FUNDP
NAMUR

Solving structured nonlinear least-squares and nonlinear feasibility problems with expensive functions

by M. Kaiser, K. Klamroth, A. Thekale, Ph. L. Toint

Report NAXYS-07-2010 9 December 2010

naXys
Namur Center for Complex SYStems

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

http://www.fundp.ac.be/sciences/naxys

Solving structured nonlinear least-squares and nonlinear feasibility problems with expensive functions

Markus Kaiser[*], Kathrin Klamroth[†], Alexander Thekale[‡], Philippe L. Toint[§]

9 December 2010

Abstract

We present an algorithm for nonlinear least-squares and nonlinear feasibility problems, i.e. for systems of nonlinear equations and nonlinear inequalities, which depend on the outcome of expensive functions for which derivatives are assumed to be unavailable. Our algorithm combines derivative-free techniques with filter trust-region methods to keep the number of expensive function evaluations low and to obtain a robust method. Under adequate assumptions, we show global convergence to a feasible point. Numerical results indicate a significant reduction in function evaluations compared to other derivative based and derivative-free solvers for nonlinear feasibility problems.

Keywords: feasibility problem, nonlinear systems, nonlinear least-squares, structured problems, derivative-free, multidimensional filter, trust-region, global convergence.

AMS Classification: 90C30, 65K05, 90C56, 90C26

1 Introduction

In this paper we consider the solution of the general feasibility problem with expensive functions, where one seeks a vector $x \in \mathbb{R}^n$ such that the *Expensive System* of nonlinear equations and inequalities

$$\begin{array}{rcl} c_{\mathcal{E}}(x, u(x)) & = & 0 \\ c_{\mathcal{I}}(x, u(x)) & \leq & 0 \end{array} \tag{ES}$$

holds, where $u : \mathbb{R}^n \to \mathbb{R}^m$ is a sufficiently smooth expensive function and $c_{\mathcal{E}} : \mathbb{R}^{n+m} \to \mathbb{R}^p$ and $c_{\mathcal{I}} : \mathbb{R}^{n+m} \to \mathbb{R}^q$ are sufficiently smooth cheap functions. In our context, we call a function *expensive*, if its evaluation is costly in some sense and plays a major part in the computational cost of the solution of the system (ES). A function is called *cheap* if the cost of its evaluation is negligible. In order to find a solution of (ES), one often uses its well-known reformulation as a nonlinear least-squares problem, in which one searches a (hopefully global) minimizer $x \in \mathbb{R}^n$ of the nonlinear unconstrained problem (see, e.g., [17])

$$\min_{x \in \mathbb{R}^n} f(x, u(x)) = \frac{1}{2} \|\vartheta(x, u(x))\|_2^2 \tag{1.1}$$

where we define

$$\vartheta(x, u(x)) \stackrel{\text{def}}{=} \begin{pmatrix} c_{\mathcal{E}}(x, u(x)) \\ [c_{\mathcal{I}}(x, u(x))]_+ \end{pmatrix} \in \mathbb{R}^{p+q} \tag{1.2}$$

[*]Department of Mathematics and Natural Science, University of Wuppertal, Wuppertal, Germany
[†]Department of Mathematics and Natural Science, University of Wuppertal, Wuppertal, Germany
[‡]Department of Mathematics, University of Erlangen-Nuremberg, Erlangen, Germany
[§]Namur Research Center for Complex Systems (NAXYS), FUNDP-University of Namur, 61, rue de Bruxelles, B-5000 Namur, Belgium

with $[c_{\mathcal{I}}(x, u(x))]_+ \stackrel{\rm def}{=} \max[0, c_{\mathcal{I}}(x, u(x))]$ is the violation of the equations and inequalities, respectively. Importantly, our developments also apply directly to the nonlinear least-squares problem (1.1).

Feasibility problems of the form (ES) or nonlinear least-squares of the form (1.1) occur in many different applications, e.g. as discretized nonlinear partial differential equations [32] or in the restoration phase of filter methods for nonlinear optimization [10]. A large number of algorithms have been developed to solve this general class of problem or related variants, such as nonlinear systems of equations and nonlinear least-squares problems. These algorithms are based on various techniques, including Newton methods [25, 30], trust-region methods [7, 12, 17, 22] or evolutionary algorithms [19]. For a survey of algorithms for nonlinear systems of equations, see also [23]. In many engineering contexts, these feasibility or least-squares systems often involve the outcome of an expensive function such as a numerical simulation. The evaluation of these expensive functions is typically very time consuming and derivative information unavailable, which suggests using derivative-free methods (for instance, see [29] for a pattern search method for solving nonlinear equation systems).

The purpose of this paper is to present a globally convergent method for solving the nonlinear feasibility problems (ES). The proposed method is a filter-trust-region algorithm akin to FILTRANE [17]. It can also be considered as hybrid in the sense that it handles expensive functions using derivative-free techniques (see [4, 6]) while more standard technology using derivatives is applied to the cheap functions.

The paper is organized as follows. In Section 2, we describe the general framework of our filter-trust-region algorithm. Under appropriate assumptions, we show its convergence to a local first-order critical point in Section 3. A very important step within our method is the determination of a trial point, which we describe in Section 4. In Section 5, we report promising numerical results and compare our method to other derivative-based and derivative-free optimization methods. Finally, concluding remarks are given in Section 6. In what follows, we use the symbols $\|\cdot\|_2$ and $\|\cdot\|_\infty$ to denote the standard Euclidean and infinity norms.

2 An algorithm for nonlinear feasibility problems with expensive functions

2.1 Using cheap systems and valid models

The main idea of our algorithm is to solve the expensive system (ES) by iteratively solving a sequence of *Cheap Systems* of nonlinear equations and inequalities in a trust-region framework where, at iteration k, the expensive function u is replaced by a cheap model function $m_k^u : \mathbb{R}^n \to \mathbb{R}^m$. Our analysis make much use of the notion that, the cheap model m_k^u may be considered *valid* in a neighbourhood defined for an iterate x_k of the iterative process and some $\delta_k > 0$ by

$$\mathcal{Q}(x_k, \delta_k) \stackrel{\rm def}{=} \{x \in \mathbb{R}^n \mid \|x - x_k\|_\infty \leq \delta_k\}, \tag{2.3}$$

This validity concept, introduced in [4], Section 9.1, is defined as follows[(1)].

Assumption 1 *(Validity of m_k^u)*
The model m_k^u coincides with u at x_k, i.e.,

$$m_k^u(x_k) = u(x_k) \tag{2.4}$$

and the following error bounds hold for some constants $\kappa_{\rm ub} > 0$, $\kappa_{\rm gu} > 0$ and all $k \geq 0$,

$$\|u(x) - m_k^u(x)\|_2 \leq \kappa_{\rm ub}\delta_k^2 \tag{2.5}$$

$$\|\nabla u(x) - \nabla m_k^u(x)\|_2 \leq \kappa_{\rm gu}\delta_k \tag{2.6}$$

for all $x \in \mathcal{Q}(x_k, \delta_k)$.

[(1)] We mention that the assumption $m_k^u(x_k) = u(x_k)$ could be omitted in this definition, but we use it nevertheless because it significantly facilitates the subsequent analysis. Similarly, (2.6) could alternatively be derived from (2.5), see [4] (page 309) for more details.

Valid models are interesting because it is known that they provide a degree of approximation of the modelled function which is adequate for their efficient use in optimizaton algorithms such as trust-region methods (see [4] again). Importantly, valid models in the neighbourhood (2.3) may be computed relatively easily in practice, and we refer the reader to [5, 6] for an explicit derivation of such models. The details of this derivation are not important in our context, and it is sufficient for our purpose to assume that validity can be checked and guaranteed in a finite number of steps, whenever needed (see, e.g. [1, 4]). We formulate this double requirement as follows.

Assumption 2 *(Checking validity)*
The validity of m_k^u in $\mathcal{Q}(x_k, \delta_k)$ can be checked in finite time for each iterate x_k and for any value of $\delta_k > 0$.

Assumption 3 *(Guaranteeing validity)*
The model m_k^u can be made valid in $\mathcal{Q}(x_k, \delta_k)$ in a finite number of model improvement steps for any x_k and any $\delta_k > 0$.

Using a valid model m_k^u for u, we may then propose to construct a cheap approximation of problem (ES) of the form

$$\begin{array}{rcl} c_{\mathcal{E}}(x, m_k^u(x)) & = & 0 \\ c_{\mathcal{I}}(x, m_k^u(x)) & \leq & 0 \\ x & \in & \mathcal{Q}(x_k, \delta_k). \end{array} \qquad (\mathrm{CS}_k)$$

The core of our proposal is then to solve the original problem (ES) using its cheap approximation (CS_k) as a local model in a method strongly inspired by the FILTRANE filter-trust-region method and the conditional trust-region algorithm described in Section 9.1 of [4].

2.2 Computing a step and accepting it

FILTRANE and its derivative-free variant described here are iterative algorithms and use the filter and trust-region methodologies (we refer the reader to [4] for an extensive coverage of the subject). They consider the least-squares formulation of the problem at hand, which is given here, for problem (CS_k), by

$$\begin{array}{lrcl} \min\limits_{x \in \mathbb{R}^n} & f(x, m_k^u(x)) & = & \frac{1}{2}\|\vartheta(x, m_k^u(x))\|_2^2 \\ s.t. & x & \in & \mathcal{Q}(x_k, \delta_k). \end{array} \qquad (2.7)$$

The idea is thus to consider $f(x, m_k^u(x))$ as a model of $f(x, u(x))$. At each iteration k, a trial point x_k^+ is computed by minimizing this model either in a relatively controlled trust region, giving

$$x_k^+ \in \mathcal{Q}(x_k, \delta_k) \qquad (2.8)$$

(where the control is exercised by suitably updating δ_k), or, when the algorithm seems to be progressing well, in a (potentially much) larger region of the same form, giving

$$x_k^+ \in \mathcal{Q}(x_k, \kappa_\delta \delta_k) \qquad (2.9)$$

for some large constant $\kappa_\delta \geq 1$. The label *RESTRICT* indicates which region is requested. If *RESTRICT* is set, x_k^+ has to satisfy (2.8), otherwise x_k^+ has to satisfy (2.9). In both cases, the trial point is computed to ensure a variant of the standard "Cauchy decrease" given by the inequality

$$f(x_k, m_k^u(x_k)) - f(x_k^+, m_k^u(x_k^+)) \geq \kappa_{\mathrm{mdcd}} \|g_k\|_2 \min\left[\frac{\|g_k\|_2}{\beta_k}, \delta_k\right] \qquad (2.10)$$

where $g_k \stackrel{\mathrm{def}}{=} \nabla f(x_k, m_k^u(x_k))$, β_k is an upper bound on the norm of the Hessian of $f(x, m_k^u(x))$ in $\mathcal{Q}(x_k, \delta_k)$ and κ_{mdcd} is a constant in $(0, 1)$. Note that the right-hand side of (2.10) depends on the norm of the gradient of the nonlinear cheap function $f(x, m_k^u(x))$ and not on the gradient of the original expensive function $f(x, u(x))$. How a step can be computed in practice to ensure (2.10) is the subject of Section 4.

Once a trial point x_k^+ satisfying (2.8) or (2.9) and (2.10) is computed, the algorithm uses a *multi-dimensional filter* technique to decide upon its acceptability as the next iterate. This concept is a variant

of the filter method introduced in [11], adapted for feasibility problems. We now outline the idea and refer to [12] for more details. A filter is based on the idea of dominance, which is borrowed (and modified) from multicriteria optimization. In our context, we say that an iterate x_{k_1} *dominates* an iterate x_{k_2} whenever

$$|\vartheta_i(x_{k_1}, u(x_{k_1}))| \leq |\vartheta_i(x_{k_2}, u(x_{k_2}))| \qquad \text{for all} \qquad i \in \{1, \ldots, p+q\}.$$

Based on this idea, we define the set F_u as a list of vectors of dimension $(p+q)$ of the form $|\vartheta(x_l, u(x_l))| \stackrel{\rm def}{=} (|\vartheta_1(x_l, u(x_l))|, \ldots, |\vartheta_{p+q}(x_l, u(x_l))|)$ where $l \in \{1, \ldots, k\}$ such that no entry is dominated by another entry, in the sense that

$$|\vartheta_i(x_{k_1}, u(x_{k_1}))| < |\vartheta_i(x_{k_2}, u(x_{k_2}))| \qquad \text{for at least one} \qquad i \in \{1, \ldots, p+q\}$$

for all $|\vartheta(x_{k_1}, u(x_{k_2}))|, |\vartheta(x_{k_2}, u(x_{k_2}))| \in F_u$ and $k_1 \neq k_2$. The set F_u is called *multidimensional filter*. Filter algorihms maintain such a filter F_u by adding and removing entries, and use it in the decision of whether or not the trial point x_k^+ can be accepted as the next iterate x_{k+1}. We say that x_k^+ is acceptable for the filter F_u when its associate entry $\left|\vartheta_i(x_k^+, u(x_k^+))\right|$ is not "dominated", that is if and only if, for all $|\vartheta(x_l, u(x_l))| \in F_u$, one has that

$$\exists i \in \{1, \ldots, p+q\} \qquad \left|\vartheta_i(x_k^+, u(x_k^+))\right| \leq |\vartheta_i(x_l, u(x_l))| - \gamma_\theta \min\left[\left|\vartheta(x_k^+, u(x_k^+))\right|, |\vartheta(x_l, u(x_l))|\right]$$

where $\gamma_\theta \in (0, 1/\sqrt{p+q})$ is a constant. The rule to accept the trial as the next iterate is then to accept it either if it is acceptable for the filter, or if it produces an achieved reduction in $f(x, u(x))$ which is at least a fraction of that predicted using the model $f(x, m_k^u(x))$, as in standard trust-region methods. This decision is also combined with that of allowing larger steps (i.e. (2.9)) or using the more conservative strategy (2.8), as is described in Algorithm 2.1 below.

2.3 The outer algorithm

We now can state our algorithm for solving (1.1).

Algorithm 2.1 ***Filter trust-region algorithm for feasibility problems with expensive functions***

Step 0: Initialization. An initial point $\bar{x}$, an initial trust-region radius $\delta_0 = \delta_{\rm ref} > 0$ and $k_0 \geq 0$ as well as the constants mentioned below are given. Build an initial model m_0^u of u that is valid in $\mathcal{Q}(\bar{x}, \delta_0)$, define $\mathcal{X}_0$ as set of corresponding interpolation points and determine x_0 such that

$$f(x_0, u(x_0)) = \min_{x \in \mathcal{X}_0} f(x, u(x)).$$

Compute $c_0 = c(x_0, u(x_0))$, ϑ_0 and f_0. Set $k = 0$, unset $RESTRICT$ and initialize the filter $F_u = \emptyset$.

Step 1: Stopping test. If $\vartheta(x_k, u(x_k)) = 0$, or if $\|g_k\|_2 < \varepsilon_{\rm end}$ for a valid model m_k^u in $\mathcal{Q}_k(x_k, \delta_{\rm end})$ for some $\delta_{\rm end} \in (0, \mu\|g_k\|_2]$, STOP.
If $\|g_k\|_2 < \varepsilon_{\rm end}$ and m_k^u is not valid in $\mathcal{Q}_k(x_k, \delta_{\rm end})$ for some $\delta_{\rm end} \in (0, \mu\|g_k\|_2]$, perform as many improvement steps as necessary until the model is valid in $\mathcal{Q}_k(x_k, \mu\|g_k\|_2)$ and return to the beginning of Step 1.

Step 2: Trial point determination. Attempt to compute a trial point x_k^+ that satisfies (2.10) and, if $RESTRICT$ is set, also satisfies (2.8), or, if $RESTRICT$ is unset and $k \geq k_0$, also satisfies (2.9). If this is impossible, set $RESTRICT$, $\delta_{k+1} = \gamma_0 \delta_k$, $\rho_k = -\infty$, $\mathcal{X}_k = \{x_k\}$ and go to Step 4.

Step 3: Evaluation of the residual at the trial point. Compute $u(x_k^+)$ and $f(x_k^+, u(x_k^+))$. Set

$$\rho_k = \frac{f(x_k, u(x_k)) - f(x_k^+, u(x_k^+))}{f(x_k, m_k^u(x_k)) - f(x_k^+, m_k^u(x_k^+))}. \tag{2.11}$$

If $\rho_k \geq \eta_1$ set $\mathcal{X}_k = \{x_k^+\}$, otherwise set $\mathcal{X}_k = \{x_k\}$.

Step 4: Model improvement. If $\rho_k < \eta_2$ or m_k^u is invalid in $\mathcal{Q}(x_k, \delta_k)$, perform model improvement steps until the model is valid, possibly enlarging $\mathcal{X}_k$ by adding the newly evaluated points from the improvement steps, and define m_{k+1}^u as the improved model.

Step 5: Trial point determination. Determine $\hat{x}_k \in \mathcal{X}_k$ such that

$$f(\hat{x}_k, u(\hat{x}_k)) = \min_{x \in \mathcal{X}_k} f(x, u(x)), \tag{2.12}$$

set $\hat{\vartheta}_k \stackrel{\text{def}}{=} \vartheta(\hat{x}_k)$ and define, if $\rho_k > -\infty$,

$$\hat{\rho}_k = \frac{f(x_k, u(x_k)) - f(\hat{x}_k, u(\hat{x}_k))}{f(x_k, m_k^u(x_k)) - f(x_k^+, m_k^u(x_k^+))}, \tag{2.13}$$

else define $\hat{\rho}_k = -\infty$.

Step 6: Acceptance test.

- If (2.8) holds for $\hat{x}_k$ and $\hat{\rho}_k \geq \eta_1$, set $x_{k+1} = \hat{x}_k$ and unset $RESTRICT$.
- Else if $\hat{x}_k$ is acceptable for the current filter F_u, set $x_{k+1} = \hat{x}_k$, unset $RESTRICT$ and add $\hat{\vartheta}_k$ to F_u.
- Else, set $x_{k+1} = x_k$ and set $RESTRICT$.

Step 7: Trust-region radius update.

- If (2.8) holds: If $\hat{\rho}_k \geq \eta_2$ holds or m_k^u is valid in $\mathcal{Q}(x_k, \delta_k)$, set $\delta_{\text{ref}} = \delta_k$. Update the trust-region radius by choosing
$$\delta_{k+1} \in \begin{cases} [\gamma_0 \delta_{\text{ref}}, \gamma_1 \delta_{\text{ref}}] & \text{if} \quad \hat{\rho}_k < \eta_1 \\ [\gamma_1 \delta_{\text{ref}}, \delta_{\text{ref}}] & \text{if} \quad \hat{\rho}_k \in [\eta_1, \eta_2) \\ [\delta_{\text{ref}}, \gamma_2 \delta_{\text{ref}}] & \text{if} \quad \hat{\rho}_k \geq \eta_2. \end{cases}$$
- If (2.8) does not hold: Set $\delta_{k+1} = \delta_k$.

Increment k by one and go to Step 1.

This algorithm uses the constants $0 < \gamma_0 \leq \gamma_1 < 1 \leq \gamma_2$, $\gamma_\vartheta \in (0, 1/\sqrt{p+q})$, $0 < \eta_1 < \eta_2 < 1$, $\mu > 0$, $\kappa_\delta \geq 1$ and $\varepsilon_{\text{end}} > 0$. A reasonable choice for these parameters appears to be $\gamma_0 = 0.1$, $\gamma_1 = 0.25$, $\gamma_2 = 7.5$, $\gamma_\vartheta = 10^{-4}$, $\eta_1 = 0.2$, $\eta_2 = 0.9$, $\mu = 0.5$, $\kappa_\delta = 10^3$ and $\varepsilon_{\text{end}} = 10^{-6}$. Some further comments on Algorithm 2.1 are also useful.

1. The title of this paragraph is motivated by the fact that Algorithm 2.1 is intended to solve (ES), but it still relies on an inner algorithm for finding a trial point in Step 2. The complete method can therefore be viewed as consisting of an outer and an inner algorithm, the latter being described in Section 4.

2. The criticality test in Step 1 causes practical problems if $\|g_k\|_2 = 0$ since in this situation $\mathcal{Q}(x_k, \delta_{\text{end}})$ degenerates to $\{x_k\}$ and it is impossible to build a valid model. Therefore, the model should be chosen in a very small region around x_k if this situation occurs. For the convergence theory it is assumed that $\mathcal{Q}(x_k, \delta_{\text{end}})$ can be arbitrarily small.

3. In Step 2, we attempt to compute a trial point x_k^+ satisfying (2.10) with some appropriate subroutine. An example of such a method satisfying the assumptions needed for our convergence analysis (see Section 3) is presented in Section 4. If this method does not succeed to determine a trial point satisfying (2.10) in iteration k (this may happen, for example, if m_k^u represents u very badly), we call iteration k a *failed iteration*. The set of all failed iterations is denoted by $\mathcal{F}$.

4. The statement of the acceptance test in Step 6 is different from that used in [17], but is equivalent and maybe more intuitive. See also Figure 2.1 on the following page.

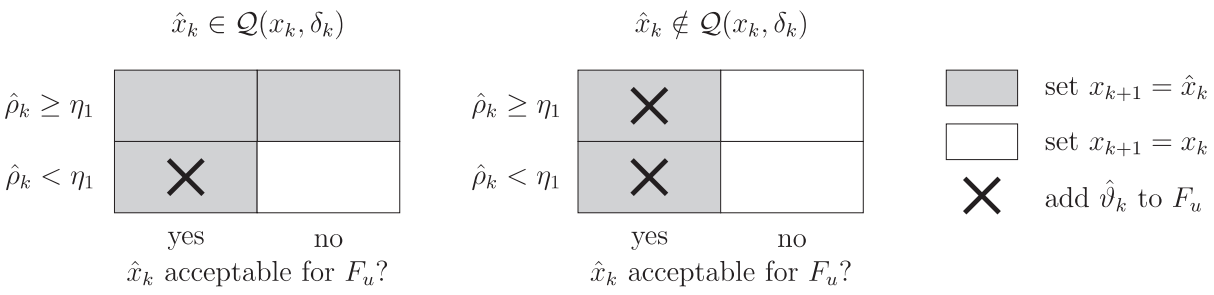

Figure 2.1: Illustration of the decision scheme (Step 6) in Algorithm 2.1

5. An important step in the convergence analysis for trust-region methods is to guarantee that the trust-region cannot become arbitrarily small whenever the current iterate is not first-order critical. Using δ_{ref} in the trust-region update (Step 7) instead of δ_k ensures this desirable property even in the unfavourable situation where arbitrarily many trial points x_k^+ are accepted because of the filter in a situation when current model is poor, i.e. $\rho_k < \eta_1$.

6. We found best in practice to allow large steps in the early iterations if *RESTRICT* is unset using $x^+ \in \mathbb{R}^n$ instead of (2.9), which motivates the restriction imposed in the algorithm (Step 2) that k has to exceed k_0 (typically $k_0 = 10n$) for this option to be considered. But note that it is important to ensure a bound on the step length $x_k^+ - x_k$ for all iterations $k \geq k_0$ for some $k_0 \in \mathbb{N}$ to maintain the convergence of Algorithm 2.1, as we show below.

3 Convergence analysis

We now investigate the convergence properties of Algorithm 2.1. In addition to the assumptions we made on the model m_k^u of the expensive function (Assumption 1, 2 and 3), we assume the following.

Assumption 4 *(Functions' smoothness)*
The functions $c_i : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, $i \in \mathcal{E} \cup I$, are twice continuously differentiable on $\mathbb{R}^n \times \mathbb{R}^m$. The expensive function $u : \mathbb{R}^n \to \mathbb{R}^m$ is twice continuously differentiable on $\mathbb{R}^n$.

Assumption 5 *(Boundedness)*
All points that are evaluated in f or m_k^u, for all k, remain in a bounded convex domain $\Omega \subset \mathbb{R}^n$.

Assumption 6 *(Model's smoothness)*
The model $m_k^u : \mathbb{R}^n \to \mathbb{R}^m$ is twice continuously differentiable in x for all k.

Assumption 7 *(Method determining the trial point)*
For all k, if the trial point x_k^+ can be determined in Step 2 of Algorithm 2.1, it satisfies (2.10)*. Moreover, for all $\varepsilon > 0$ there exists an $\varepsilon_{\text{mdc}} > 0$ such that, if RESTRICT is set, $\|g_k\|_2 > \varepsilon$ and $\delta_k \leq \varepsilon_{\text{mdc}}$, then x_k^+ can be determined, i.e. $k \notin \mathcal{F}$.*

Note that Assumptions 2 and 3 imply that the validity test in Steps 1 and 4 of Algorithm 2.1 is always possible and that, whenever a model improvement is necessary to ensure its validity, no infinite loop is produced by the algorithm. Assumptions 4 and 5 together imply that there exists a bounded convex domain $\Omega^* \subseteq \mathbb{R}^m$ such that

$$u(\Omega) \subseteq \Omega^*.$$

Assumption 7 is not common for trust-region methods and it is not obvious that it can be satisfied. We show in Section 4 how to ensure it. Also note that whenever m_k^u is valid, (2.4) in Assumption 1 implies that

$$f(x, m_k^u(x_k)) = \frac{1}{2}\|\vartheta(x_k, m_k^u(x_k))\|_2^2 = \frac{1}{2}\|\vartheta(x_k, u(x_k))\|_2^2 = f(x, u(x_k)).$$

Assumptions 4 and 5 together imply that there exists a constant $\kappa_{\rm lbf}$ such that, for all $x \in \mathbb{R}^n$,

$$f(x, u(x)) \geq \kappa_{\rm lbf}$$

and that there exists a constant $\kappa_{\rm ubc} > 0$ such that

$$|u(x)| \leq \kappa_{\rm ubc}, \quad \|\nabla u(x)\|_2 \leq \kappa_{\rm ubc} \quad \text{and} \quad \|\nabla^2 u(x)\|_2 \leq \kappa_{\rm ubc} \tag{3.1}$$

for all $x \in \Omega$ and

$$|c_i(x, u)| \leq \kappa_{\rm ubc}, \quad \|\nabla c_i(x, u)\|_2 \leq \kappa_{\rm ubc} \quad \text{and} \quad \|\nabla^2 c_i(x, u)\|_2 \leq \kappa_{\rm ubc} \tag{3.2}$$

for all $x \in \Omega$, $u \in \Omega^*$ and all $i \in \mathcal{E} \cup \mathcal{I}$.

The first step of our convergence analysis considers the case where infinitely many entries $\vartheta_k \stackrel{\rm def}{=} |\vartheta(x_k, u(x_k))|$ are added to the filter F_u. In this case, we may directly apply Theorem 3.1 in [12] (with adapted notation), because it is independent of the definition of ϑ.

Lemma 3.1 (Theorem 3.1 in [12]) *Suppose that Assumptions 4, 5 and 6 hold and that infinitely many values of ϑ_k are added to the filter by Algorithm 2.1. Then*

$$\lim_{k \to \infty,\, i \in \mathcal{E} \cup \mathcal{I}} \|\vartheta_i(x_k, u(x_k))\|_2 = \lim_{k \to \infty} \|\nabla f(x_k, u(x_k))\|_2 = 0.$$

Thus, if infinitely many entries are added to the filter, the sequence of iterates generated by Algorithm 2.1 converges to a feasible point. Algorithm 2.1 therefore terminates after finitely many iterations in the stopping test of Step 1 if $\vartheta_k = 0$ is replaced by $\|\vartheta_k\|_2 \leq \varepsilon$ for some small threshold $\varepsilon > 0$.

After having considered the case where infinitely many points are added to the filter, we now investigate the case where only finitely many points are added to the filter. Convergence to a first-order critical point therefore has to rely on the trust-region mechanism for this case. We may restrict our attention to the iterations that occur after the last iteration where the corresponding iterate is added to the filter, and the mechanism of the algorithm ensures that we may view them as standard trust-region iterations, except for the update of the model and the trust-region radius. In particular, this guarantees that (2.8) holds whenever a trial point is accepted as next current iterate. We verify below that the model is valid whenever necessary for the convergence analysis.

Our analysis now proceeds analogously to the convergence theory of the derivative-free trust-region method presented in Chapter 9.2 in [4], with suitable modifications in the proofs needed to accommodate the composite structure of (1.1). First we investigate the error between the expensive objective function in (1.1) and the objective in the cheap model problem (2.7) as well as that between the corresponding gradients at a given point $x \in \mathcal{Q}(x_k, \delta_k)$.

Lemma 3.2 *Suppose that Assumptions 1, 4-6 hold and that m_k^u is valid in $\mathcal{Q}(x_k, \delta_k)$. Then*

$$|f(x, u(x)) - f(x, m_k^u(x))| \leq \kappa_{\rm ubh} \delta_k^2 \tag{3.3}$$

$$\|\nabla f(x, u(x)) - \nabla f(x, m_k^u(x))\|_2 \leq \kappa_{\rm ubh} \max[\delta_k^2, \delta_k^4] \tag{3.4}$$

for all $x \in \mathcal{Q}(x_k, \delta_k)$, where $\kappa_{\rm ubh} \stackrel{\rm def}{=} (p+q)\kappa_{\rm ubc}^2 \kappa_{\rm ub}(1 + \kappa_{\rm ub})$.

Proof: Due to Assumptions 4 and 5, the bounds (3.2) hold. To prove the desired result, we first state some additional error bounds. Let us first consider the error caused by the use of m_k^u instead of u within a single constraint function c_i, $i \in \mathcal{E} \cup \mathcal{I}$, at $x \in \mathcal{Q}(x_k, \delta_k)$. A Taylor expansion yields that

$$|c_i(x, u(x)) - c_i(x, m_k^u(x))| \leq \|\nabla c_i(x, \zeta)\|_2 \|u(x) - m_k^u(x)\|_2 \tag{3.5}$$

for some $\zeta \in \mathbb{R}^m$ on the line segment connecting $u(x)$ and $m_k^u(x)$. As $x \in \Omega$ and $\zeta \in \Omega^*$, we obtain from (3.2) that $\|\nabla c_i(x, \zeta)\|_2 \leq \kappa_{\rm ubc}$. Due to Assumption 1, (2.5) holds and we obtain from (3.5) that

$$|c_i(x, u(x)) - c_i(x, m_k^u(x))| \leq \kappa_{\rm ubc} \kappa_{\rm ub} \delta_k^2. \tag{3.6}$$

Analogously, we obtain for the gradient ∇c_i, $i \in \mathcal{E} \cup \mathcal{I}$, that

$$\|\nabla c_i(x, u(x)) - \nabla c_i(x, m_k^u(x))\|_2 \leq \|\nabla^2 c_i(x, \xi)\|_2 \|u(x) - m_k^u(x)\|_2 \leq \kappa_{\text{ubc}} \kappa_{\text{ub}} \delta_k^2 \tag{3.7}$$

for some $\xi \in \Omega^*$ on the line segment connecting $u(x)$ and $m_k^u(x)$. Next we derive an upper bound for $\nabla c_i(x, m_k^u(x))$ using the triangle inequality, (3.7) and (3.2) and obtain that

$$\|\nabla c_i(x, m_k^u(x))\|_2 \leq \|\nabla c_i(x, u(x)) - \nabla c_i(x, m_k^u(x))\|_2 + \|\nabla c_i(x, u(x))\|_2 \leq \kappa_{\text{ubc}} \kappa_{\text{ub}} \delta_k^2 + \kappa_{\text{ubc}}. \tag{3.8}$$

Moreover note that case differentiation directly yields that

$$|[c_i(x, u(x))]_+ - [c_i(x, m_k^u(x))]_+| \leq |c_i(x, u(x)) - c_i(x, m_k^u(x))|. \tag{3.9}$$

Having obtained these preliminary error bounds, we next prove (3.3). Successively using the definition of f and ϑ, Assumption 4, the triangle inequality and the fact that $\|\cdot\|_2 \leq \|\cdot\|_1$, we obtain that

$$\begin{aligned}
|f(x, u(x)) - f(x, m_k^u(x))| &= \frac{1}{2}\Big| \|\vartheta(x, u(x))\|_2^2 - \|\vartheta(x, m_k^u(x))\|_2^2 \Big| \\
&= \frac{1}{2}\Big| \|c_{\mathcal{E}}(x, u(x))\|_2^2 - \|c_{\mathcal{E}}(x, m_k^u(x))\|_2^2 \\
&\quad + \|[c_{\mathcal{I}}(x, u(x))]_+\|_2^2 - \|[c_{\mathcal{I}}(x, m_k^u(x))]_+\|_2^2 \Big| \\
&\leq \frac{1}{2} \cdot 2\kappa_{\text{ubc}} \Big| \|c_{\mathcal{E}}(x, u(x)) - c_{\mathcal{E}}(x, m_k^u(x))\|_2 \\
&\quad + \|[c_{\mathcal{I}}(x, u(x))]_+ - [c_{\mathcal{I}}(x, m_k^u(x))]_+\|_2 \Big| \\
&\leq \kappa_{\text{ubc}} \left(\sum_{i \in \mathcal{E}} \Big| c_i(x, u(x)) - c_i(x, m_k^u(x)) \Big| \right. \\
&\quad \left. + \sum_{i \in \mathcal{I}} \Big| [c_i(x, u(x))]_+ - [c_i(x, m_k^u(x))]_+ \Big| \right).
\end{aligned}$$

Using now (3.9) and (3.6), we directly deduce that

$$|f(x, u(x)) - f(x, m_k^u(x))| \leq (p+q)\kappa_{\text{ubc}}^2 \kappa_{\text{ub}} \delta_k^2 \leq \kappa_{\text{ubh}} \delta_k^2.$$

Let us now consider (3.4) and note that

$$\nabla f(x, u(x)) = \sum_{i \in \mathcal{E}} c_i(x, u(x)) \nabla c_i(x, u(x)) + \sum_{i \in \mathcal{I}} [c_i(x, u(x))]_+ \nabla c_i(x, u(x))$$

and thus
$\|\nabla f(x, u(x)) - \nabla f(x, m_k^u(x))\|_2$

$$\begin{aligned}
&\leq \sum_{i \in \mathcal{E}} \|c_i(x, u(x)) \nabla c_i(x, u(x)) - c_i(x, m_k^u(x)) \nabla c_i(x, m_k^u(x))\|_2 \\
&\quad + \sum_{i \in \mathcal{I}} \|[c_i(x, u(x))]_+ \nabla c_i(x, u(x)) - [c_i(x, m_k^u(x))]_+ \nabla c_i(x, m_k^u(x))\|_2 \\
&\leq \sum_{i \in \mathcal{E}} \left[|c_i(x, u(x))| \|\nabla c_i(x, u(x)) - \nabla c_i(x, m_k^u(x))\|_2 \right. \\
&\quad \left. + |c_i(x, u(x)) - c_i(x, m_k^u(x))| \|\nabla c_i(x, m_k^u(x))\|_2 \right] \\
&\quad + \sum_{i \in \mathcal{I}} \left[|[c_i(x, u(x))]_+| \|\nabla c_i(x, u(x)) - \nabla c_i(x, m_k^u(x))\|_2 \right. \\
&\quad \left. + |[c_i(x, u(x))]_+ - [c_i(x, m_k^u(x))]_+| \|\nabla c_i(x, m_k^u(x))\|_2 \right].
\end{aligned}$$

As $|[c_i(x,u(x))]_+| \leq |c_i(x,u(x))|$ and (3.9) hold, we obtain that

$$\begin{aligned}\|\nabla f(x,u(x)) - \nabla f(x,m_k^u(x))\|_2 &\leq \sum_{i\in\mathcal{E}\cup\mathcal{I}} [|c_i(x,u(x))|\|\nabla c_i(x,u(x)) - \nabla c_i(x,m_k^u(x))\|_2 \\ &\quad +|c_i(x,u(x)) - c_i(x,m_k^u(x))|\|\nabla c_i(x,m_k^u(x))\|_2]\,.\end{aligned}$$

Using the bounds (3.2), (3.7), (3.6) and (3.8), we may finally conclude that

$$\begin{aligned}\|\nabla f(x,u(x)) - \nabla f(x,m_k^u(x))\|_2 &\leq (p+q)[\kappa_{\rm ubc}^2\kappa_{\rm ub}\delta_k^2 + \kappa_{\rm ubc}\kappa_{\rm ub}\delta_k^2(\kappa_{\rm ubc}\kappa_{\rm ub}\delta_k^2 + \kappa_{\rm ubc})] \\ &\leq (p+q)\kappa_{\rm ubc}^2\kappa_{\rm ub}(1+\kappa_{\rm ub})\max[\delta_k^2,\delta_k^4] \\ &= \kappa_{\rm ubh}\max[\delta_k^2,\delta_k^4].\end{aligned}$$

□

Analogously to the convergence analysis in Section 9.2 in [4], we use (3.4) to derive an error bound for the gradient of the original objective function $f(x,u(x))$ in (1.1). This is necessary because, due to the lack of derivative information of u, the only gradient information available is that with respect to the model m_k^u. Thus we can only test $g_k = \nabla f(x,m_k^u(x))$ within the stopping test in Step 1 in Algorithm 2.1. How do we reconcile this information with our desire to detect if $\nabla f(x,u(x))$ is small, which indicates that we have found a first-order critical point? The following lemma yields a suitable error bound on the norm of $\nabla f(x,u(x))$.

Lemma 3.3 (similar to Lemma 9.2.4 in [4]) *Suppose that Assumptions 1 and 4–6 hold. Suppose furthermore that x_k is the current iterate, $\|g_k\|_2 < \varepsilon_{\rm end}$ and that m_k^u is valid in $\mathcal{Q}_k(x_k,\delta_{\rm end})$ for some $\delta_{\rm end} \in (0,\mu\|g_k\|_2]$, $\mu > 0$. Then*

$$\|\nabla f(x_k,u(x_k))\|_2 \leq \kappa_{\rm end}\max[\varepsilon_{\rm end},\varepsilon_{\rm end}^4], \tag{3.10}$$

where

$$\kappa_{\rm end} \stackrel{\rm def}{=} (1+\kappa_{\rm ubh})\max[1,\mu^4].$$

Furthermore, if

$$\lim_{k\to\infty}\|g_k\|_2 = 0,$$

then

$$\lim_{k\to\infty}\|\nabla f(x_k,u(x_k))\|_2 = 0.$$

Proof: As Assumptions 1 and 4–6 hold and m_k^u is valid in $\mathcal{Q}_k(x_k,\delta_{\rm end})$, we can apply Lemma 3.2 and obtain, using successively triangle inequality, (3.4), the definition of g_k and the assumptions,

$$\begin{aligned}\|\nabla f(x_k,u(x_k))\|_2 &\leq \|\nabla f(x_k,u(x_k)) - \nabla f(x_k,m_k^u(x_k))\|_2 + \|\nabla f(x_k,m_k^u(x_k))\|_2 \\ &\leq \kappa_{\rm ubh}\max[\delta_{\rm end}^2,\delta_{\rm end}^4] + \|g_k\|_2 \\ &\leq \kappa_{\rm ubh}\max[\mu^2\|g_k\|_2^2,\mu^4\|g_k\|_2^4] + \|g_k\|_2 \\ &\leq (1+\kappa_{\rm ubh})\max[1,\mu^4]\max[\|g_k\|_2,\|g_k\|_2^4] \\ &\leq \kappa_{\rm end}\max[\varepsilon_{\rm end},\varepsilon_{\rm end}^4]\end{aligned}$$

This proves the first statement. The second statement follows immediately from the penultimate line of this chain of inequalities. □

The assumptions in Lemma 3.3 correspond to the situation of termination in Step 1 in Algorithm 2.1, and therefore the bound (3.10) holds for the point x_k that is returned as approximate solution of problem (ES). In practice, it might be reasonable to choose $\varepsilon_{\rm end} \ll 1$. In this situation, (3.10) reduces to

$$\|\nabla f(x_k,u(x_k))\|_2 \leq (1+\kappa_{\rm ubh})\max[1,\mu^4]\varepsilon_{\rm end}.$$

As $\kappa_{\rm ubh}$ might be large, it is reasonable to choose a small $\varepsilon_{\rm end}$ to achieve a good accuracy. For the convergence analysis of Algorithm 2.1, the second statement of Lemma 3.3 ensures that it is sufficient to investigate the convergence properties of the sequence $\{g_k\}_k$ instead of $\{\nabla f(x_k,u(x_k))\}_k$.

We now pursue our convergence analysis for Algorithm 2.1. In trust-region methods, the acceptance of a trial point x_k^+ and the update of the trust-region radius δ_k are usually performed using the ratios of achieved to predicted reductions ρ_k. In Algorithm 2.1, both are performed using the modified ratio $\hat{\rho}_k$. Thus we have to revisit our definition of a successful iteration: we now defined the set of successful iterations by

$$\mathcal{S} \stackrel{\rm def}{=} \{k \geq 0 \mid \hat{\rho}_k \geq \eta_1\}.$$

Lemma 9.1.2 in [4] then enables us to link ρ_k and $\hat{\rho}_k$ and we obtain the following two properties.

Lemma 3.4 (part of Lemma 9.1.2 in [4]) *Suppose that Assumptions 2–3 hold. Then the following statements hold.*

1. *If $\rho_k \geq \eta_2$, then $\hat{\rho}_k \geq \eta_2 \geq \eta_1$, and thus iteration k is very successful. Moreover, $\delta_{\rm ref} = \delta_k$ is set whenever $k \in \mathcal{S}$.*
2. *There can only be a finite number of iterations such that $\rho_k < \eta_2$ before $\delta_{\rm ref} = \delta_k$ is set.*

The following lemma shows that the trust-region radius δ_k is bounded away from zero if g_k is also bounded away from zero. This is important since otherwise the algorithm might get stuck at a non-critical iterate.

Lemma 3.5 (Theorem 9.2.1 in [4]) *Suppose that Assumptions 1-7 hold. Suppose furthermore that $\|g_k\|_2 > \varepsilon$ for all k and some $\varepsilon > 0$. Then*

$$\delta_k \geq \gamma_0 \min\left[\frac{\gamma_0 \kappa_{\rm mdcd} \varepsilon (1-\eta_2)}{\max[\kappa_{\rm ubh}, \kappa_{\rm umh}]}, \varepsilon_{\rm mdc}\right] \stackrel{\rm def}{=} \kappa_{\rm lbd}$$

for all k.

Proof: Because $\|g_k\|_2 > \varepsilon$, Assumption 7 guarantees that x_k^+ exists and satisfies (2.10) whenever $\delta_k \leq \varepsilon_{\rm mdc}$. The desired result now follows exactly as in the proof of Theorem 9.2.1 in [4] with adapted notation. □

The next step in our analysis is to investigate the situation where only finitely many successful iterations are generated by Algorithm 2.1.

Lemma 3.6 (Theorem 9.2.4 in [4]) *Suppose that Assumptions 1-7 hold. Suppose furthermore that there are only finitely many successful iterations. Then $x_k = x^*$ for all sufficiently large k and x^* is first-order critical.*

The only remaining case is thus that where infinitely many successful iterations are produced by Algorithm 2.1. First, we show that the sequence of iterates $\{x_k\}$ has at least one accumulation point x^* with $\nabla f(x^*, m_k^u(x^*)) = 0$.

Lemma 3.7 (Theorem 9.2.5 in [4]) *Suppose that Assumptions 1-7 hold. Then one has that*

$$\liminf_{k\to\infty} \|\nabla f(x_k, u(x_k))\|_2 = 0.$$

We may now directly state the final convergence result from Section 9.2 in [4], in the proof of which (2.9) plays a crucial role.

Lemma 3.8 (Theorem 9.2.6 in [4]) *Suppose that Assumptions 1-7 hold. Then every limit point x^* of the sequence $\{x_k\}$ is first-order critical, that is, $\nabla f(x^*, u(x^*)) = 0$.*

Combining this with Lemma 3.1 allows us to state our final convergence result.

Theorem 3.9 *Suppose that Assumptions 1-7 hold. Then every limit point x^* of the sequence $\{x_k\}$ is first-order critical, that is, $\nabla f(x^*, u(x^*)) = 0$. Moreover, if infinitely many entries are added to the filter, then we have that $\lim_{k\to\infty} \|\vartheta(x_k, u(x_k))\|_2 = 0$.*

4 Computing the trial point

In this section we concentrate on the question of determining a trial point x_k^+ in Step 2 of Algorithm 2.1 that satisfies Assumption 7. In standard trust-region methods, the model of the original objective function is usually a rather simple, in many cases a quadratic model of the objective function in the unconstrained case (see, e.g. Chapter 6 in [4]) or of the Lagrangian in the constrained case (see, e.g. [10]). The main advantage of these "easy" models is the existence of highly specialized subproblem solvers directly constructed to determine a trial point that automatically satisfies a sufficient model decrease condition analogous to (2.10), see, e.g. Chapter 6.3 in [4]. For *trust-region subproblem* solvers for quadratic models see, e.g. [4, 9, 13, 26]. In our case, the trust-region subproblem (CS_k) is still "easy" compared to the original problem (ES), in the sense that function evaluations are fast and derivative information is accessible, which suggests that the determination of the trial point could also be fast. Observe at this stage that the least-squares formulation of the subproblem comes in three very similar flavors, depending on the $RESTRICT$ flag and the value of k: if $RESTRICT$ is set, the subproblem is defined by (2.7), while it is given by

$$\begin{array}{lrcl} \min\limits_{x\in\mathbb{R}^n} & f(x, m_k^u(x)) & = & \frac{1}{2}\|\vartheta(x, m_k^u(x))\|_2^2 \\ s.t. & x & \in & \mathcal{Q}(x_k, \kappa_\delta \delta_k). \end{array} \tag{4.1}$$

if $RESTRICT$ is unset and $k \geq k_0$, and by

$$\min_{x\in\mathbb{R}^n} \quad f(x, m_k^u(x)) \quad = \quad \frac{1}{2}\|\vartheta(x, m_k^u(x))\|_2^2 \tag{4.2}$$

if $RESTRICT$ is unset and $k < k_0$.

4.1 Using a trust-region inner solver

Modeling the subproblem as a general nonlinear feasibility problem (CS_k) as shown in Section 2 has the advantage to maintain all the available information. However, since the problem (CS_k) remains arbitrarily nonlinear, (2.10) cannot be guaranteed in general. Even the *model minimizer*, the global solution of (2.7), need not to satisfy a sufficient model decrease condition in all situations. Nevertheless, it is usually a good candidate as it predicts the best reduction of f within the current trust-region $\mathcal{Q}_k$ in iteration k. Our strategy is thus to attempt to generate this model minimizer in Step 2 of Algorithm 2.1 using an *inner solver*. But since we also need to guarantee (2.10), we have to add an additional criterion to this inner solver. Note that (2.10) is Cauchy condition of the type guaranteed by typical trust-region methods. It therefore makes sense to consider an algorithm of this class as inner solver. The trial points generated within this solver then satisfy a sufficient model decrease condition in the inner (typically quadratic) model used inside this method, which we denote by q. Our main idea is now to link this inner sufficient decrease with the outer model decrease we need in Step 2 of Algorithm 2.1. To be able to describe this, we need some further notation. We denote the iterates generated within our trust-region inner solver by $y_{k,p}$ and the associated trial points by $y_{k,p}^+$ with double indices (k, p), where k denotes the iteration in Algorithm 2.1 (the outer solver) and p the current iteration within the inner solver. The trust-region radii of the inner solver are denoted by $\Delta_{k,p}$ and the corresponding trust-region is then given by

$$\mathcal{B}_{k,p}(y_{k,p}, \Delta_{k,p}) \stackrel{\text{def}}{=} \{y \in \mathbb{R}^n : \|y - y_{k,p}\|_\infty \leq \Delta_{k,p}\}.$$

Once we have applied our inner trust-region solver to either (2.7), (4.1) or (4.2) and produced some feasible approximate solution $\overline{y}$ of this problem, we must then decide if $\overline{y}$ can be accepted as a trial point x_k^+ in the outer solver. We propose the following test.

Acceptance Test 1 *The feasible outcome $\bar{y}$ produced by the inner solver is accepted as new trial point x_k^+ if and only if*

$$\frac{f(x_k, m_k^u(x_k)) - f(\bar{y}, m_k^u(\bar{y}))}{q_{k,0}(y_{k,0}) - q_{k,0}(y_{k,0}^+)} \geq \kappa_{\delta\Delta} \tag{4.3}$$

for some $\kappa_{\delta\Delta} \in (0, \eta_2]$. Otherwise, we declare the outer iteration k to be a failed iteration, that is $k \in \mathcal{F}$.

The condition (4.3) guarantees that the model reduction in the cheap nonlinear objective function of (2.7) at the trial point is at least a fraction of the predicted model decrease in the first iteration of the inner solver. To guarantee the desired sufficient model decrease (2.10), we need the following assumptions on the initialization of the inner solver.

Assumption 8 *(Conditions on the inner solver)*

- *The inner subproblem solver is a trust-region algorithm initialized with trust-region radius* $\Delta_{k,0} = \delta_k$ *and starting point* $y_{k,0} = x_k$.
- β_k *is sufficiently large such that it is also an upper bound on the Hessian of* $q_{k,0}$*, e.g.* $\beta_{k,0} \leq \beta_k$.
- *All iterates* $y_{k,p}$ *generated by the inner solver are feasible in the sense that they belong to* $\mathcal{Q}(x_k, \delta_k)$ *if* $RESTRICT$ *is set, or to* $\mathcal{Q}(x_k \kappa_\delta \delta_k)$ *if* $RESTRICT$ *is unset and* $k \geq k_0$.

These conditions are very reasonable and can be guaranteed very easily since the global convergence theory for trust-region methods is independent of the choice of the initial starting point and of the initial trust-region radius.

The following corollary now directly shows that the desired sufficient decrease condition (2.10) holds for any point passing Acceptance Test 1.

Corollary 4.1 *Suppose that the inner solver satisfies Assumption 8. Then, every point* $\bar{y}$ *passing the Acceptance Test 1 also satisfies* (2.10).

Proof: Observe first that, because of item 1 in Assumption 8, we have that the trust region $\mathcal{B}_{k,0}(y_{k,0}, \Delta_{k,0})$ is entirely included in the feasible domain of the subproblem. Hence iteration 0 of the subproblem solver computes a trial point minimizing the model $q_{k,0}$ within a trust region of radius $\Delta_{k,0} = \delta_k$. The standard Cauchy condition therefore applies for this first minimization (see Section 6.3 of [4]), yielding that

$$q_{k,0}(y_{k,0}) - q_{k,0}(y_{k,0}^+) \geq \kappa_{\rm mdc} \|g_k\|_2 \min\left[\frac{\|g_k\|_2}{\beta_{k,0}}, \Delta_{k,0}\right] \geq \kappa_{\rm mdc} \|g_k\|_2 \min\left[\frac{\|g_k\|_2}{\beta_k}, \delta_k\right] \tag{4.4}$$

for some constant $\kappa_{\rm mdc} \in (0,1)$, where we have used Assumption 8 to deduce the second inequality. As a consequence, we obtain from (4.3) that

$$f(x_k, m_k^u(x_k)) - f(\bar{y}, m_k^u(\bar{y})) \geq \kappa_{\delta\Delta}[q_{k,0}(y_{k,0}) - q_{k,0}(y_{k,0}^+)] \geq \kappa_{\rm mdcd} \|g_k\|_2 \min\left[\frac{\|g_k\|_2}{\beta_k}, \delta_k\right]$$

with $\kappa_{\rm mdcd} \stackrel{\rm def}{=} \kappa_{\delta\Delta}\kappa_{\rm mdc}$, and the conclusion follows. □

This result directly implies that the first part of Assumption 7 holds under Acceptance Test 1. It also allows us to formulate our algorithmic framework for the determination of the trial point x_k^+ in iteration k of Algorithm 2.1 as follows.

Algorithm 4.1 ***Computation of the trial point at Step 2 of Algorithm 2.1***

Step 1: (Solve the cheap problem) Solve Problem (2.7) using an (inner) trust-region method for which Assumption 8 holds. Let

$$\mathcal{Y} \stackrel{\rm def}{=} \{y_{k,p} \mid y_{k,p} \neq y_{k,0}\} \tag{4.5}$$

be the set of (feasible) iterates produced by this trust-region method beyond the starting point. Save $q_{k,0}(y_{k,0})$ and $q_{k,0}(y_{k,0}^+)$.

Step 2: (Trial point) Choose $\bar{y} \in \mathcal{Y}$ such that the Acceptance Test 1 is satisfied and set $x_k^+ \stackrel{\rm def}{=} \bar{y}$. If the acceptance test fails for all $y_{k,p} \in \mathcal{Y}$ or if $\mathcal{Y} = \emptyset$, declare iteration k failed, i.e. $k \in \mathcal{F}$.

If there are several iterates in $\mathcal{Y}$ satisfying Acceptance Test 1, the theory does not specify which one to choose. In practice, an iterate with a low value of $f(\bar{y}, m_k^u(\bar{y}))$ is typically preferred. Note that $\mathcal{Y}$ might include, in addition to what is specified by (4.5), other candidate points that could be generated using some other subproblem solver.

To complete our discussion on how to determine the trial point, we finally have to ensure that the inner trust-region method applied in Step 1 of Algorithm 4.1 generates at least one feasible $\bar{y} \in \mathcal{Y}$ that passes Acceptance Test 1 if we are in the situation of Assumption 7, i.e. if the gradient g_k is still large and the trust-region radius δ_k is sufficiently small. We use the following lemma from the trust-region convergence theory for this purpose.

Lemma 4.2 (Lemma 6.4.2 in [4]) *Suppose that Assumptions 4 and 5 hold, that there exists a constant* $\kappa_{\rm umh} > 0$ *such that*

$$|f(y_{k,0}^+, m_k^u(y_{k,0}^+)) - q_{k,0}(y_{k,0}^+)| \leq \kappa_{\rm umh} \Delta_{k,0}^2 \tag{4.6}$$

and that $\|y_{k,0} - y_{k,0}^+\|_\infty \leq \Delta_{k,0}$. *Suppose furthermore that* $g_k \neq 0$ *and that*

$$\Delta_{k,0} \leq \frac{\kappa_{\rm mdc} \|g_k\|_2 (1-\eta_2)}{\kappa_{\rm umh}}. \tag{4.7}$$

Then we have that iteration $k,0$ *is very successful, e.g.* $\rho_{k,0} \geq \eta_2$ *and* $\Delta_{k,1} \geq \Delta_{k,0}$.

The existence of $\kappa_{\rm umh}$ satisfying (4.6) follows under standard trust-region assumptions, see the Assumptions AF.1, AF.3, AM.1-AM.4 and Lemma 6.4.1 in [4]. This is analogous to stating Lemma 3.2 in our framework. The following theorem then guarantees that Algorithm 4.1 satisfies the second part of Assumption 7.

Theorem 4.3 *Suppose that Assumption 4, 5 and 8 hold. Suppose that we apply a trust-region algorithm as subproblem solver in Algorithm 4.1 holds. Suppose furthermore that RESTRICT is set in iteration* k, $\|g_k\|_2 > \varepsilon$ *and*

$$\delta_k \leq \frac{\kappa_{\rm mdc} \varepsilon (1-\eta_2)}{\kappa_{\rm umh}} \stackrel{\rm def}{=} \varepsilon_{\rm mdc}.$$

Then Assumption 7 is satisfied for iteration k.

Proof: We first verify that

$$\Delta_{k,0} = \delta_k \leq \frac{\kappa_{\rm mdc} \varepsilon (1-\eta_2)}{\kappa_{\rm umh}} \leq \frac{\kappa_{\rm mdc} \|g_k\|_2 (1-\eta_2)}{\kappa_{\rm umh}}.$$

Therefore, using Lemma 4.2, iteration $(k,0)$ is very successful, the trial point $y_{k,0}^+$ is accepted, yielding $y_{k,0} \neq y_{k,1} \stackrel{\rm def}{=} y_{k,0}^+ \in \mathcal{Y}$, and $y_{k,1}$ passes the Acceptance Test 1 because $\rho_{k,0} \geq \eta_2 \geq \kappa_{\delta\Delta}$. Thus, Algorithm 4.1 produces a trial point x_k^+. As our assumptions also imply those of Corollary 4.1, x_k^+ also satisfies Assumption 7. □

Algorithm 2.1 is therefore well-defined and, in view of Section 3, converges to first-order critical points, as desired.

Our choice of inner solver is to use the filter-trust-region method for bound constrained optimization described in [31]. This method has to advantage that it can be made to conform to our requirements (Assumption 8) and provides excellent reliability and efficiency.

5 Numerical results

In this section we test the practical behaviour of Algorithm 2.1 (with parameter values as specified after the algorithm's statement, in particular with $\varepsilon_{\rm end} = 10^{-6}$). The algorithm is implemented in Matlab and its implementation is called EFNES[(2)] in the following. For a comparison with existing algorithms, we

[(2)] solver for Expensive Function based Nonlinear Equation Systems

apply EFNES to the set of 54 test problems described in [20]. These test problems are originally a subset of the CUTEr collection [14] and are modified to simulate inequality constraints and the existence of one or more nonlinear expensive functions. Since many engineering applications only have a small number of variables (often called design variables) and a small number of outcomes of the expensive function, the test set contains mainly problems with less than 10 variables, but also problems up to 100 variables are included. These problems are fully described in [20].

5.1 Using available information

The nested function formulation of the feasibility problems (ES) provides considerable freedom and gives the possibility of incorporating as much as possible of the problem description in the (cheap) outer functions instead of within the expensive function for which derivatives are unavailable, leading in turn to improved performance compared with pure derivative-free techniques. This ability is often useful in practical applications, for example, in optimization involving numerical simulations where only a part of the problem is actually simulated. The following example shows the effect of varying the part of the problem description that is considered as expensive and for which derivatives are unavailable. Let us consider the test problem `ARGAUSS` from the CUTEr collection, which is the following nonlinear system of equations (without any expensive function):

$$c_i \stackrel{\text{def}}{=} x_1 e^{\frac{x_2}{2}(\alpha_i - x_3)^2} - \beta_i = 0$$

for $i = 1, \ldots, 15$, where $\alpha_i = 4 - \frac{i}{2}$ and $\beta_i \in \mathbb{R}$ are small constants, see [20]. For this system, we artificially define expensive functions e_i, $i = 1, \ldots, 15$, and gradually expand the part of the functions c_i that is defined to be expensive. The corresponding number of expensive function evaluations and the runtime of EFNES (excluding the time for the evaluation of the expensive function) are given in Table 5.1. The drawbacks of augmenting the expensive/non-derivative part in the problem description appear clearly.

	$e_i = (\alpha - x_3)^2$	$e_i = \frac{x_2}{2}(\alpha_i - x_3)^2$	$e_i = e^{\frac{x_2}{2}(\alpha_i - x_3)^2}$	$e_i = x_1 e^{\frac{x_2}{2}(\alpha_i - x_3)^2} - \beta_i$
CPU time (s)	0.44	1.48	1.89	2.41
evaluations	5	20	22	26

Table 5.1: Effect of different definitions of expensive functions in problem `ARGAUSS`

5.2 Comparison with other methods

We now compare our method to four other solvers for nonlinear feasibility problems. As EFNES is primarily based on the filter trust-region method FILTRANE [17], we have chosen FILTRANE as part of the GALAHAD package [15] for comparison, although it primary aims at large scale problems. Moreover, we consider TRESNEI [28], which is another trust-region method, as well as `fmincon` and the derivative free function `patternsearch` from the optimization and direct search toolbox in Matlab, respectively. All tests mentioned in this section were run on a workstation with a 2.8 GHz dual-core processor and 1GB of memory, openSUSE 11.1, Fortran compiler g95 0.91 and Matlab 7.

All algorithms are compared in terms of the number of evaluations of the expensive function and the required computation time, the latter being measured using the tools provided by Matlab (or by the operation system in case of FILTRANE) to obtain comparable results. The required number of function evaluations of `fmincon` and `patternsearch` are part of their output. TRESNEI also provides information about the number of evaluations, but without consideration of the additional function evaluations necessary for building the Jacobian. We therefore added these to obtain the final total number of expensive function evaluations. We ran FILTRANE using forward differentiation to obtain derivatives, in order to simulate the lack of derivative information assumed for expensive function.

We stop EFNES, FILTRANE, TRESNEI and `fmincon`, if the norm of the gradient of f is smaller than 10^{-6}, and `patternsearch` if the grid width is smaller than 10^{-6}.
Figure 5.2 presents a performance profile [8] comparing all codes in terms of number of evaluations of the expensive function. For every $s \geq 1$, a performance profile shows the fraction $p(\ln(s))$ of test problems,

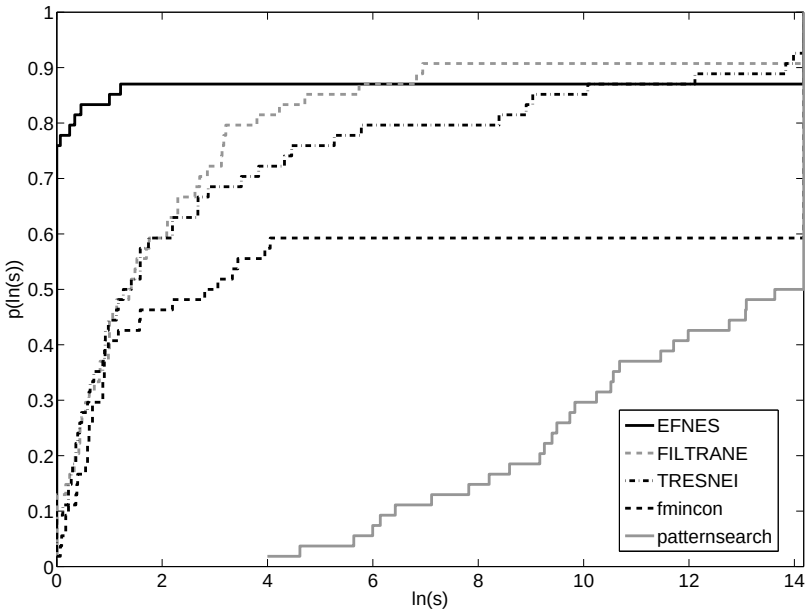

Figure 5.2: Performance profile for the number of expensive function evaluation

on which the considered algorithm has solved the problem within time or expensive function evaluation, respectively, given by a factor s of the best. For a better visualization, the $s-$axis is chosen to be logarithmic.

The numerical results show a considerable decrease in the number of expensive function evaluations by EFNES on this set of test problems. It is also notable that EFNES is almost as reliable as the other trust-region methods. The problems on which the algorithms fail are listed in the appendix.
Figure 5.3 shows the performance profile in terms of computation time. Obviously, the pure computation time of EFNES is, in general, significantly larger than that of the best solver because in every iteration, the complete nonlinear feasibility problem (CS_k) has to be solved at every iteration. Taking into account in addition the evaluation time of the expensive function, the total runtime is completely different due to the smaller number of expensive function evaluations in EFNES. On our set of testproblems, EFNES would be faster than FILTRANE if the evaluation of an expensive function needed more than 0.61 seconds (respectively 0.17 seconds if only those problems that are solved by both algorithms are taken into account). Compared to TRESNEI, EFNES would be faster if an evaluation needed more than 37.69 (respectively 9.56) seconds[(3)]. Our new method is therefore extremely competitive in the frequent practical case where simulation time dominates total execution.

6 Conclusion

We have introduced a new algorithm for solving feasibility problems that involve expensive function evaluations whose derivatives are unavailable, a situation which is common in simulation-based optimization. This algorithm combines the trust-region ideas with the filter techniques of Fletcher and Leyffer [12] to achieve feasibility. It applies conditional models for the expensive function to keep, on average, the number of evaluations of such functions low. Moreover, a nested approach for the problem formulation provides considerable flexibility and allows exploiting the cheap part of the problem and available derivative information to a very large extent, thereby increasing optimization efficiency.

[(3)] The fact that, in average, the pure computation time of FILTRANE is longer compared to TRESNEI might come from the fact that FILTRANE is designed for large scale problems, which causes significant numerical overhead for the small scale problems we have used.

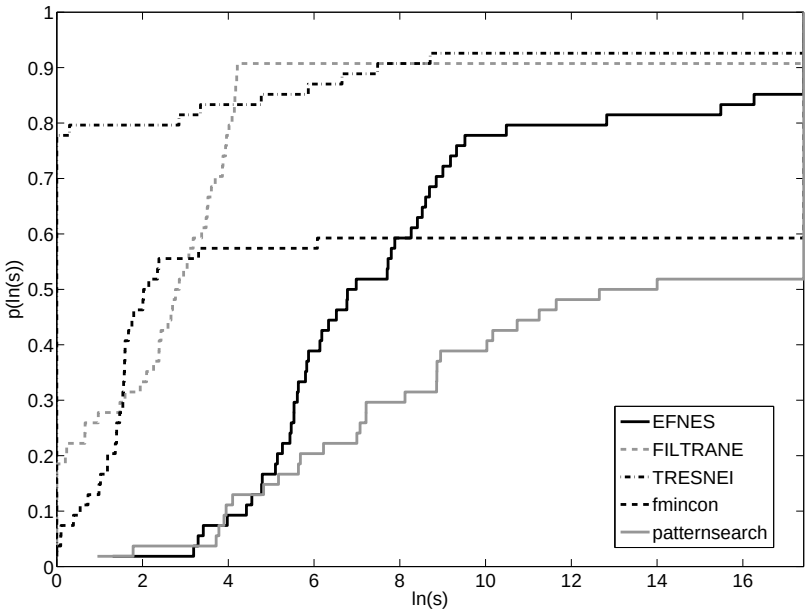

Figure 5.3: Performance profile comparing the computation time

We prove convergence of the new algorithm to a first-order critical point of a least-squares reformulation of the feasibility problem from an arbitrary starting point under very standard assumptions. Numerical results on a set of 54 test problems show a considerable reduction of expensive function evaluations for the new method when compared to four other algorithms for feasibility problems. If the functions are expensive in terms of evaluation time, our algorithm turns out to be more effective on average if a single function evaluation takes longer than a few seconds. The competitive advantage is therefore extremely significant in practical settings where simulation may take up to several hours to produce a function value.

It is clearly interesting to apply this approach in the context of constrained optimization problems, an approach which is currently under investigation.

Acknowledgement

The content of this work was partially funded by the Elite Network of Bavaria within the International Doctorate Program *Identification, Optimization and Control with Applications in Modern Technologies.*

References

[1] Benoît Colson. *Trust-Region Algorithms for Derivative-Free Optimization and Nonlinear Bilevel Programming.* PhD thesis, Facultés Universitaires Notre-Dame de la Paix, Namur, Belgium, 2003.

[2] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis*, 25(2):433–460, 1988.

[3] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. Testing a class of methods for solving minimization problems with simple bounds on the variables. *Mathematics of Computation*, 50(182):399–430, 1988.

[4] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. *Trust-Region Methods.* MPS-SIAM Series on Optimization. Society for Industrial Mathematics, 2000.

[5] Andrew R. Conn, Katya Scheinberg, and Luís N. Vicente. Geometry of interpolation sets in derivative free optimization. *Mathematical Programming*, 111(1):141–172, 2007.

[6] Andrew R. Conn, Katya Scheinberg, and Luís N. Vicente. *Introduction to Derivative-Free Optimization*. MPS-SIAM Series on Optimization. Society for Industrial Mathematics, 2009.

[7] John E. Dennis, Mahmoud El-Alem, and Karen Williamson. A trust-region approach to nonlinear systems of equalities and inequalities. *SIAM Journal on Optimization*, 9(2):291–315, 1999.

[8] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[9] Jennifer B. Erway, Philip E. Gill, and Joshua D. Griffin. Iterative methods for finding a trust-region step. Technical Report NA 07-02, Department of Mathematics, University of California, San Diego, USA, 2007.

[10] Roger Fletcher, Nicholas I. M. Gould, Sven Leyffer, Philippe L. Toint, and Andreas Wächter. Global convergence of a trust-region SQP-filter algorithm for general nonlinear programming. *SIAM Journal on Optimization*, 13(3):635–659, 2002.

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

[12] Nicholas I. M. Gould, Sven Leyffer, and Philippe L. Toint. A multidimensional filter algorithm for nonlinear equations and nonlinear least-squares. *SIAM Journal on Optimization*, 15(1):17–38, 2005.

[13] Nicholas I. M. Gould, Stefano Lucidi, Massimo Roma, and Philippe L. Toint. Solving the trust-region subproblem using the Lanczos method. *SIAM Journal on Optimization*, 9(2):504–525, 1999.

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

[15] Nicholas I. M. Gould, Dominique Orban, and Philippe L. Toint. GALAHAD, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization. *ACM Transactions on Mathematical Software*, 29(4):353–372, 2003.

[16] Nicholas I. M. Gould, Caroline Sainvitu, and Philippe L. Toint. A filter-trust-region method for unconstrained optimization. *SIAM Journal on Optimization*, 16(2):341–357, 2005.

[17] Nicholas I. M. Gould and Philippe L. Toint. FILTRANE, a Fortran 95 filter-trust-region package for solving nonlinear least-squares and nonlinear feasibility problems. *ACM Transactions on Mathematical Software*, 33(1):3–25, 2007.

[18] Serge Gratton, Annick Sartenaer, and Philippe L. Toint. Recursive trust-region methods for multiscale nonlinear optimization. *SIAM Journal on Optimization*, 19(1):414–444, 2008.

[19] Crina Grosan and Ajith Abraham. A new approach for solving nonlinear equations systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 38(3):698–714, 2008.

[20] Markus Kaiser, Kathrin Klamroth, and Alexander Thekale. Test examples for nonlinear feasibility problems with expensive functions. Technical report, University of Erlangen-Nuremberg, Germany, 2010.

[21] Chih-Jen Lin and Jorge J. Moré. Newton's method for large bound-constrained optimization problems. *SIAM Journal on Optimization*, 9(4):1100–1127, 1999.

[22] Maria Macconi, Benedetta Morini, and Margherita Porcelli. Trust-region quadratic methods for nonlinear systems of mixed equalities and inequalities. *Applied Numerical Mathematics*, 59(5):859–876, 2009.

[23] José Mario Martínez. Algorithms for solving nonlinear systems of equations. In E. Spedicato, editor, *Algorithms For Continuous Optimization, The State Of The Art*, pages 81–108. Kluwer Academic Publishers, 1994.

[24] Jorge J. Moré. Trust regions and projected gradients. In M. Iri and K. Yajima, editors, *System Modelling and Optimization Proceedings of the 13th IFIP Conference, Tokyo, Japan, Aug./Sept. 1987*, volume 113 of *Lecture Notes in Control and Information Sciences*, pages 1–13. Springer Verlag, Berlin, Germany, 1988.

[25] Jorge J. Moré, Burton S. Garbow, and Kenneth E. Hillstrom. User Guide for MINPACK-1. *ANL-80-74, Argonne National Laboratory*, 1980.

[26] Jorge J. Moré and Danny C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computation*, 4:553–572, 1983.

[27] Jorge J. Moré and Gerardo Toraldo. On the solution of large quadratic programming problems with bound constraints. *SIAM Journal Optimization*, 1:93–113, 1991.

[28] Benedetta Morini and Margherita Porcelli. TRESNEI, a Matlab trust-region solver for systems of nonlinear equalities and inequalities. Technical report, Dipartimento di Energetica, Università di Firenze, Italy, 2009.

[29] Pu-Yan Nie. A derivative-free method for the system of nonlinear equations. *Nonlinear Analysis: Real World Applications*, 7:378–384, 2006.

[30] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, August 1999.

[31] Caroline Sainvitu and Philippe L. Toint. A filter-trust-region method for simple-bound constrained optimization. *Optimization Methods Software*, 22(5):835–848, 2007.

[32] Jonas Tölke, Manfred Krafczyk, and Ernst Rank. A multigrid-solver for the discrete Boltzmann equation. *Journal of Statistical Physics*, 107(1–2):573–591, 2002.

Appendix

The problems from our test set for which the algorithms in Section 5 fail are listed below. The problem names refer to the expensive reformulations of the corresponding problems from the CUTEr collection [14] presented in [20].

EFNES: `ARGLCLE`, `CHRSBNE`, `EIGENC`, `PFIT4`, `POWELLBS`, `POWELLSQ`

FILTRANE: `ARGLCLE`, `EIGENB`, `HIMMELBD`, `PFIT2`, `SEMICON1`

TRESNEI: `ARGLBLE`, `EIGENC`, `HATFLDF`, `POWELLBS`

`fmincon`: `ARGAUSS`, `ARGLBLE`, `ARGLCLE`, `BRATU2DT`, `BROYDNBD`, `CHEMRCTA`, `CHNRSBNE`, `EIGENA`, `EIGENB`, `EIGENB`, `GROWTH`, `HATFLDF`, `HATFLDG`, `HIMMELBD`, `MSQRTB`, `PFIT1`, `PFIT2`, `PFIT3`, `PFIT4`, `POWELLBS`, `POWELLSQ`, `SEMICON1`

`patternsearch`: `ARGAUSS`, `ARGLALE`, `ARGLBLE`, `ARGLCLE`, `BRATU2DT`, `BROWNALE`, `CHANDHEQ`, `CHEMRCTA`, `COOLHANS`, `DECONVNE`, `EIGENA`, `EIGENB`, `EIGENB`, `GOTTFR`, `GROWTH`, `HATFLDF`, `HATFLDG`, `HIMMELBD`, `PFIT2`, `PFIT3`, `PFIT4`, `POWELLBS`, `POWELLSQ`, `SEMICON1`, `SEMICON2`, `SINVALNE`