the high-fidelity model, i.e. with the actual disciplines, using the tuned parameters from the first stage and no global constraints. The optimization run was manually stopped after two weeks of runtime (five iterations of the optimizer) since this was the allocated maximum time budget. However, it is expected that any further improvement to the objective would have been within the threshold of numerical noise from the disciplinary suboptimizations which themselves had relaxed convergence criteria what is known to produce a substantial amount of noise in the optimization of the global variables. The convergence history of the second-stage optimization can be seen on Figure 5. The expended fuel mass was reduced through optimization by 10.2%, which is a significant result.

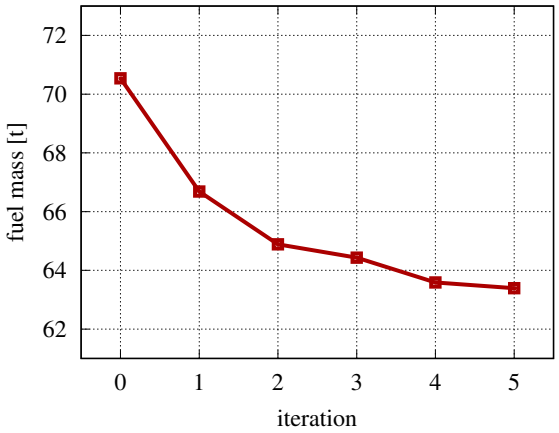


FIGURE 5. Convergence history of the high-fidelity model

As a side note, when comparing the fuel mass values in the optimization with the reduced model and the full model (Figures 4 and 5), it can be seen that the reduced model is just a fairly good representation of the full model. This was due to the reduced model being both, calibrated around the initial point and without any data from the full model. But this was not an issue, since in the context of tuning the parameters of the optimization algorithm, the reduced model had only to mirror the qualitative aspects of the full model.

Eventually, when applied to the full overall aircraft design problem, SQPDFO performed exactly as it was hoped for, converging the design to a practical level within the allotted computational time and resources.

5. CONCLUSIONS

We presented our algorithm SQPDFO, a trust-region algorithm for constrained optimization without derivatives which uses the well-known Sequential Quadratic Programming technique to handle the constraints. As common in model-based derivative-free optimization, 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-correction strategy.

The numerical experiments reported suggest that such an algorithm may be efficient and robust for solving analytic problems. Also our implementation which allows for parallel function evaluations has shown to perform well on simulation-based engineering design applications of different levels of fidelity. Especially the optimization of the high-fidelity model has shown a significant decrease of the aircraft fuel mass of 10.2% within only five iterations of the algorithm.

REFERENCES

1. K. Scheinberg and Ph. L. Toint, "Self-correcting geometry in model-based algorithms for derivative-free unconstrained optimization," SIAM Journal on Optimization **20**, 3512–3532 (2010).
2. A. Tröltzsch, "A sequential quadratic programming algorithm for equality-constrained optimization without derivatives," Optimization Letters **10**, 383–399 (2016).
3. S. Gratton, Ph. L. Toint, and A. Tröltzsch, "An active set trust-region method for derivative-free nonlinear bound-constrained optimization," Optimization Methods and Software **26**, 875–896 (2011).
4. A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *Trust-Region Methods*, MPS-SIAM Series on Optimization (Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000).
5. M. A. Diniz-Ehrhardt, J. M. Martinez, and L. G. Pedroso, "Derivative-free methods for nonlinear programming with general lower-level constraints," Computational and Applied Mathematics **30**, 19 – 52 (2011).
6. A. R. Conn, K. Scheinberg, and Ph. L. Toint, "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).
7. G. Liuzzi, S. Lucidi, and M. Sciandrone, "Sequential penalty derivative-free methods for nonlinear constrained optimization," SIAM Journal on Optimization **20**, 2614–2635 (2010).

8. B. Colson, *Trust-region algorithms for derivative-free optimization and nonlinear bilevel programming*, PhD thesis, The University of Namur, Rempart de la Vierge 8, 5000 Namur, Belgium (2003).
9. L. Bueno, A. Friedlander, J. Martinez, and F. Sobral, "Inexact restoration method for derivative-free optimization with smooth constraints," SIAM Journal on Optimization **23**, 1189–1213 (2013).
10. L. Driessen, *Simulation-based Optimization for Product and Process Design*, PhD thesis, Tilburg University, Tilburg, Netherlands (2006).
11. M. Powell, "Direct search algorithms for optimization calculations," Acta Numerica **7**, 287–336 (1998).
12. A. R. Conn, K. Scheinberg, and L. N. Vicente, "Geometry of interpolation sets in derivative free optimization," Mathematical Programming, Series A and B **111**, 141–172 (2008).
13. A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-Free Optimization*, MPS-SIAM Series on Optimization (SIAM, Philadelphia, PA, USA, 2008).
14. A. Tröltzsch, *An active-set trust-region method for bound-constrained nonlinear optimization without derivatives applied to noisy aerodynamic design problems*, PhD thesis, Institut National Polytechnique de Toulouse (2011).
15. R. H. Byrd, "Robust trust region methods for constrained optimization," in *Third SIAM Conference on Optimization, Houston, Texas* (1987).
16. E. O. Omojokun, *Trust region algorithms for optimization with nonlinear equality and inequality constraints*, Ph.D. thesis, University of Colorado, Boulder, CO, USA (1989).
17. M. J. D. Powell, "A direct search optimization method that models the objective and constraint funcions by linear interpolation," in *Advances in Optimization and Numerical Analysis*, Proceedings of the Sixth Workshop on Optimization and Numerical Analysis, Oaxaca, Mexico, vol. 275, edited by S. Gomez and J. P. Hennart (Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994) pp. 51–67.
18. N. I. M. Gould, D. Orban, and Ph. L. Toint, "CUTEst: a constrained and unconstrained testing environment with safe threads," Tech. Rep. (Rutherford Appleton Laboratory, Scientific Computing Department, Oxfordshire, GB, 2013).
19. E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," Math. Program. **91**, 201–203 (2002).
20. A. Wächter, *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*, PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania (2002).
21. A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," Math. Program. **106**, 25–57 (2006).
22. Časlav Ilić, A. Merle, M. Abu-Zurayk, S. Görtz, M. Leitner, Özge Süelözgen, T. Kier, M. Schulze, T. Klimmek, C. Kaiser, D. Q. Martin, A. Schuster, M. Petsch, D. Kohlgrüber, J. Häßy, R.-G. Becker, S. Gottfried, and A. Tröltzsch, "Cybermatrix protocol: a novel approach to highly collaborative and computationally intensive multidisciplinary aircraft optimization," in *AIAA Aviation 2020* (2020).
23. A. Merle, A. Stueck, and A. Rempke, "An adjoint-based aerodynamic shape optimization strategy for trimmed aircraft with active engines," in *AIAA Aviation 2017* (2017).
24. T. Klimmek, M. Schulze, M. Abu-Zurayk, Časlav Ilić, and A. Merle, "An independent and in high-fidelity based MDO tasks integrated process for the structural and aeroelastic design of aircraft configurations," in *IFASD Conference 2019* (2019).
25. S. Woehler, G. Atanasov, D. Silberhorn, B. Fröhler, and T. Zill, "Preliminary aircraft design within a multidisciplinary and multifidelity design environment," in *Aerospace Europe Conference 2020* (2020).
26. D. Maruyama, S. Görtz, and D. Liu, "Robust and reliability-based design optimization of airfoils considering geometrical uncertainties," in *2nd UMRIDA UQ and RDO Symposium* (2016).