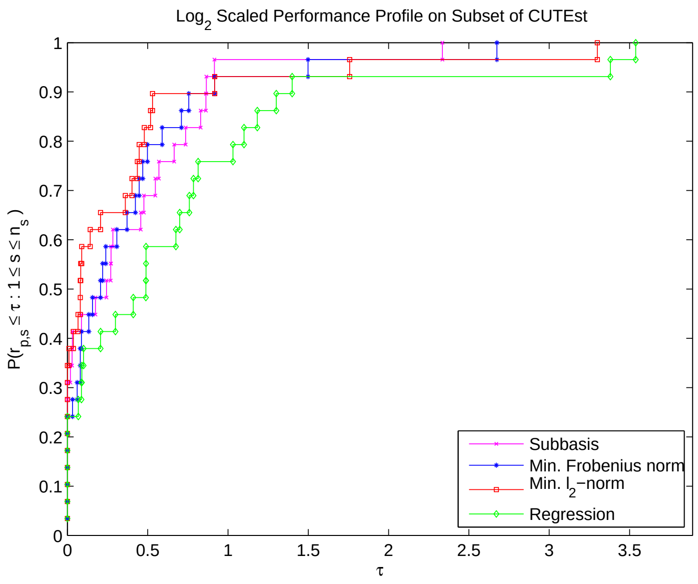
A derivative-free trust-funnel method for equality-constrained nonlinear optimization

by Ph. R. Sampaio and Ph. L. Toint

Report NAXYS-??-2014 8 February 2014



University of Namur, 61, rue de Bruxelles, B5000 Namur (Belgium)

http://www.unamur.be/sciences/naxys

A derivative-free trust-funnel method for equality-constrained nonlinear optimization

Ph. R. Sampaio[*] and Ph. L. Toint[†]

8 February 2014

1 Introduction

Most available algorithms in the domain of derivative-free optimization (DFO) essentially address the unconstrained case (as those proposed by Powell [27, 28, 29], Conn Scheinberg and Toint [4], Scheinberg and Toint [31]) or the bound-constrained one (see Powell [30], Lewis and Torczon [16], Gratton, Toint and Tröltzsch [15] and Tröltzsch [35]). Problems with more general linear or nonlinear constraints are understudied: besides the contributions of Powell [25], Lewis and Torczon [17, 18, 19] and Colson [1], very little is known and many algorithmic possibilities have not been considered as yet, either theoretically or practically. In this work, we consider a derivative-free adaptation with self-correcting geometry steps of the trust-funnel method presented by Gould and Toint [13] for the solution of the equality-constrained nonlinear optimization problem

$$\left\{ \begin{array}{ll} \min & f(x) \\ \text{s.t.:} & c(x) = 0, \end{array} \right. \tag{1.1}$$

where we assume that $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ are twice continuously differentiable, and that f is bounded below on the feasible domain.

Within the range of DFO methods devised thus far, three main classes may be distinguished, namely: direct-search methods, derivative estimation by finite differences and model-based algorithms. The first class, also called *zero-order methods*, is rooted on the exploration of the variable space by generating a set of trial points at each iteration and having their function values compared to the best solution previously obtained. The popularity of this class of methods is ascribed to its simplicity and robustness. The fact that no assumption of smoothness of the objective function is demanded makes it applicable to a wide range of problems. Nevertheless, a relatively large number of function evaluations is often performed. Besides that, the number of evaluations rapidly increases as the number of variables grows. Among the methods of this class, is the popular *Nelder-Mead* algorithm or *simplex search* algorithm (see [22]).

The second class of methods concerns the use of finite differences along with quasi-Newton methods (we refer the reader to the papers by Stewart [33] and Moré [21], and the textbooks by Dennis and Schnabel [7], Gill, Murray and Wright [11], and Nocedal and Wright [23]). An inherent drawback of this approach is in cases where the function evaluations are costly, for a single gradient estimation

[*]Namur Center for Complex Systems (naXys) and Department of Mathematics, University of Namur, 61, rue de Bruxelles, B-5000 Namur, Belgium. Email: phillipe.sampaio@unamur.be

[†]Namur Center for Complex Systems (naXys) and Department of Mathematics, University of Namur, 61, rue de Bruxelles, B-5000 Namur, Belgium. Email: philippe.toint@unamur.be

requires at least the number of variables plus one function evaluations. Another one it is when the functions are noisy, often making the gradient estimation useless.

The third class was introduced by Winfield [36, 37] with the minimization of quadratic interpolation models in a neighbourhood of the current iteration where the models are assumed to be valid. Later, Powell [25, 26] proposed COBYLA, a method for constrained optimization, which supports arbitrary nonlinear inequality and equality constraints by using linear multivariate interpolation-based models for both of them in a trust-region framework. The introduction of the criticality step and the first interpolation-based derivative-free trust-region method with global convergence to first-order critical points is due to Conn, Scheinberg and Toint [3]. Since the cost of maintaining a valid interpolation model — i.e., to keep a geometrically reasonable sample set — is expensive, Fasano, Nocedal and Morales [9] suggested to ignore any geometry control and obtained good performance in practice. Later, Scheinberg and Toint [31] proved that the resulting method may loose the property of global convergence to first-order critical points and showed that maintaining the quality of the geometry of the interpolation set is critical for maintaing this desirable property. They then proposed a suitable choice of interpolation points yielding a self-correcting geometry scheme, which is the cornerstone of our derivative-free version of the trust-funnel method for nonlinear programming problems.

For a thorough survey on direct search, interpolation models and other derivative-free methods, we refer the reader to the paper by Lewis, Torczon and Trosset [20] and to the textbooks of Conn, Scheinber and Vicente [5] and Conn, Gould and Toint [2].

The algorithm developed in this work, named *DEFT-FUNNEL* (DErivative-Free Trust FUNNEL), belongs to the third class of methods, i.e., it is based on models built by multivariate interpolation of the objective and the constraint functions. Since our goal is to have a trust-funnel method for solving DFO problems, the main features of the proposal by Gould and Toint [13] are thus preserved:

- independence of the objective function and constraints by using different models and trust regions for f and c;
- a sequential quadratic programming (SQP) approach to compute the step;
- the flexibility resulting from the fact that the algorithm doesn't necessarily compute both normal and tangent steps at every iteration (the computation is done for whichever is/are likely to improve feasibility and optimality significantly);
- the specific nature of the algorithm which uses neither merit functions (penalty or otherwise) — thereby avoiding practical issues with regard to parameters control — nor the filter technique proposed by Fletcher and Leyffer [10];
- the trust-funnel driven convergence — the gist of the algorithm and what makes it different from other composite-step approaches —, a progressively decreasing limit on the permitted infeasibility of the successive iterates.

The outline of this paper is as follows. Section 2 introduces some fundamental points of multivariate polynomial interpolation, showing how to build the surrogate models to be optimized. Section 3 then addresses the derivative-free trust-funnel algorithm itself, while in Section 4 we detail our numeric experiments. Finally, we draw some final conclusions in Section 5.

Notation. Unless otherwise specified, our norm $\|\cdot\|$ is the standard Euclidean norm. Given any set $\mathcal{S}$, $|\mathcal{S}|$ denotes the cardinality of $\mathcal{S}$. We let $\mathcal{B}$ denote a closed ball in $\mathbb{R}^n$ and $\mathcal{B}(z;\Delta)$ denote the closed ball centered at z, with radius $\Delta > 0$. By $\mathcal{P}_n^d$, we mean the space of all polynomials of degree $\leq d$ in $\mathbb{R}^n$.

2 Multivariate interpolation

Before going further into details of the algorithm, we first introduce some concepts and results of multivariate polynomial interpolation that we make use throughout and that can be found to a more extent in Conn, Scheinberg and Vicente [5].

Definition 2.1. *Let $\Lambda > 0$ and $\mathcal{P}$ be a space of polynomials on $\mathbb{R}^n$ with a basis $\varphi = \{\varphi_0(x), \varphi_1(x), \ldots, \varphi_p(x)\}$. Then, a set $\mathcal{Y} = \{y^0, y^1, \ldots, y^p\}$ is said to be Λ-poised in $\mathcal{B}$ for $\mathcal{P}$ (in the interpolation sense) with respect to the basis φ if and only if for any $x \in \mathcal{B} \subset \mathbb{R}^n$ there exists $\lambda(x) \in \mathbb{R}^{p+1}$ such that:*

$$\sum_{i=0}^{p} \lambda_i(x)\varphi(y^i) = \varphi(x) \quad \textit{with} \quad \|\lambda(x)\|_\infty \leq \Lambda.$$

The algorithm developed in this work employs the commonly used idea of starting with incomplete interpolation models with linear accuracy and then enhancing them with curvature information, thereby having an actual accuracy at least as good as that for linear models and, hopefully, better. We thus consider underdetermined quadratic interpolation — i.e., $n+1 \leq |\mathcal{Y}| \leq (n+1)(n+2)/2$ — with initial sample sets that are poised for linear interpolation. Since the bound in the underdetermined case depends on the basis used, we adopt the natural basis in our definition.

We now have the following interpolation linear system to solve in order to build our model:

$$M(\phi, \mathcal{Y})\alpha_\phi = f(\mathcal{Y}), \tag{2.2}$$

where

$$M(\phi,\mathcal{Y}) = \begin{bmatrix} \phi_0(y^0) & \phi_1(y^0) & \cdots & \phi_q(y^0) \\ \phi_0(y^1) & \phi_1(y^1) & \cdots & \phi_q(y^1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(y^p) & \phi_1(y^p) & \cdots & \phi_q(y^p) \end{bmatrix}, \quad f(\mathcal{Y}) = \begin{bmatrix} f(y^0) \\ f(y^1) \\ \vdots \\ f(y^p) \end{bmatrix}, \quad p < q.$$

Since the linear system (2.2) is potentially underdetermined, the resulting interpolating polynomials will be no longer unique, and there are different ways to construct the model $m(x)$.

One simple approach, called *subbasis selection*, consists in considering only $p+1$ columns of the matrix $M(\phi, \mathcal{Y})$ in the linear system (2.2), having then a square matrix. Basically, this means that one is choosing a *subbasis* $\tilde{\phi}$ of ϕ with only $p+1$ elements. By removing the other $q-p$ columns, it will cause $q-p$ elements of α_ϕ to be zero. An inherent shortcoming of this approach is that the selected $p+1$ columns might be linearly dependent, which could be taken as a lack of poisedness of the sample set and thus cause the change of some of the points in the set. This may happen even if selecting a different set of columns could have provided well poisedness without changing any point. Information about f, e.g., sparsity structure of the derivatives, can be taken into account in the choice of a set of columns over another.

Another way to build the model is to take the minimum ℓ_2-norm solution of (2.2). In Conn, Scheinberg and Vicente [5], the authors show that this approach is more robust with respect to small perturbations of the data than choosing a subbasis. As expected, it also resulted in better performance than the subbasis approach in our experiments, as it will be shown in due course.

A third possibility stems from the fact that we want to build models for which the norm of the Hessian is moderate, as it plays a relevant role on the error bounds for quadratic interpolation models (see Theorem 5.4 in [5]). For that purpose, the interpolation model can be written as

$$m(x) = \alpha_L^T \phi_L(x) + \alpha_Q^T \phi_Q(x),$$

where α_L and ϕ_L are related to the linear components of the natural basis ϕ, while α_Q and ϕ_Q, to the quadratic ones. The minimum Frobenius norm solution α^{mfn} is then the solution of the following optimization problem in α_L and α_Q:

$$\begin{array}{ll} \min & \frac{1}{2}\|\alpha_Q\|_2^2 \\ \text{s.t.:} & M(\phi_L, \mathcal{Y})\alpha_L + M(\phi_Q, \mathcal{Y})\alpha_Q = f(\mathcal{Y}). \end{array}$$

Due to the choice of the basis ϕ and the separation $\alpha = (\alpha_L, \alpha_Q)$, minimizing the norm of α_Q is equivalent to minimizing the Frobenius norm of the Hessian of $m(x)$, thereby originating the name of the model.

Throughout the paper, we also require the following assumption, which is readily achieved, for instance, by applying the procedures described in Conn, Scheinberg and Vicente [5] once the interpolant model is built with the Lagrange polynomials.

Assumption 2.2. *Assume we are given any set $\mathcal{Y} \subset \mathcal{B}(z, \Delta)$ with $n+1 \leq |\mathcal{Y}| \leq (n+1)(n+2)/2$ and $z \in \mathbb{R}^n$. Then we can apply a finite number of substitutions of the points in $\mathcal{Y}$, in fact, at most $|\mathcal{Y}|-1$, such that the new resultant set is Λ-poised in $B(z, \Delta)$ for a polynomial space $\mathcal{P}$, with dimension $|\mathcal{Y}|$ and $\mathcal{P}_n^1 \subseteq \mathcal{P} \subseteq \mathcal{P}_n^2$.*

The next lemma states the bounds for at most fully quadratic models. As one might expect, the error bounds are at least as good as those for the linear interpolation case, thus being linear in Δ for the first derivatives and quadratic for the function values. The proof of the lemma can be found in [6].

Lemma 2.3. Given any $\Delta > 0$, $z \in \mathbb{R}^n$ and $\mathcal{Y} = \{0, y^1, \ldots, y^p\} \subset \mathcal{B}(z, \Delta)$ a Λ-poised in $\mathcal{B}(z, \Delta)$ with $n+1 \leq |\mathcal{Y}| \leq (n+1)(n+2)/2$, let $p(\cdot) \in \mathcal{P}_n^2$ be a interpolating polynomial of f on $\mathcal{Y}$, i.e.

$$p(y^i) = f(y^i), \quad i = 1, \ldots, |\mathcal{Y}|.$$

If $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and ∇f is Lipschitz continuous with constant L in an open set containing $\mathcal{B}(z, \Delta)$, then for any $s \in \mathcal{B}(z, \Delta)$ we have

$$\begin{array}{rcl} \|\nabla f(z+s) - \nabla p(z+s)\| & \leq & \hat{\kappa}_{eg}(n, \Lambda, L)(\|\nabla^2 p\| + 1)\Delta, \\ |f(z+s) - p(z+s)| & \leq & \hat{\kappa}_{ef}(n, \Lambda, L)(\|\nabla^2 p\| + 1)\Delta^2, \end{array}$$

where $\hat{\kappa}_{eg}$ and $\hat{\kappa}_{ef}$ are positive constants depending only on n, Λ and L.

In our experiments, we also exploit regression models whenever the number of sampling points is greater than the necessary for complete quadratic interpolation. Similar error bounds for the regression case can be found in [5].

3 A derivative-free trust-funnel method

After having presented different DFO techniques from which our algorithm gleans, we explain now how the DEFT-FUNNEL assembles them all into the trust funnel framework.

Given a poised set of sample points $\mathcal{Y}_0 = \{y^0, y^1, \ldots, y^p\}$ with an initial point $x_0 \in \mathcal{Y}_0$, the first step of our algorithm is to replace the functions $f(x)$ and $c(x) = (c_1(x), c_2(x), \ldots, c_m(x))$ by surrogate models m^f and $m^c(x) = (m_1^c(x), m_2^c(x), \dots, m_m^c(x))$ built from polynomial interpolation, having therefore the following required interpolation conditions being satisfied

$$\begin{aligned} m^f(y^i) &= f(y^i), \\ m_j^c(y^i) &= c_j(y^i), \end{aligned} \tag{3.3}$$

for all $y^i \in \mathcal{Y}_0$.

At each iteration k, we have $x_k \in \mathcal{Y}_k$. Whenever $\mathcal{Y}_k$ changes, the models m_k^c and m_k^f are updated to satisfy the interpolation conditions (3.3) for the new set $\mathcal{Y}_{k+1}$, thereby implying that new function evaluations of f and c are carried out for the additional points obtained at iteration k.

As mentioned previously, our method employs a composite-step approach of the type suggested by Omojokun [24] in his doctoral thesis under the supervision of R. H. Byrd. In this technique, each full SQP (*Sequential Quadratic Programming*) step is decomposed as

$$s_k = n_k + t_k,$$

where the *normal step* component n_k aims to improve feasibility and the *tangent step* component t_k reduces the model while preserving any gains in feasibility obtained through n_k. We now show how each component is computed in our algorithm.

3.1 The normal step

We measure the constraint violation at any point x by

$$\theta(x) \stackrel{\text{def}}{=} \tfrac{1}{2}\|c(x)\|^2 \tag{3.4}$$

where $\|\cdot\|$ denotes the Euclidean norm.

If, at iteration k, the constraint violation is significant with respect to the measure of optimality, yet to be defined, a normal step n_k is computed by reducing the Gauss-Newton model of $\theta(x_k + n_k)$ within a trust region, i.e., by solving the following trust-region linear least-squares problem

$$\left\{ \begin{array}{ll} \min & \tfrac{1}{2}\|c_k + J_k n\|^2 \\ \text{s.t.:} & \quad n_k \in \mathcal{N}_k, \end{array} \right. \tag{3.5}$$

where $c_k \stackrel{\text{def}}{=} c(x_k) = m_k^c(x_k)$, $J_k \stackrel{\text{def}}{=} J(x_k)$ is the Jacobian of m_k^c at x_k and

$$\mathcal{N}_k \stackrel{\text{def}}{=} \{v \in \mathbb{R}^n \mid \|v\| \le \Delta_k^c\}, \tag{3.6}$$

for some radius $\Delta_k^c > 0$. We do not require an exact Gauss-Newton step, rather the computed step n_k must reduce sufficiently the model within $\mathcal{N}_k$ in the sense that it satisfies the Cauchy condition for the problem (3.5)

$$\delta_k^{c,n} \stackrel{\text{def}}{=} \tfrac{1}{2}\|c_k\|^2 - \tfrac{1}{2}\|c_k + J_k n_k\|^2 \ge \kappa_{nC}\|J_k^T c_k\| \min\left[\frac{\|J_k^T c_k\|}{1 + \|W_k\|}, \Delta_k^c\right] \ge 0, \tag{3.7}$$

where $W_k = J_k^T J_k$ is the symmetric Gauss-Newton approximation of the Hessian of

$$\chi_k(x) \stackrel{\text{def}}{=} \tfrac{1}{2}\|m_k^c(x)\|^2 \tag{3.8}$$

at x_k and $\kappa_{nC} \in (0, \tfrac{1}{2}]$.

Since there is nothing assuring that the normal step is indeed "normal", we add the following condition requiring that it mostly lies in the space spanned by the columns of the matrix J_k^T

$$\|n_k\| \le \kappa_n \|c_k\|, \tag{3.9}$$

for some $\kappa_n \ge 1$. Such imposed conditions are very reasonable in practice, being satisfied, for instance, if one applies a truncated conjugate-gradient method (see [32] and [34]). If x_k is feasible, we choose a null normal step ($n_k = 0$).

We also have a bound $\theta_k^{\max}$, the so-called "funnel" that names the method, such that, for each iteration k,

$$\chi_k(x_k) = \theta(x_k) < \theta_k^{\max},$$

where the first equality follows from the fact the interpolation conditions (3.3) are satisfied for each $x_k \in \mathcal{Y}_k$. This bound is monotonically decreased as the algorithm is driven towards feasibility, driving the convergence of the algorithm.

3.2 The tangent step

Once the normal step n_k has been calculated, the computation of the tangent step to reduce the model m_k^f is carried out while care is taken to not deteriorate the improvement on feasibility obtained through the former.

We define then the quadratic model function

$$\psi_k(x_k + s) \stackrel{\text{def}}{=} f_k + \langle g_k, s\rangle + \tfrac{1}{2}\langle s, G_k s\rangle, \tag{3.10}$$

where $f_k \stackrel{\text{def}}{=} f(x_k) = m_k^f(x_k)$, $g_k \stackrel{\text{def}}{=} \nabla m_k^f(x_k)$ and G_k is a symmetric approximation of the Hessian of the Lagrangian $\mathcal{L}(x, y) = m_k^f(x) + \langle \mu, m_k^c(x)\rangle$ given by

$$G_k \stackrel{\text{def}}{=} H_k + \sum_{i=1}^{m} [\hat{\mu}_k]_i C_{ik}. \tag{3.11}$$

In the last definition, H_k is a bounded symmetric approximation of $\nabla^2 m_k^f(x_k)$, the matrices C_{ik} are bounded symmetric approximations of the constraints' models Hessians $\nabla^2 m_{ik}^c(x_k)$ and the vector $\hat{\mu}_k$ may be viewed as a bounded local approximation of the Lagrange multipliers, in the sense that we require that

$$\|\hat{\mu}_k\| \le \kappa_\mu, \tag{3.12}$$

for some $\kappa_\mu > 0$.

By using the decomposition $s_k = n_k + t_k$, we have

$$\psi_k(x_k + n_k) = f_k + \langle g_k, n_k\rangle + \tfrac{1}{2}\langle n_k, G_k n_k\rangle \tag{3.13}$$

and

$$\begin{aligned} \psi_k(x_k + n_k + t) &= f_k + \langle g_k, n_k + t\rangle + \tfrac{1}{2}\langle n_k + t, G_k(n_k + t)\rangle \\ &= \psi_k(x_k + n_k) + \langle g_k^n, t\rangle + \tfrac{1}{2}\langle t, G_k t\rangle \end{aligned} \tag{3.14}$$

where

$$g_k^n \stackrel{\text{def}}{=} g_k + G_k n_k. \tag{3.15}$$

We have thus that (3.14) a quadratic model of the function $m_k^f(x_k + n_k + t)$. In the interest of assuring that it is a proper local approximation, the complete step $s = n_k + t$ must belong to

$$\mathcal{T}_k \overset{\text{def}}{=} \{s \in \mathbb{R}^n \mid \|s\| \leq \Delta_k^f\}, \tag{3.16}$$

for some radius Δ_k^f. The minimization of (3.14) should then be restricted to the intersection of $\mathcal{N}_k$ and $\mathcal{T}_k$, which imposes that the *tangent step* t_k results in a complete step $s_k = n_k + t_k$ that satisfies the inclusion

$$s_k \in \mathcal{R}_k \overset{\text{def}}{=} \mathcal{N}_k \cap \mathcal{T}_k \overset{\text{def}}{=} \{s \in \mathbb{R}^n \mid \|s\| \leq \Delta_k\}, \tag{3.17}$$

where the radius Δ_k of $\mathcal{R}_k$ is thus given by

$$\Delta_k = \min[\Delta_k^c, \Delta_k^f]. \tag{3.18}$$

Due to (3.17), we ask n_k to belong to $\mathcal{R}_k$ before attempting the computation of t_k by requiring that

$$\|n_k\| \leq \kappa_B \Delta_k, \tag{3.19}$$

for some $\kappa_B \in (0, 1)$. If (3.19) happens, which means that there is "enough space left" to make another step without crossing the trust region border, the tangent step is finally computed by (approximately) solving the following problem

$$\left\{\begin{array}{ll} \min & \langle g_k^n, t\rangle + \frac{1}{2}\langle t, G_k t\rangle \\ \text{s.t.:} & \qquad J_k t = 0, \\ & \quad \|n_k + t\| \leq \Delta_k. \end{array}\right. \tag{3.20}$$

In practice, we do not solve the problem (3.20) exactly, rather we only require a "sufficient" reduction of (3.14) within the hyperplane tangent to the constraints intersected to the trust region. Note that the original proposal of Gould and Toint [13] also allows for "tangent" steps which do not lie exactly in the nullspace of the Jacobian J_k, but we neglect this possibility here because the computation of exactly tangent steps (i.e. satisfying $J_k t = 0$) is acceptable in the context of small-scale problems. In order to compute an approximate projected gradient at $x_k + n_k$, we first compute a new local estimate of the Lagrange multipliers μ_k by solving the least-squares problem

$$\min_{\mu} \tfrac{1}{2}\|g_k^n + J_k^T \mu\|^2. \tag{3.21}$$

Since the number of variables of the problems considered in the DFO context are usually small, we allow (3.21) to be solved exactly in our experiments, although again only an approximation could be required. The orthogonal projection of g_k^n onto the nullspace of J_k is then denoted by

$$r_k \overset{\text{def}}{=} g_k^n + J_k^T \mu_k, \tag{3.22}$$

which motivates that we require the tangent step to produce a reduction in the model ψ_k which is at least a fraction of that achieved by solving the modified Cauchy point subproblem

$$\min_{\substack{\tau > 0 \\ x_k + n_k - \tau r_k \in \mathcal{R}_k}} \psi_k(x_k + n_k - \tau r_k), \tag{3.23}$$

where we have assumed that $\|r_k\| > 0$. We know from Section 8.1.5 of [2] that this procedure ensures, for some $\kappa_{tC1} \in (0, 1]$, the modified Cauchy condition

$$\delta_k^{f,t} \overset{\text{def}}{=} \psi_k(x_k + n_k) - \psi_k(x_k + n_k + t_k) \geq \kappa_{tC1} \pi_k \min\left[\frac{\pi_k}{1 + \|G_k\|}, \tau_k \|r_k\|\right] > 0 \tag{3.24}$$

on the decrease of the objective function model within $\mathcal{R}_k$, where we have set

$$\pi_k \stackrel{\text{def}}{=} \frac{\langle g_k^n, r_k \rangle}{\|r_k\|} \geq 0 \tag{3.25}$$

(by convention, we define $\pi_k = 0$ whenever $r_k = 0$), and where τ_k is the maximal step length along $-r_k$ from $x_k + n_k$ which remains in the trust-region $\mathcal{R}_k$. But we have that

$$\tau_k \|r_k\| \geq (1 - \kappa_B)\Delta_k$$

by construction and thus the modified Cauchy condition (3.24) may now be rewritten as

$$\delta_k^{f,t} \stackrel{\text{def}}{=} \psi_k(x_k + n_k) - \psi_k(x_k + n_k + t_k) \geq \kappa_{tC} \pi_k \min\left[\frac{\pi_k}{1 + \|G_k\|}, \Delta_k\right] \tag{3.26}$$

with $\kappa_{tC} \stackrel{\text{def}}{=} \kappa_{tC1}(1 - \kappa_B) \in (0, 1)$. We see from (3.26) that π_k may be considered as an optimality measure in the sense that it measures how much decrease could be obtained locally along the negative of the approximate projected gradient r_k.

To solve the subproblem (3.20), a truncated projected conjugate gradient method might be used. In order to avoid significant rounding errors in the computation of the tangent step, a stable version of such method described in Gould, Hribar and Nocedal [14] is used in our implementation.

3.3 Which steps to compute and retain

When $r_k = 0$, the computation of tangent step is not needed, and we simply define $\pi_k = 0$ and $t_k = 0$. If π_k is small compared to the current infeasibility, i.e., for a given a monotonic bounding function ω_t, the condition

$$\pi_k > \omega_t(\|c_k\|) \tag{3.27}$$

fails, then the current iterate is still too far from feasibility to worry about optimality, and we again skip the tangent step computation by setting $t_k = 0$.

The normal step, in turn, is computed when $k = 0$ or the current violation is "significant", which is now formally defined by the conditions

$$\|c_k\| > \omega_n(\pi_{k-1}) \text{ or } \theta_k > \kappa_{\theta\theta}\theta_k^{\max}, \tag{3.28}$$

where ω_n is some bounding function, $\kappa_{\theta\theta} \in (0, 1)$ is a constant and $\theta_k \stackrel{\text{def}}{=} \theta(x_k)$. If (3.28) fails, the computation of the normal step is not required and so we set $n_k = 0$. We also require that

$$\omega_n(t) = 0 \Longleftrightarrow t = 0 \text{ and } \omega_t(\omega_n(t)) \leq \kappa_\omega t, \tag{3.29}$$

for all $t \geq 0$ and for some $\kappa_\omega \in (0, 1)$. An example for the function $\omega_t(z)$, which is used in our algorithm, might be

$$\omega_t(z) \stackrel{\text{def}}{=} 0.01 \min[1,\ z]. \tag{3.30}$$

We say that x_k is an infeasible stationary point if $c(x_k) \neq 0$ and $J(x_k)^T c_k = 0$. While (3.27) and (3.28) together provide considerable flexibility in our algorithm in that a normal or tangent step is only computed when relevant, our setting also produces the possibility that both these conditions fail. In this case, we have that $s_k = n_k + t_k$ is identically zero, and the sole computation in the iteration is that of the new Lagrange multiplier μ_k.

The conditions (3.29) are important to prove the following lemma, which can be found in [13].

Lemma 3.1. For all k such that $s_k = 0$ and x_k is not an infeasible stationary point,

$$\pi_k \leq \kappa_\omega \pi_{k-1}.$$

It is argued in that reference that this lemma, the initial assumption that is f bounded below on the feasible domain and the fact the the algorithm terminates when an infeasible stationary point is found imply that such behavior ($s_k = 0$) cannot persist unless x_k is optimal.

Finally, we may evaluate the usefulness of the tangent step t_k after (or during) its computation, in the sense that we would like a relatively large tangent step to cause a clear decrease in the model (3.14) of the objective function. We therefore check whether the conditions

$$\|t_k\| > \kappa_{\mathcal{CS}} \|n_k\| \tag{3.31}$$

and

$$\delta_k^f \stackrel{\text{def}}{=} \delta_k^{f,t} + \delta_k^{f,n} \geq \kappa_\delta \delta_k^{f,t} \tag{3.32}$$

are satisfied for some $\kappa_{\mathcal{CS}} > 1$ and for $\kappa_\delta = 1 - 1/\bar{\kappa}_\delta \in (0, 1)$. The latter inequality indicates that the *predicted* improvement in the objective function obtained in the tangent step is not negligible compared to the *predicted* change in f resulting from the normal step. If (3.31) holds but (3.32) fails, the tangent step is not useful in the sense discussed at the beginning of this paragraph, and we choose to ignore it by resetting $t_k = 0$.

3.4 Iterations types

Once we have computed the step s_k and the trial point

$$x_k^+ \stackrel{\text{def}}{=} x_k + s_k \tag{3.33}$$

completely, we are left with the task of accepting or rejecting it. If $n_k = t_k = 0$, iteration k is said to be a μ-iteration because the only computation potentially performed is that of a new vector of Lagrange multiplier estimates. We will say that iteration k is an f-iteration if $t_k \neq 0$, (3.32) holds, and

$$\theta(x_k^+) \leq \theta_k^{\max}. \tag{3.34}$$

Condition (3.34) ensures that the step keeps feasibility within reasonable bounds. Thus the iteration's expected major achievement is, in this case, a decrease in the value of the objective function f, hence its name. If iteration k is neither a μ-iteration nor a f-iteration, then it is said to be a c-iteration. If (3.32) fails, then the expected major achievement (or failure) of iteration k is, *a contrario*, to improve feasibility, which is also the case when the step only contains its normal component.

The main idea behind the technique for accepting the trial point is to measure whether the major expected achievement of the iteration has been realized.

- If iteration k is a μ-iteration, we do not have any other choice than to restart with $x_{k+1} = x_k$ using the new multipliers. We then define

$$\Delta_{k+1}^f = \Delta_k^f \text{ and } \Delta_{k+1}^c = \Delta_k^c \tag{3.35}$$

  and keep the current value of the maximal infeasibility $\theta_{k+1}^{\max} = \theta_k^{\max}$.

- If iteration k is an f-iteration, we accept the trial point (i.e., $x_{k+1} = x_k^+$) if

$$\rho_k^f \overset{\text{def}}{=} \frac{f(x_k) - f(x_k^+)}{\delta_k^f} \geq \eta_1, \tag{3.36}$$

  and reject it (i.e., $x_{k+1} = x_k$), otherwise. The value of the maximal infeasibility measure is left unchanged, that is $\theta_{k+1}^{\max} = \theta_k^{\max}$. Note that $\delta_k^f > 0$ (because of (3.26) and (3.32)) unless x_k is first-order critical, and hence that condition (3.36) is well-defined.

- If iteration k is a c-iteration, we accept the trial point if the improvement in feasibility is comparable to its predicted value

$$\delta_k^c \overset{\text{def}}{=} \tfrac{1}{2}\|c_k\|^2 - \tfrac{1}{2}\|c_k + J_k s_k\|^2,$$

  and the latter is itself comparable to its predicted decrease along the normal step, that is

$$n_k \neq 0, \delta_k^c \geq \kappa_{cn}\delta_k^{c,n} \text{ and } \rho_k^c \overset{\text{def}}{=} \frac{\theta(x_k) - \theta(x_k^+)}{\delta_k^c} \geq \eta_1 \tag{3.37}$$

  for some $\kappa_{cn} \in (0, 1 - \kappa_{tg}]$. If (3.37) fails, the trial point is rejected. We update the value of the maximal infeasibility by

$$\theta_{k+1}^{\max} = \begin{cases} \max\left[\kappa_{tx1}\theta_k^{\max}, \theta(x_k^+) + \kappa_{tx2}(\theta(x_k) - \theta(x_k^+))\right] & \text{if (3.37) hold,} \\ \theta_k^{\max} & \text{otherwise,} \end{cases} \tag{3.38}$$

  for some $\kappa_{tx1} \in (0, 1)$ and $\kappa_{tx2} \in (0, 1)$.

We now describe why the last condition in (3.37) is well-defined. Firstly, we only check the third condition *after* the first two conditions have been verified. Assuming that $n_k \neq 0$, the Cauchy condition (3.7) and $c(x_k) \neq 0$ ensure that $\delta_k^{c,n} > 0$ provided $J_k^T c_k \neq 0$. Thus the third condition is well defined, unless x_k is an infeasible stationary point, in which case the algorithm is terminated.

3.5 Self-correcting geometry scheme

In [31], Scheinberg and Toint looked further into the question of whether it is possible to ignore geometry considerations and still have a globally convergent algorithm. Their solution relies mainly on the idea that one can still use a trial point x_k^+ to maintain the quality of geometry at unsuccessful iterations. Instead of using costly model improvement steps to obtain new points at such iterations, which might require another optimization problem to be globally solved, they proposed to try first to replace an interpolation point that is far from the current point x_k by the trial point x_k^+. If there is no such a far point, one try to replace an interpolation point $y^{k,j}$ whose associated Lagrange polynomial value at x_k^+, $\ell_{k,j}(x_k^+)$, is bigger than a predefined threshold Λ. By that means, one attempt to obtain a Λ-poised set $\mathcal{Y}_{k+1}$, or, since at most one interpolation point is replaced by iteration, improve its poisedness, at least. If no point can be replaced, the trust region is then reduced.

A slight modification on the scheme is used on DEFT-FUNNEL. Originally, the trust region is only reduced at unsuccessful iterations where it is impossible to improve poisedness by replacing the suitable choice of interpolation points aforementioned. In our method, and following Gratton, Toint and Tröltzsch [15], it can also be reduced in cases where it is possible to improve poisedness.

3.6 The algorithm

We are now ready to state our complete algorithm, which puts the above discussion in a more formal context.

Algorithm 3.1: DEFT-FUNNEL

Step 0: Initialization. An initial accuracy threshold ϵ_0, an initial vector of multipliers μ_{-1} and positive initial trust-region radii Δ_0^f and Δ_0^c are given, as well as the constants

$$\alpha \in (0,1), \quad 0 < \gamma_1 < 1 < \gamma_2, \quad \zeta \geq 1, \quad 0 < \eta_1 < \eta_2 < 1 \quad \text{and} \quad \beta, \eta_3 > 0.$$

An initial set of interpolation points is also given, $\mathcal{Y}_0$, with $x_0 \in \mathcal{Y}_0 \subset \mathcal{B}(x_0, \Delta_0)$ and $|\mathcal{Y}_0| \geq n+1$, as well as the maximum number of interpolation points $p_{\max} \geq |\mathcal{Y}_0|$ in $\mathcal{Y}_k$ at the end. Let p_0 denote the cardinality of $\mathcal{Y}_0$. This interpolation set defines interpolation models m_0^f and m_0^c around x_0 and associated Lagrange polynomials $\{l_{0,j}\}_{j=0}^p$. Define $\Delta_0 = \min[\Delta_0^f, \Delta_0^c] \leq \Delta^{\max}$, and $\theta_0^{\max} = \max[\kappa_{ca}, \kappa_{cr}\theta(x_0)]$ for some constants $\kappa_{ca} > 0$ and $\kappa_{cr} > 1$. Define $\nu_f^{\max} > 0$ and $\nu_c^{\max} > 0$, the maximum number of times that the tangential and normal trust regions sizes can be reduced when an interpolation point is replaced at unsuccessful iterations. Initialize the corresponding counters $\nu_f = \nu_c = 0$. Define $k = 0$ and $i = 0$.

Step 1: Criticality step. Define $\hat{m}_i^f = m_k^f$, $\hat{m}_i^c = m_k^c$ and $\nabla \mathcal{L}_k = \nabla g_k + J_k^T \mu_k$.

Step 1.1: If $\|c_k\| \leq \epsilon_i$ and $\|\nabla \mathcal{L}_k\| \leq \epsilon_i$, set $\epsilon_{i+1} = \max[\alpha\|c_k\|, \alpha\|\nabla \mathcal{L}_k\|, \epsilon]$, modify $\mathcal{Y}_k$ as needed to ensure it is Λ-poised in $\mathcal{B}(x_k, \epsilon_{i+1})$ and increment i by one. If $\|c_k\| \leq \epsilon$ and $\|\nabla \mathcal{L}_k\| \leq \epsilon$, return x_k; otherwise, start Step 1.1 again;

Step 1.2: Set $m_k^f = \hat{m}_i^f$ and $m_k^c = \hat{m}_i^c$, $\Delta_k = \beta \max[\|c_k\|, \|\nabla \mathcal{L}_k\|]$ and define $v_i = x_k$ if a new model has been computed.

Step 2: Normal step. Compute a normal step n_k that sufficiently reduces the linearized infeasibility (in the sense that (3.7) holds), under the constraint that (3.6) and (3.9) also hold. This computation must be performed if $k = 0$ or if (3.28) holds when $k > 0$.

If n_k has not been computed, set $n_k = 0$.

Step 3: Tangent step. If (3.19) holds, then

Step 3.1: select a vector $\hat{\mu}_k$ satisfying (3.12) and define G_k by (3.11);

Step 3.2: compute μ_k by solving (3.21) and r_k by (3.22);

Step 3.3: if (3.27) holds, compute a tangent step t_k that sufficiently reduces the model (3.14) (in the sense that (3.26) holds) and such that the complete step $s_k = n_k + t_k$ satisfies (3.17).

If (3.19) fails, set $\mu_k = 0$. In this case, or if (3.27) fails, or if (3.31) holds but (3.32) fails, set $t_k = 0$ and $s_k = n_k$. In all cases, define $x_k^+ = x_k + s_k$.

Step 4: Conclude a μ-iteration. If $n_k = t_k = 0$, then

Step 4.1: set $x_{k+1} = x_k$;

Step 4.2: define $\Delta_{k+1}^f = \Delta_k^f$ and $\Delta_{k+1}^c = \Delta_k^c$;

Step 4.3: set $\theta_{k+1}^{\max} = \theta_k^{\max}$ and $\Delta_{k+1} = \min[\Delta_{k+1}^f, \Delta_{k+1}^c]$.

Step 5: Conclude an f-iteration. If $t_k \neq 0$ and (3.32) and (3.34) hold,

Step 5.1: Augment the interpolation set. If $p_k < p_{\max}$, then define $\mathcal{Y}_{k+1} = \mathcal{Y}_k \cup \{x_k^+\}$.

- If $\rho_k^f \geq \eta_1$, set $x_{k+1} = x_k^+$ and $\nu_f = 0$.
  If $\rho_k^f \geq \eta_2$, set $\Delta_{k+1}^f = \min[\max[\gamma_2\|s_k\|, \Delta_k^f], \Delta^{\max}]$; otherwise, set $\Delta_{k+1}^f = \Delta_k^f$.
  If $\theta(x_k^+) < \eta_3\, \theta_k^{\max}$, set $\Delta_{k+1}^c = \min[\max[\gamma_2\|n_k\|, \Delta_k^c], \Delta^{\max}]$; otherwise, set $\Delta_{k+1}^c = \Delta_k^c$.
- If $\rho_k^f < \eta_1$, set $x_{k+1} = x_k$ and $\Delta_{k+1}^c = \Delta_k^c$.
  If $\nu_f \leq \nu_f^{\max}$, set $\Delta_{k+1}^f = \gamma_1 \Delta_k^f$ and $\nu_f = \nu_f + 1$; otherwise, set $\Delta_{k+1}^f = \Delta_k^f$.

Step 5.2: Successful iteration. If $p_k = p_{\max}$, $\rho_k^f \geq \eta_1$, set $x_{k+1} = x_k^+$ and define $\mathcal{Y}_{k+1} = \mathcal{Y}_k \setminus \{y^{k,r}\} \cup \{x_k^+\}$ for

$$y^{k,r} = \arg \max_{y^{k,j} \in \mathcal{Y}_k} \|y^{k,j} - x_k^+\|^2 |\ell_{k,j}(x_k^+)|. \tag{3.39}$$

Set $\nu_f = 0$. If $\rho_k^f \geq \eta_2$, set $\Delta_{k+1}^f = \min[\max[\gamma_2\|s_k\|, \Delta_k^f], \Delta^{\max}]$; otherwise, set $\Delta_{k+1}^f = \Delta_k^f$. If $\theta(x_k^+) < \eta_3\, \theta_k^{\max}$, set $\Delta_{k+1}^c = \min[\max[\gamma_2\|n_k\|, \Delta_k^c], \Delta^{\max}]$; otherwise, set $\Delta_{k+1}^c = \Delta_k^c$.

Step 5.3: Replace a far interpolation point. If $p_k = p_{\max}$, $\rho_k^f < \eta_1$, either $x_k \neq v_i$ or $\Delta_k \leq \epsilon_i$, and the set

$$\mathcal{F}_k \stackrel{\text{def}}{=} \{y^{k,j} \in \mathcal{Y}_k \text{ such that } \|y^{k,j} - x_k\| > \zeta\Delta \text{ and } \ell_{k,j}(x_k^+) \neq 0\} \tag{3.40}$$

is non-empty, then define $x_{k+1} = x_k$, and set $\Delta_{k+1}^f = \gamma_1\|s_k\|$ if $\nu_f \leq \nu_f^{\max}$ or $\Delta_{k+1}^f = \Delta_k^f$ otherwise. Define $\mathcal{Y}_{k+1} = \mathcal{Y}_k \setminus \{y^{k,r}\} \cup \{x_k^+\}$, where

$$y^{k,r} = \arg \max_{y^{k,j} \in \mathcal{F}_k} \|y^{k,j} - x_k^+\|^2 |\ell_{k,j}(x_k^+)|. \tag{3.41}$$

If $\nu_f \leq \nu_f^{\max}$, update $\nu_f = \nu_f + 1$.

Step 5.4: Replace a close interpolation point. If $p_k = p_{\max}$, $\rho_k^f < \eta_1$, either $x_k \neq v_i$ or $\Delta_k \leq \epsilon_i$, the set $\mathcal{F}_k$ is empty, and the set

$$\mathcal{C}_k \stackrel{\text{def}}{=} \{y^{k,j} \in \mathcal{Y}_k \text{ such that } \|y^{k,j} - x_k\| \leq \zeta\Delta \text{ and } \ell_{k,j}(x_k^+) > \lambda\} \tag{3.42}$$

is non-empty, then define $x_{k+1} = x_k$ and set $\Delta_{k+1}^f = \gamma_1\|s_k\|$ if $\nu_f \leq \nu_f^{\max}$ or $\Delta_{k+1}^f = \Delta_k^f$ otherwise. Define $\mathcal{Y}_{k+1} = \mathcal{Y}_k \setminus \{y^{k,r}\} \cup \{x_k^+\}$, where

$$y^{k,r} = \arg \max_{y^{k,j} \in \mathcal{C}_k} \|y^{k,j} - x_k^+\|^2 |\ell_{k,j}(x_k^+)|. \tag{3.43}$$

If $\nu_f \leq \nu_f^{\max}$, update $\nu_f = \nu_f + 1$.

Step 5.5: Reduce the trust-region radius. If $p_k = p_{\max}$, $\rho_k^f < \eta_1$ and either $x_k = v_i$ and $\Delta_k^f > \epsilon_i$ or $\mathcal{F}_k \cup \mathcal{C}_k = \emptyset$, then define $x_{k+1} = x_k$, $\Delta_{k+1}^c = \Delta_k^c$, choose $\Delta_{k+1}^f = \gamma_1\|s_k\|$ and define $\mathcal{Y}_{k+1} = \mathcal{Y}_k$.

Step 5.6: Update the combined radius. Set $\Delta_{k+1} = \min[\Delta_{k+1}^f, \Delta_{k+1}^c]$ and $\theta_{k+1}^{\max} = \theta_k^{\max}$.

Step 6: Conclude a c-iteration. If either $n_k \neq 0$ and $t_k = 0$, or either one of (3.32) or (3.34) fails,

Step 6.1: Augment the interpolation set. If $p_k < p_{\max}$, then define $\mathcal{Y}_{k+1} = \mathcal{Y}_k \cup \{x_k^+\}$.

- If $\rho_k^c \geq \eta_1$, set $x_{k+1} = x_k^+$, $\Delta_{k+1}^f = \Delta_k^f$ and $\nu_c = 0$.
  If $\rho_k^c \geq \eta_2$, set $\Delta_{k+1}^c = \min[\max[\gamma_2 \|n_k\|, \Delta_k^c], \Delta^{\max}]$; otherwise, set $\Delta_{k+1}^c = \Delta_k^c$.
- If $\rho_k^c < \eta_1$, set $x_{k+1} = x_k$ and $\Delta_{k+1}^f = \Delta_k^f$.
  If $\nu_c \leq \nu_c^{\max}$, then, set $\Delta_{k+1}^c = \gamma_1 \|n_k\|$ if $\|n_k\| \neq 0$, and $\Delta_{k+1}^c = \gamma_1 \Delta_k^c$, otherwise ($\|n_k\| = 0$). Update $\nu_c = \nu_c + 1$. If $\nu_c > \nu_c^{\max}$, set $\Delta_{k+1}^c = \Delta_k^c$.

Step 6.2: Successful iteration. If $p_k = p_{\max}$, (3.37) holds, set $x_{k+1} = x_k^+$ and define $\mathcal{Y}_{k+1} = \mathcal{Y}_k \setminus \{y^{k,r}\} \cup \{x_k^+\}$ for

$$y^{k,r} = \arg \max_{y^{k,j} \in \mathcal{Y}_k} \|y^{k,j} - x_k^+\|^2 |\ell_{k,j}(x_k^+)|. \tag{3.44}$$

Set $\Delta_{k+1}^f = \Delta_k^f$ and $\nu_c = 0$. Set $\Delta_{k+1}^c = \min[\max[\gamma_2 \|n_k\|, \Delta_k^c], \Delta^{\max}]$ if $\rho_k^c \geq \eta_2$ or $\Delta_{k+1}^c = \Delta_k^c$ otherwise.

Step 6.3: Replace a far interpolation point. If $p_k = p_{\max}$, (3.37) fails, either $x_k \neq v_i$ or $\Delta_k \leq \epsilon_i$, and the set

$$\mathcal{F}_k \stackrel{\text{def}}{=} \{y^{k,j} \in \mathcal{Y}_k \text{ such that } \|y^{k,j} - x_k \mid > \zeta \Delta \text{ and } \ell_{k,j}(x_k^+) \neq 0\} \tag{3.45}$$

is non-empty, then define $x_{k+1} = x_k$ and set $\Delta_{k+1}^f = \Delta_k^f$. If $\nu_c \leq \nu_c^{\max}$, then set $\Delta_{k+1}^c = \gamma_1 \|n_k\|$ if $\|n_k\| \neq 0$, or $\Delta_{k+1}^c = \gamma_1 \Delta_k^c$ otherwise ($\|n_k\| = 0$). If $\nu_c > \nu_c^{\max}$, set $\Delta_{k+1}^c = \Delta_k^c$. Define $\mathcal{Y}_{k+1} = \mathcal{Y}_k \setminus \{y^{k,r}\} \cup \{x_k^+\}$, where

$$y^{k,r} = \arg \max_{y^{k,j} \in \mathcal{F}_k} \|y^{k,j} - x_k^+\|^2 |\ell_{k,j}(x_k^+)|. \tag{3.46}$$

If $\nu_c \leq \nu_c^{\max}$, update $\nu_c = \nu_c + 1$.

Step 6.4: Replace a close interpolation point. If $p_k = p_{\max}$, (3.37) fails, either $x_k \neq v_i$ or $\Delta_k \leq \epsilon_i$, the set $\mathcal{F}_k$ is empty, and the set

$$\mathcal{C}_k \stackrel{\text{def}}{=} \{y^{k,j} \in \mathcal{Y}_k \text{ such that } \|y^{k,j} - x_k\| \leq \zeta \Delta \text{ and } \ell_{k,j}(x_k^+) > \lambda\} \tag{3.47}$$

is non-empty, then set $x_{k+1} = x_k$ and $\Delta_{k+1}^f = \Delta_k^f$. If $\nu_c \leq \nu_c^{\max}$, then set $\Delta_{k+1}^c = \gamma_1 \|n_k\|$ if $\|n_k\| \neq 0$, or $\Delta_{k+1}^c = \gamma_1 \Delta_k^c$ otherwise ($\|n_k\| = 0$). If $\nu_c > \nu_c^{\max}$, set $\Delta_{k+1}^c = \Delta_k^c$. Define $\mathcal{Y}_{k+1} = \mathcal{Y}_k \setminus \{y^{k,r}\} \cup \{x_k^+\}$, where

$$y^{k,r} = \arg \max_{y^{k,j} \in \mathcal{C}_k} \|y^{k,j} - x_k^+\|^2 |\ell_{k,j}(x_k^+)|. \tag{3.48}$$

If $\nu_c \leq \nu_c^{\max}$, update $\nu_c = \nu_c + 1$.

Step 6.5: Reduce the trust-region radius. If $p_k = p_{\max}$, (3.37) fails and either $x_k = v_i$ and $\Delta_k^c > \epsilon_i$ or $\mathcal{F}_k \cup \mathcal{C}_k = \emptyset$, then set $x_{k+1} = x_k$ and $\Delta_{k+1}^f = \Delta_k^f$. If $\|n_k\| \neq 0$, set $\Delta_{k+1}^c = \gamma_1 \|n_k\|$, otherwise set $\Delta_{k+1}^c = \gamma_1 \Delta_k^c$. Define $\mathcal{Y}_{k+1} = \mathcal{Y}_k$.

Step 6.6: Update the combined radius and the maximal infasibility.
Set $\Delta_{k+1} = \min[\Delta_{k+1}^f, \Delta_{k+1}^c]$ and update $\theta_k^{\max}$ using (3.38).

Step 7: Update the model and Lagrange polynomials. If $\mathcal{Y}_{k+1} \neq \mathcal{Y}_k$, compute the interpolation models m_{k+1}^f and m_{k+1}^c around x_{k+1} using $\mathcal{Y}_{k+1}$ and the associated Lagrange polynomials $\{l_{k+1,j}\}_{j=0}^p$. Increment k by one and go to Step 1.

4 Numerical experiments

The DEFT-FUNNEL has been implemented in Matlab and some initial experiments were run with the CUTEst testing environment (see Gould, Orban and Toint [12]). The selected test set contains 29 small-scale equality-constrained optimization problems used by Colson [1] in his numerical tests, and our results are compared to those obtained with the software CDFO in this reference, which is a derivative-free adaptation of the Filter-SQP method, originally proposed by Fletcher and Leyffer [10], and to those obtained with the software COBYLA by Powell [25], a trust-region method for constrained problems that models the objective and constraint functions by linear interpolation. The criterion for comparison between the methods is solely based on the number of calls on the routine that evaluates the objective function and the constraints at the required points, which is motivated by the fact that these costs very often dominate those of all other algorithmic computations.

Since the stopping criteria present in the three methods differ from each other, we decided to make two types of comparison between them. In the first one, we used the built-in convergence conditions of each algorithm and attempted to balance them by varying their parameters' values. In the CDFO method, convergence is declared when both conditions $\|\nabla\mathcal{L}(x_k)\| \le 10^{-3}$ and $\|c(x_k)\| \le 10^{-5}$ are satisfied, which are also the default values in the original code. Since we use the same threshold ϵ for verifying feasibility and optimality in the DEFT-FUNNEL method, we picked up a value in between, $\epsilon = 10^{-4}$. We have noticed, however, that for most of the problems, we obtained $\|c(x_k)\| \le 10^{-5}$, or even better. In practice, the DEFT-FUNNEL also terminates when the sample set is well poised — i.e., the error between the models and the real functions is sufficiently small — and either $\Delta_k \le 10^{-7}$ or $\|s_k\| \le 10^{-7}$ occurs. The stopping criterion used in COBYLA, in turn, is related to trust-region size. Since it never decreases the trust-region bound without checking the geometry of the sample set first, it does not stop without having a certain degree of trust on the models as well. In our experiments, the final value for the trust-region bound in COYBLA was set to 10^{-4}, which is often a much weaker condition to declare convergence than those of DEFT-FUNNEL and CDFO methods.

As for the second type of comparison, much for benchmark purposes, we assume that the optimal objective function value f^* of each problem is known *a priori* and, thus, we declare convergence for a method at iteration k if and only if one has

$$|f(x_k) - f^*| \le 10^{-4} \text{ and } \|c(x_k)\| \le 10^{-4},$$

thereby providing a common criterion for optimality as well.

In the DEFT-FUNNEL method, we fixed the trust-region parameters to $\Delta_0 = 1$, $\eta_1 = 0.0001$, $\eta_2 = 0.9$, $\eta_3 = 0.5$, $\gamma_1 = 0.5$, $\gamma_2 = 2.5$ and $\Delta^{\max} = 10^{10}$. The parameter ζ used in the definition of the sets $\mathcal{F}_k$ and $\mathcal{C}_k$ of far points and close points, respectively, is set to $\zeta = 1$. For the limit number of times to reduce the trust regions sizes when a far or close interpolation point is replaced at unsuccessful iterations, we choose $\nu_f^{\max} = \nu_c^{\max} = 10$. Finally, we set $\epsilon_0 = 0.01$ as the initial value for the loop in the criticality step, $\alpha = 0.1$ and $\beta = 1$.

Tables reporting the number of function evaluations required by the methods to solve each one of the problems are given in the appendix. The first four tables, associated with the four different choices to build the models in our algorithm, namely, subbasis selection, minimum Frobenius norm, minimum ℓ_2-norm and regression approaches, concern the first type of comparison aforementioned. We present not only the number of function evaluations performed (Feval.T) but also the number of function evaluations required to reach a solution to the problem (Feval.S) for DEFT-FUNNEL and CDFO methods. The values on the column Feval.S are usually strictly smaller than those on Feval.T because the algorithm usually has to compute new function values to make the sample set well poised — as required to declare convergence — after having already encountered a solution. Finally, the fifth table contain the number of function evaluations to achieve convergence in the second type of comparison. The problems where the number of functions evaluations equal to 1500 are those where it was not able to declare convergence within the limit of function evaluations imposed, which is defined as $300 \times n$, with n being the number of variables in the problem.

We also present *performance profiles* of all the methods. Such profiles were introduced by Dolan and Moré [8] as a manner to compare the performance of a set of solvers $\mathcal{S}$ on a test $\mathcal{P}$. Let n_s denote the number of solvers, and n_p, the number of problems. We are interested in using the number of function evaluations as a performance measure. For each problem p and solver s, we define

$$t_{p,s} = \text{ number of function evaluations required to solve problem } p \text{ by solver } s.$$

We compare the performance on problem p by solver s with the best performance by any solver on this problem; that is, we use the *performance ratio*

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in \mathcal{S}\}}.$$

If we define

$$\rho_s(\tau) = \frac{1}{n_p}\text{size}\{p \in \mathcal{P} : r_{p,s} \leq \tau\},$$

then $\rho_s(\tau)$ is the probability for solver $s \in \mathcal{S}$ that a performance ratio $r_{p,s}$ is within a factor $\tau \in \mathbb{R}$ of the best possible ratio. The function ρ_s is the (cumulative) distribution function for the performance ratio.

The performance profiles for the first type of comparison are shown in Figure 4.1 and in Figure 4.2. In the former, the four different approaches to build the models in DEFT-FUNNEL are compared to each other, and, in the latter, each of these approaches are individually compared to the CDFO and COBYLA methods. The performance profiles for the second type of comparison, in turn, can be found in Figure 4.3, where each approach to build the models in DEFT-FUNNEL is individually compared to the CDFO and COBYLA methods as well. We remind the reader here that COBYLA uses linear models only, which may be a disadvantage and might bias the comparison somewhat against this approach. Globally the results obtained by DEFT-FUNNEL are encouraging.

5 Conclusions

We have presented a derivative-free optimization algorithm for equality-constrained optimization, based on the trust-funnel approach of Gould and Toint [13]. The new algorithm, called DEFT-FUNNEL, was tested in a set of numerical experiments whose results are encouraging. Four described variants of the new method differing by their choice of objective function model surpassed the CDFO and COBYLA methods. Among these variants, the minimum ℓ_2-norm model variant expectedly outperformed the subbasis selection due to it is robustness, but also somewhat surprisingly, the minimum Frobenius norm approach. Clearly, continued experimentation with the DEFT-FUNNEL approach is needed to fully assess its true numerical potential, but the preliminary results give hope that it can be useful in practice.

The intention of the authors is now to extend the method to deal with problems involving inequality constraints as well, with the goal to obtain a derivative-free method for problems with general constraints.

Acknowledgements

The first author gratefully acknowledges a CERUNA-UNamur scholarship.

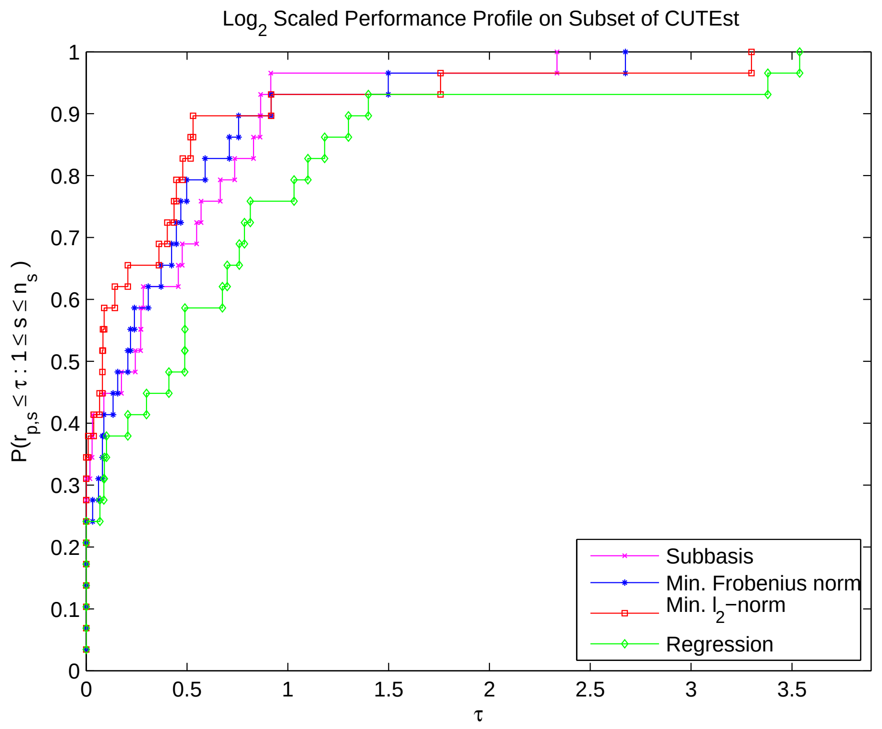


Figure 4.1: *Log*$_2$-scaled performance profile of DEFT-FUNNEL with different approaches to build the models for the first type of comparison on a subset of CUTEst

References

[1] B. Colson. *Trust-Region Algorithms for Derivative-Free Optimization and Nonlinear Bilevel Programming.* PhD thesis, Department of Mathematics, FUNDP - University of Namur, Namur, Belgium, 2004.

[2] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-Region Methods*. MOS-SIAM Series on Optimization, Philadelphia, 2000.

[3] A. R. Conn, K. Scheinberg, and Ph. L. Toint. On the convergence of derivative-free methods for unconstrained optimization. In A. Iserles and M. Buhmann, editors, *Approximation Theory and Optimization: Tributes to M. J. D. Powell*, pages 83–108, Cambridge, England, 1997. Cambridge University Press.

[4] A. R. Conn, K. Scheinberg, and Ph.L. Toint. A derivative-free optimization algorithm in practice. In *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO, 1998.

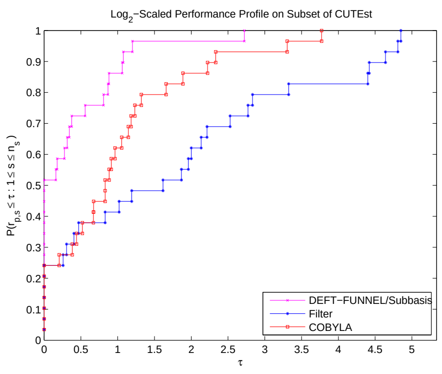
[5] A. R. Conn, K. Scheinberg, and L. N. Vincente. *Introduction to derivative-free optimization*. MPS-SIAM Book Series on Optimization, Philadelphia, 2009.

[6] A. R. Conn, K. Scheinberg, and H. Zang. A derivative-free algorithm for least-squares minimization. *SIAM Journal on Optimization*, 20(6):3555–3576, 2010.

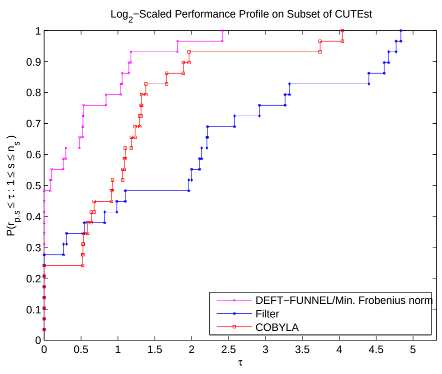
[7] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1983. Reprinted as *Classics in Applied Mathematics 16*, SIAM, Philadelphia, USA, 1996.

[8] E. Dolan and J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–13, 2002.

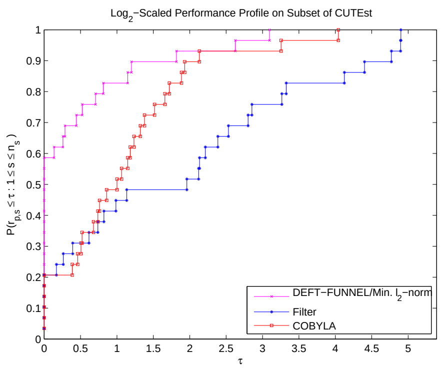
[9] G. Fasano, J. Nocedal, and J.-L. Morales. On the geometry phase in model-based algorithms for derivative-free optimization. *Optimization Methods and Software*, 24(1):145–154, 2009.



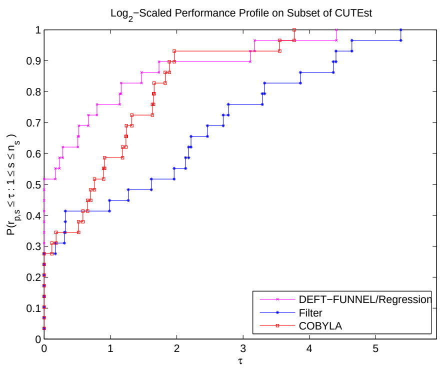
(a) Subbasis



(b) Min. Frobenius norm



(c) Min. ℓ_2-norm



(d) Regression

Figure 4.2: *Log*$_2$-scaled performance profile of the methods DEFT-FUNNEL (with different approaches to build the models), CDFO and COBYLA for the first type of comparison on a subset of CUTEst

[10] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, 91(2):239–269, 2002.

[11] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, London, 1981.

[12] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEr, a constrained and unconstrained testing environment, revisited. *ACM Transactions on Mathematical Software*, 29(4):373–394, 2003.

[13] N. I. M. Gould and Ph. L. Toint. Nonlinear programming without a penalty function or a filter. *Mathematical Programming*, 122(1):155–196, 2010.

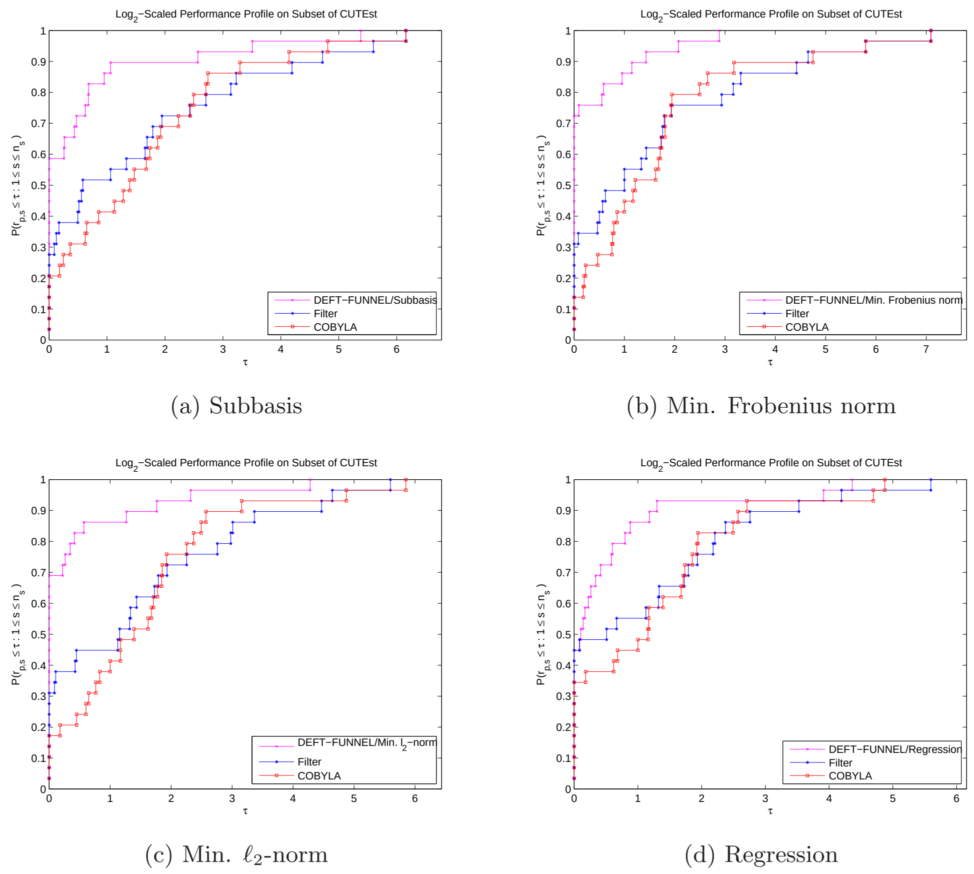
[14] Nicholas I. M. Gould, Mary E. Hribar, and Jorge Nocedal. On the solution of equality constrained quadratic programming problems arising in optimization. *SIAM J. Sci. Comput.*, 23(4):1376–1395, April 2001.

[15] S. Gratton, Ph. L. Toint, and A. Tröltzsch. An active-set trust-region method for bound-constrained nonlinear optimization without derivatives. *Optimization Methods and Software*, 26(4-5):875–896, 2011.

[16] R. M. Lewis and V. Torczon. Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization*, 9:1082–1099, 1999.

[17] R. M. Lewis and V. Torczon. Pattern search algorithms for linearly constrained minimization. *SIAM Journal on Optimization*, 10:917–941, 2000.

[18] R. M. Lewis and V. Torczon. A globally convergent augmented langragian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Optimization*, 12(4):1075–1089, 2002.



(a) Subbasis (b) Min. Frobenius norm

(c) Min. ℓ_2-norm (d) Regression

Figure 4.3: *Log*$_2$-scaled performance profile of the methods DEFT-FUNNEL (with different approaches to build the models), CDFO and COBYLA for the second type of comparison on a subset of CUTEst

[19] R. M. Lewis and V. Torczon. Active set identification for linearly constrained minimization without explicit derivatives. *SIAM Journal on Optimization*, 20(3):1378–1405, 2009.

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

[21] J. J. Moré. The Levenberg-Marquardt algorithm: Implementation and theory. In G. A. Watson, editor, *Numerical Analysis*, Lecture Notes in Mathematics, chapter 10. Springer Berlin / Heidelberg, 1978.

[22] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[23] J. Nocedal and S. J. Wright. *Numerical Optimization*. Series in Operations Research. Springer Verlag, Heidelberg, Berlin, New York, 1999.

[24] E. O. Omojokun. *Trust region algorithms for optimization with nonlinear equality and inequality constraints*. PhD thesis, University of Colorado, Boulder, Colorado, USA, 1989.

[25] M. J. D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in Optimization and Numerical Analysis, Proceedings of the Sixth Workshop on Optimization and Numerical Analysis, Oaxaca, Mexico*, volume 275, pages 51–67, Dordrecht, The Netherlands, 1994. Kluwer Academic Publishers.

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

[27] M. J. D. Powell. UOBYQA: unconstrained optimization by quadratic interpolation. *Mathematical Programming, Series A*, 92:555–582, 2002.

[28] M. J. D. Powell. The newuoa software for unconstrained optimization without derivatives. In G. Pillo, M. Roma, and Panos Pardalos, editors, *Large-Scale Nonlinear Optimization*, volume 83 of *Nonconvex Optimization and Its Applications*, pages 255–297. Springer US, 2006.

[29] M. J. D. Powell. Developments of NEWUOA for minimization without derivatives. *IMA Journal of Numerical Analysis*, 28(4):649–664, 2008.

[30] M. J. D. Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. Technical report, Department of Applied Mathematics and Theoretical Physics, Cambridge University, Cambridge, England, 2009.

[31] K. Scheinberg and Ph. L. Toint. Self-correcting geometry in model-based algorithms for derivative-free unconstrained optimization. *SIAM Journal on Optimization*, 20(6):3512–3532, 2010.

[32] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983.

[33] G. W. Stewart. A modification of davidon's minimization method to accept difference approximations of derivatives. *Journal of the ACM*, 14(1):7283, 1967.

[34] Ph. L. Toint. Towards an efficient sparsity exploiting Newton method for minimization. In I. S. Duff, editor, *Sparse Matrices and Their Uses*, pages 57–88, London, 1981. Academic Press.

[35] A. Tröltzsch. *An active-set trust-region method for bound-constrained nonlinear optimization without derivatives applied to noisy aerodynamic design problems.* PhD thesis, University of Toulouse, France, 2011.

[36] D. Winfield. *Function and functional optimization by interpolation in data tables.* PhD thesis, Harvard University, Cambridge, USA, 1969.

[37] D. Winfield. Function minimization by interpolation in a data table. *Journal of the Institute of Mathematics and its Applications*, 12:339–347, 1973.

A Appendix

Table 1.2: Numerical results of DEFT-FUNNEL/Subbasis, CDFO and COBYLA methods for the first type of comparison.

Problem	n	m	CDFO			DEFT-FUNNEL			COBYLA
			Iter.	Feval.T	Feval.S	Iter.	Feval.T	Feval.S	Feval.T
BT1	2	1	56	105	104	36	46	41	53
BT2	3	1	35	191	183	417	439	430	249
BT3	5	3	4	237	217	32	65	16	135
BT4	3	2	6	42	42	22	34	33	64
BT5	3	2	3	53	13	13	40	31	78
BT6	5	2	89	912	911	96	134	134	91
BT7	5	3	26	676	648	49	99	79	176
BT8	5	2	24	499	498	27	70	50	95
BT9	4	2	136	1500	1500	22	52	38	115
BT10	2	2	9	101	96	12	48	7	23
BT11	5	3	34	466	445	133	184	164	152
BT12	5	3	29	1098	1077	20	44	24	601
HS6	2	1	6	16	12	14	28	23	40
HS7	2	1	236	685	685	21	32	27	51
HS8	2	2	2	68	58	9	19	14	17
HS9	2	1	10	23	18	37	52	47	33
HS26	3	1	1500	1500	1500	54	90	81	71
HS27	3	1	133	548	540	69	95	86	939
HS28	3	1	4	19	10	13	35	16	60
HS39	4	2	9	179	151	22	52	38	46
HS40	4	3	66	1374	1359	19	49	35	78
HS42	4	2	3	54	54	12	39	26	72
HS46	5	2	57	302	282	76	149	129	751
HS48	5	2	9	37	16	21	46	26	87
HS49	5	2	165	295	285	178	235	215	1096
HS50	5	3	10	40	19	219	264	244	148
HS51	5	3	8	37	17	23	78	16	84
HS61	3	2	1500	1500	1500	791	844	844	1500
HS100LNP	7	2	92	905	890	146	253	218	195

Table 1.3: Numerical results of DEFT-FUNNEL/Frobenius norm, CDFO and COBYLA methods for the first type of comparison.

Problem	n	m	CDFO			DEFT-FUNNEL			COBYLA
			Iter.	Feval.T	Feval.S	Iter.	Feval.T	Feval.S	Feval.T
BT1	2	1	56	105	104	55	65	55	53
BT2	3	1	35	191	183	97	131	122	249
BT3	5	3	4	237	217	28	55	17	135
BT4	3	2	6	42	42	15	34	25	64
BT5	3	2	3	53	44	11	30	21	78
BT6	5	2	89	912	911	77	131	111	91
BT7	5	3	26	676	648	67	113	93	176
BT8	5	2	24	499	498	20	66	46	95
BT9	4	2	136	1500	1500	25	55	41	115
BT10	2	2	9	101	96	15	51	10	23
BT11	5	3	34	466	445	50	101	81	152
BT12	5	3	29	1098	1077	21	45	45	601
HS6	2	1	6	16	12	13	23	18	40
HS7	2	1	236	685	685	14	24	19	51
HS8	2	2	2	68	58	8	18	13	17
HS9	2	1	10	23	18	37	52	47	33
HS26	3	1	1500	1500	1500	45	76	67	71
HS27	3	1	133	548	540	28	57	49	939
HS28	3	1	4	19	10	14	34	15	60
HS39	4	2	9	179	151	25	55	41	46
HS40	4	3	66	1374	1359	14	54	40	78
HS42	4	2	3	54	54	15	45	31	72
HS46	5	2	57	302	282	324	421	401	751
HS48	5	2	9	37	16	29	76	25	87
HS49	5	2	150	281	262	1500	1500	1500	1096
HS50	5	3	10	40	19	92	140	120	148
HS51	5	3	8	37	17	23	77	12	84
HS61	3	2	1500	1500	1500	604	700	700	1500
HS100LNP	7	2	92	905	890	193	282	247	195

Table 1.4: Numerical results of DEFT-FUNNEL/ℓ_2-norm, CDFO and COBYLA methods for the first type of comparison.

Problem	n	m	CDFO			DEFT-FUNNEL			COBYLA
			Iter.	Feval.T	Feval.S	Iter.	Feval.T	Feval.S	Feval.T
BT1	2	1	56	105	104	438	453	448	53
BT2	3	1	35	191	183	62	87	78	249
BT3	5	3	4	237	217	23	41	23	135
BT4	3	2	6	42	42	13	32	23	64
BT5	3	2	3	53	44	11	30	21	78
BT6	5	2	89	912	911	57	111	91	91
BT7	5	3	26	676	648	54	97	77	176
BT8	5	2	24	499	498	25	69	49	95
BT9	4	2	136	1500	1500	25	55	41	115
BT10	2	2	9	101	96	15	51	10	23
BT11	5	3	34	466	445	57	107	87	152
BT12	5	3	29	1098	1077	20	63	43	601
HS6	2	1	6	16	12	13	23	18	40
HS7	2	1	236	685	685	13	23	18	51
HS8	2	2	2	68	58	7	13	8	17
HS9	2	1	10	23	18	113	142	139	33
HS26	3	1	1500	1500	1500	48	78	69	71
HS27	3	1	133	548	540	28	57	48	939
HS28	3	1	4	19	10	11	31	12	60
HS39	4	2	9	179	151	25	55	41	46
HS40	4	3	66	1374	1359	15	46	32	78
HS42	4	2	3	54	54	15	45	31	72
HS46	5	2	57	302	282	125	197	177	751
HS48	5	2	9	37	16	19	65	45	87
HS49	5	2	150	281	262	175	250	230	1096
HS50	5	3	10	40	19	95	141	121	148
HS51	5	3	8	37	17	30	85	65	84
HS61	3	2	1500	1500	1500	731	899	899	1500
HS100LNP	7	2	92	905	890	165	265	230	195

Table 1.5: Numerical results of DEFT-FUNNEL/Regression, CDFO and COBYLA methods for the first type of comparison.

Problem	n	m	CDFO			DEFT-FUNNEL			COBYLA
			Iter.	Feval.T	Feval.S	Iter.	Feval.T	Feval.S	Feval.T
BT1	2	1	56	105	104	452	479	468	53
BT2	3	1	35	191	183	117	153	153	249
BT3	5	3	4	237	217	23	43	43	135
BT4	3	2	6	42	42	12	34	23	64
BT5	3	2	3	53	44	10	22	22	78
BT6	5	2	89	912	911	95	252	212	91
BT7	5	3	26	676	648	60	104	104	176
BT8	5	2	24	499	498	22	51	26	95
BT9	4	2	136	1500	1500	24	73	48	115
BT10	2	2	9	101	96	10	27	18	23
BT11	5	3	34	466	445	65	171	171	152
BT12	5	3	29	1098	1077	19	44	44	601
HS6	2	1	6	16	12	24	24	24	40
HS7	2	1	236	685	685	25	47	36	51
HS8	2	2	2	68	58	7	15	8	17
HS9	2	1	10	23	18	380	488	490	33
HS26	3	1	1500	1500	1500	44	101	82	71
HS27	3	1	133	548	540	38	80	80	939
HS28	3	1	4	19	10	11	33	12	60
HS39	4	2	9	179	151	24	73	48	46
HS40	4	3	66	1374	1359	14	33	33	78
HS42	4	2	3	54	54	15	48	32	72
HS46	5	2	57	302	282	87	242	201	751
HS48	5	2	9	37	16	18	45	45	87
HS49	5	2	150	281	262	493	620	620	1096
HS50	5	3	10	40	19	299	345	345	148
HS51	5	3	8	37	17	49	123	123	84
HS61	3	2	1500	1500	1500	865	1500	1500	1500
HS100LNP	7	2	92	905	890	212	436	436	195

Table 1.6: Number of function evaluations required by DEFT-FUNNEL, CDFO and COBYLA methods to converge in the second type of comparison.

Problem	n	m	Subbasis	Min. Frobenius norm	Min. ℓ_2-norm	Regression	CDFO	COBYLA
BT1	2	1	408	62	447	473	58	23
BT2	3	1	409	101	64	120	69	230
BT3	5	3	16	17	21	21	18	107
BT4	3	2	13	11	11	11	42	36
BT5	3	2	21	11	1500	1500	1500	1500
BT6	5	2	98	70	60	123	1500	82
BT7	5	3	1500	1500	1500	1500	1500	1500
BT8	5	2	50	25	47	26	37	58
BT9	4	2	21	22	25	48	553	77
BT10	2	2	7	10	10	18	27	17
BT11	5	3	125	39	35	86	78	120
BT12	5	3	23	24	22	22	48	645
HS6	2	1	11	12	12	12	12	24
HS7	2	1	27	19	18	36	145	32
HS8	2	2	14	13	8	8	20	18
HS9	2	1	29	29	51	34	15	17
HS26	3	1	43	27	46	57	1500	31
HS27	3	1	85	27	26	58	268	1500
HS28	3	1	16	15	12	12	10	38
HS39	4	2	21	22	25	48	197	38
HS40	4	3	100	26	13	13	88	47
HS42	4	2	1500	16	16	16	53	36
HS46	5	2	57	82	89	91	128	37
HS48	5	2	18	16	20	20	15	48
HS49	5	2	144	1500	151	308	203	780
HS50	5	3	216	80	95	286	19	107
HS51	5	3	16	12	16	16	17	42
HS61	3	2	230	1500	314	1500	1500	1500
HS100LNP	7	2	153	166	168	246	229	1500