Markus Kaiser and Alexander Thekale

Solving nonlinear feasibility problems with expensive functions

January 2010

http://www.math.uni-wuppertal.de

Solving nonlinear feasibility problems with expensive functions

M. Kaiser, A. Thekale

January 11, 2010

Abstract

We present an algorithm for nonlinear feasibility problems, i.e. for systems of nonlinear equations and nonlinear inequalities, which depend on the outcome of expensive functions. 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 state a significant reduction in function evaluations compared to other derivative based and derivative-free solvers for nonlinear feasibility problems.

Keywords: feasibility problem, nonlinear system, equation, inequality, derivative-free, multidimensional filter, trust-region, global convergence

1 Introduction

In this paper we intend to solve the following general feasibility problem with expensive functions: Find a vector $x \in \mathbb{R}^n$ such that the *Expensive System* of nonlinear equations and inequalities

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

is satisfied, 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 rather costly in some sense and plays the major role in the solution cost 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), we use the following well known reformulation to a nonlinear least-squares problem. We try to find a local minimizer $x \in \mathbb{R}^n$ of the nonlinear unconstrained problem (see, e.g., [17]):

$$\min_x f(x, u(x)) = \frac{1}{2}\|\vartheta(x, u(x))\|_2^2 \tag{1}$$

where we define

$$\vartheta(x, u(x)) := \begin{pmatrix} c_{\mathcal{E}}(x, u(x)) \\ [c_{\mathcal{I}}(x, u(x))]_+ \end{pmatrix} \in \mathbb{R}^{p+q} \tag{2}$$

with $[c_{\mathcal{I}}(x, u(x))]_+ := \max[0, c_{\mathcal{I}}(x, u(x))]$ as the violation of the equations and inequalities, respectively.

Feasibility problems occur in many different applications, e.g. as discretized nonlinear partial differential equations [31] or in the restoration phase of filter methods for nonlinear optimization [10]. A large number of algorithms have been developed to solve this general type of problem or related types of problems as nonlinear systems of equations and nonlinear least-squares problems based on, e.g. Newton methods [24, 29], trust-region methods [7, 12, 17, 21] or evolutionary algorithms [19]. For a survey of algorithms for nonlinear systems of equations, see also [22]. Especially in engineering, these systems are often modeled using the outcome of an expensive function, e.g. a simulation. The evaluation of these expensive functions is often very time consuming and derivative information cannot be provided. Thus, derivative-free methods are sometimes applied for solving such systems, see, e.g. [28] for a pattern search method for solving nonlinear equation systems. The aim of this paper is to construct a globally convergent method for solving nonlinear feasibility problems. It is based on and extends the filter trust-region algorithm FILTRANE [17] to problems with expensive functions using derivative-free techniques for the expensive function, which can mainly be found in [4, 6]. Therefore, our method is a hybrid method using derivative-based techniques as well as derivative-free ones.
This paper is organized as follows: In Section 2 we describe the general framework of the filter-trust-region algorithm we propose for solving (ES). 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 give promising numerical results and compare our method to FILTRANE [17] and a pattern search method for nonlinear systems of equations [28]. Finally, concluding remarks are given in Section 6.

2 An algorithm for nonlinear feasibility problems with expensive functions

2.1 The general setting of the algorithm

The main idea of our algorithm is to solve (ES) by iteratively solving a sequence of *Cheap Systems* (CS_k) of nonlinear equations and inequalities in a trust-region framework where, in iteration k, the expensive function u is replaced by a cheap model function $m_k^u : \mathbb{R}^n \to \mathbb{R}^m$. We denote the iterates generated by this sequence with x_k and restrict the analysis to the case where, if necessary, m_k^u is a valid model of u in a neighborhood of x_k given by

$$\mathcal{Q}(x_k, \delta_k) := \{x \in \mathbb{R}^n : \|x - x_k\|_\infty \leq \delta_k\}$$

for some $\delta_k > 0$. In our situation, validity means that m_k^u satisfies the following assumption:

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

$$m_k^u(x_k) = u(x_k)$$

and the following error bounds hold for some constants $\kappa_{\mathrm{u}} > 0$, $\kappa_{\mathrm{gu}} > 0$:

$$\begin{aligned} \|u(x) - m_k^u(x)\|_2 &\leq \kappa_{\mathrm{u}}\delta_k^2 & (3)\\ \|\nabla_x u(x) - \nabla_x m_k^u(x)\|_2 &\leq \kappa_{\mathrm{gu}}\delta_k & (4)\end{aligned}$$

for all $x \in \mathcal{Q}(x_k, \delta_k)$, where $\|\cdot\|_2$ is the spectral norm in (4).

Note that the assumption $m_k^u(x_k) = u(x_k)$ could also be omitted but significantly facilitates the subsequent analysis. Similarly, (4) could alternatively be derived from (3), see [4] for more details. For an explicit derivation of a model satisfying Assumption 1 we refer to [5, 6]. In our method, the following two assumptions are necessary to guarantee that it is always possible to have access to a model satisfying Assumption 1 whenever we need it (see, e.g. [1, 4]):

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

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 k and any $\delta_k > 0$.

Having the local model at hand, problem (CS$_k$) can be formulated as follows:

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

The unconstrained problem (1) is then replaced by the box constrained problem

$$\begin{aligned} \min_x \quad f(x, m_k^u(x)) &= \tfrac{1}{2}\|\vartheta(x, m_k^u(x))\|_2^2\\ s.t. \qquad\qquad x &\in \mathcal{Q}(x_k, \delta_k).\end{aligned} \qquad (5)$$

In order to decide if a trial point x_k^+, i.e. a point that occurred during the attempt to solve (CS$_k$) and that seems to be probably useful for the progress of the iterative procedure, is suitable to be the next iterate x_{k+1}, we consider, beside the trust-region mechanism, the concept of a *multidimensional filter*. This is a variant of the well known filter method introduced in [11], adapted for feasibility problems. We will summarize the idea in the following 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 $\|\vartheta(x_l, u(x_l))\| := (|\vartheta_1(x_l, u(x_l))|, \ldots, |\vartheta_{p+q}(x_l, u(x_l))|)$ where $l \in \{1, \ldots, k\}$ such that no entry is dominated by another entry, i.e.

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

holds 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*. Based on F_u, we can now state the following acceptance criterion for the trial point x_k^+: x_k^+ is acceptable for the filter F_u if and only if for all $\|\vartheta(x_l, u(x_l))\| \in F_u$ holds:

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

where $\gamma_\theta \in (0, \frac{1}{\sqrt{p+q}})$. If x_k^+ is not only an acceptable trial point, but is also added to the filter by the method presented in this paper, every dominated filter entry must be removed. The use of this filter technique introduces an additional criterion to accept new iterates and therewith the availability of more potentially suitable iterates.

2.2 On the computation of the trial point

The problems (CS$_k$) are now standard nonlinear systems of equations and inequalities which can be solved cheaply compared to an evaluation of the expensive function, see Section 4. In comparison to standard trust-region methods we do not require for the *trial point* x_k^+ that

$$x_k^+ \in \mathcal{Q}(x_k, \delta_k) \tag{6}$$

holds in every iteration k, where x_k^+ is the outcome of some trial point generating subroutine. An example for such a subroutine will be discussed in Section 4. x_k^+ shall give a good candidate point that helps to find a solution of (ES). In many cases, x_k^+ is the solution of the nonlinear systems of equations and inequalities (CS$_k$), see Section 4. To state if (6) shall hold or not, we introduce the label RESTRICT which is set to $'true'$ if (6) is required and to $'false'$ if not, see also [12]. The advantage of this strategy is to allow trial points outside the current trust-region if the model m_k^u is assumed to be good in a bigger region than the current trust-region to hopefully converge faster to a solution of (ES). Note that this is an alternative strategy to the one used in FILTRANE [17], but this does not affect the theoretical properties of the method.
For the convergence analysis of our method which we present in this paper, we try to ensure that, as usual in trust-region methods, x_k^+ satisfies the following sufficient model reduction condition in every iteration k:

$$f(x_k, m_k^u(x_k)) - f(x_k^+, m_k^u(x_k^+)) \geq \kappa_{\mathrm{mdc}\delta} \|g_k\|_2 \min\left(\frac{\|g_k\|_2}{\beta_k}, \delta_k\right) \tag{7}$$

where $g_k := \nabla f(x_k, m_k^u(x_k))$, 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 $\kappa_{\mathrm{mdc}\delta}$ is a constant in $(0,1)$. Note that (7) is different from the standard sufficient model reduction condition since the right hand side depends on the norm of the gradient of the nonlinear cheap function $f(x, m_k^u(x))$ and not, as usual, of the gradient of the original expensive function $f(x, u(x))$.

2.3 The algorithm

Now we can state our algorithm to determine the solution of (1).

Algorithm 1 Multidimensional filter algorithm with expensive functions

Step 0: (Initialization) *An initial point x_0 and an initial trust-region radius $\delta_0 = \delta_{\text{ref}} > 0$ are given as well as the constants mentioned below. Compute $c_0 = c(x_0, u(x_0))$, ϑ_0 and an initial model m_0^u of u. Set $k = 0$, RESTRICT=false, and the initial filter F_u to the empty set.*

Step 1: (Optimality test) *STOP if $\vartheta(x_k, u(x_k)) = 0$ or $\|g_k\|_2 < \varepsilon_{\text{end}}$ for a valid model m_k^u in $\mathcal{Q}_k(x_k, \delta_{\text{end}})$ for some $\delta_{\text{end}} \in (0, \mu\|g_k\|_2]$.*

If m_k^u is not valid in $\mathcal{Q}_k(x_k, \mu\|g_k\|_2)$, perform as many improvement steps as necessary to ensure that the updated model is valid in $\mathcal{Q}_k(x_k, \alpha\mu\|g_k\|_2)$ and return to the beginning of Step 1.

Step 2: (Trial point determination) *Try to compute a trial point x_k^+ satisfying (7) and, if RESTRICT = true, also satisfying (6). If this is impossible, set $x_{k+1} = x_k$, RESTRICT = true, $\delta_{k+1} = \gamma_0\delta_k$, perform model improvement steps, define m_{k+1} as the improved model and go to Step 1.*

Step 3: (Evaluation of the residual at the trial point) *Compute $u(x_k^+)$, $c_{\mathcal{E}}(x_k^+, u(x_k^+))$ and $c_{\mathcal{I}}(x_k^+, u(x_k^+))$. Define*

$$\rho_k = \frac{f(x_k, u(x_k)) - f(x_k^+, u(x_k^+))}{f(x_k, m_k^u(x_k)) - f(x_k^+, m_k^u(x_k^+))}. \tag{8}$$

If $\rho_k \geq \eta_1$ then define $\mathcal{X}_k = \{x_k^+\}$, else $\mathcal{X}_k = \{x_k\}$.

Step 4: (Model improvement) *If $\rho_k < \eta_2$ and m_k^u is invalid in $\mathcal{Q}(x_k, \delta_k)$, perform model improvement steps, 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: (New trial point determination) *Determine $\hat{x}_k \in \mathcal{X}_k$ such that*

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

and set $\hat{\vartheta}_k := \vartheta(\hat{x}_k)$ and define

$$\hat{\rho}_k = \frac{f(x_k, u(x_k)) - f(\hat{x}_k, u(\hat{x}_k))}{f(x_k, m_k^u(x_k)) - f(x_k^+, m_k^u(x_k^+))}. \tag{9}$$

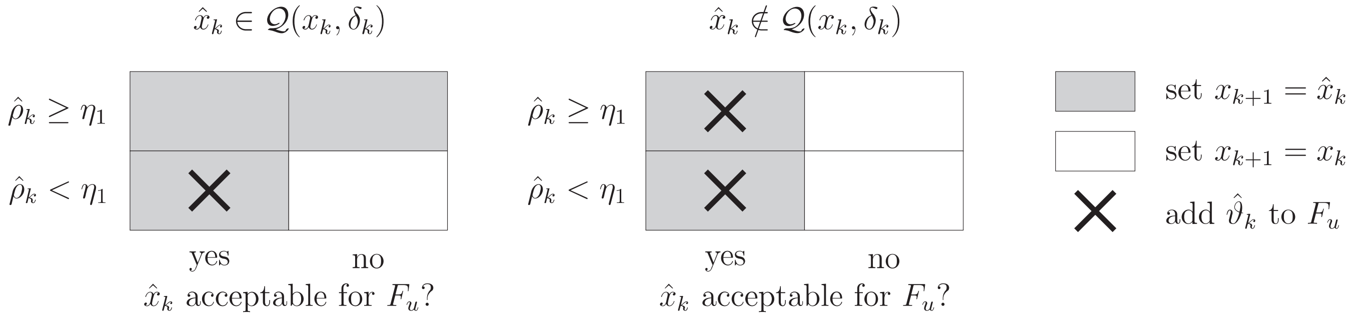
Step 6: (Acceptance test)

- *If (6) holds for $\hat{x}_k$ and $\hat{\rho}_k \geq \eta_1$:*
  set $x_{k+1} = \hat{x}_k$ and RESTRICT=false.
- *Elseif $\hat{x}_k$ is acceptable for the current filter F_u:*
  set $x_{k+1} = \hat{x}_k$, RESTRICT=false and add $\hat{\vartheta}_k$ to F_u.
- *Else:*
  set $x_{k+1} = x_k$ and RESTRICT=true.

Step 7: (Trust-region radius update) *If (6) holds, set $\delta_{\text{ref}} = \delta_k$ if $\hat{\rho}_k \geq \eta_1$ or if m_k^u is valid in $\mathcal{Q}(x_k, \delta_k)$ and update the trust-region radius by choosing*

$$\delta_{k+1} \in \begin{cases} [\gamma_0\delta_{\text{ref}}, \gamma_1\delta_{\text{ref}}] & \textit{if} \quad \hat{\rho}_k < \eta_1 \\ [\gamma_1\delta_{\text{ref}}, \delta_{\text{ref}}] & \textit{if} \quad \hat{\rho}_k \in [\eta_1, \eta_2) \\ [\delta_{\text{ref}}, \gamma_2\delta_{\text{ref}}] & \textit{if} \quad \hat{\rho}_k \geq \eta_2; \end{cases}$$

otherwise, set $\delta_{k+1} = \delta_k$. Increment k by one and go to Step 1.



$\hat{x}_k \in \mathcal{Q}(x_k, \delta_k)$
$\hat{x}_k \notin \mathcal{Q}(x_k, \delta_k)$
$\hat{\rho}_k \geq \eta_1$
$\hat{\rho}_k < \eta_1$
yes
no
$\hat{x}_k$ acceptable for F_u?
set $x_{k+1} = \hat{x}_k$
set $x_{k+1} = x_k$
add $\hat{\vartheta}_k$ to F_u

Figure 1: Illustration of the decision scheme (Step 6) in Algorithm 1

In the initialitation, we use the constants $0 < \gamma_0 \leq \gamma_1 < 1 \leq \gamma_2$, $\gamma_\vartheta \in \left(0, \frac{1}{\sqrt{p+q}}\right)$, $0 < \eta_1 < \eta_2 < 1$, $\mu > 0$, $\varepsilon_{\text{end}} > 0$ and $\alpha \in (0, 1)$. A reasonable choice for these parameters is $\gamma_0 = 0.1$, $\gamma_1 = 0.25$, $\gamma_2 = 7.5$, $\gamma_\vartheta = 10^{-4}$, $\eta_1 = 0.2$, $\eta_2 = 0.9$, $\mu = 0.5$, $\varepsilon_{\text{end}} = 10^{-6}$ and $\alpha = 0.9$.
Some further comments on Algorithm 1 are necessary. The criticality test in Step 1 causes 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.
In Step 2, we try to compute a trial point x_k^+ satisfying (7) with some appropriate subroutine. An example if 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 (7) in iteration k (this may happen, for example, if m_k^u represents u very badly), we call iteration k a *failure iteration.* The set of all failure iterations is denoted by $\mathcal{F}$.
The acceptance test in Step 6 is stated differently from the one in FILTRANE [17], as it gives, in our opinion, a better insight in this step, see also Figure 1.
Using δ_{ref} in the trust-region update (Step 7) ensures that the trust-region radius does not reduce too fast if the model is invalid. Otherwise, the algorithm could be trapped in a small trust-region at the time when the model is valid again, and relatively many iterations would be needed to proceed. See also the comments after Algorithm 9.1.1 in [4].

3 Convergence analysis

We now investigate the convergence properties of Algorithm 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)*
$u(x)$ is a twice continuously differentiable function in x, $c_\mathcal{E}(x, u(x))$ and $c_\mathcal{I}(x, u(x))$ are twice continuously differentiable functions in x and $u(x)$. The functions $u(x)$, $m_k^u(x)$, $c_\mathcal{E}(x, u(x))$,

$c_{\mathcal{I}}(x, u(x))$,

$$\nabla c_{\mathcal{E}}(x, u(x)) := \nabla_x c_{\mathcal{E}}(x, u(x)) + \nabla_u c_{\mathcal{E}}(x, u(x)) \nabla_x u(x)$$

and

$$\nabla c_{\mathcal{I}}(x, u(x)) := \nabla_x c_{\mathcal{I}}(x, u(x)) + \nabla_u c_{\mathcal{I}}(x, u(x)) \nabla_x u(x)$$

are Lipschitz continuous with Lipschitz constant $\gamma_{\mathrm{l}} > 0$ and bounded from above by a constant $\gamma_{\mathrm{b}} > 0$.

Assumption 5 *(Iterates)*
All points that are evaluated remain in a bounded domain $\Omega \subset \mathbb{R}^n$.

Assumption 6 *(Model continuity)*
The model m_k^u is twice continuously differentiable in x for all k.

Assumption 7 *(Method determining the trial point)*
For all $\varepsilon > 0$ there exists an $\varepsilon_{\mathrm{mdc}} > 0$ such that, if $\|g_k\|_2 > \varepsilon$ and $\delta_k \leq \varepsilon_{\mathrm{mdc}}$ hold, the trial point x_k^+ determined in Step 2 of Algorithm 1 satisfies (7), i.e. $k \notin \mathcal{F}$.

Note that Assumptions 2 and 3 imply that the validity test in Steps 1, 2 and 4 of Algorithm 1 is always possible and that, whenever a model improvement is necessary, no infinite loop is produced by the algorithm.
Also note that Assumptions 4 and 5 directly imply that we can assume the Hessian of $f(x, u(x))$ to be bounded from above in the convex hull of the iterates $\{x_k\}$, i.e. there exists a constant $\kappa_{\mathrm{ufh}} \geq 1$ such that

$$1 + \|\nabla^2 f(x, u(x))\|_2 \leq \kappa_{\mathrm{ufh}}.$$

Assumption 7 is not common for trust-region methods and its satisfaction is not obvious. How to satisfy this assumption will be discussed in Section 4.
In the first step of our convergence analysis, we consider the case where infinitely many values $\vartheta_k := \vartheta(x_k, u(x_k))$ are added to the filter F_u. Here, we can directly adopt the result of Theorem 3.1 in [12], with adapted notation, as it is independent of the definition of ϑ:

Lemma 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 1. Then*

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

We now proceed our analysis analogue to the convergence theory of the derivative-free trust-region method presented in Chapter 9.2 in [4] but with some modifications in the proofs needed because of the composite structure of (1) and the usage of the label *RESTRICT*. First we investigate the error between the expensive objective function in (1) and the objective in the cheap model problem (5) as well as between the corresponding gradients at a given point $x \in \mathcal{Q}(x_k, \delta_k)$.

Lemma 2 *Suppose that Assumptions 1-6 hold. Then*

$$\begin{aligned} |f(x,u(x)) - f(x,m_k^u(x))| &\leq \kappa_{\text{ubh}}\delta_k^2 & (10)\\ \|\nabla f(x,u(x)) - \nabla f(x,m_k^u(x))\|_2 &\leq \kappa_{\text{ubh}}\max[\delta_k,\delta_k^2] & (11)\end{aligned}$$

hold for all $x \in \mathcal{Q}(x_k,\delta_k)$ *where*

$$\kappa_{\text{ubh}} := \max[2\gamma_{\text{b}}\gamma_{\text{l}}\kappa_{\text{u}}, (p+q)\gamma_{\text{b}}\left(2\gamma_{\text{l}}\kappa_{\text{u}} + \gamma_{\text{b}}\kappa_{\text{gu}} + \gamma_{\text{b}}\gamma_{\text{l}}\kappa_{\text{u}}\right), \kappa_{\text{ufh}}].$$

Proof: We first prove (10). Using sequently the definition of f and ϑ, the binomial theorem, Assumption 4 and the triangle inequality, we obtain

$$\begin{aligned} |f(x,u(x)) - f(x,m_k^u(x))| &= \frac{1}{2}\Big|\|\vartheta(x,u(x))\|_2^2 - \|\vartheta(x,m_k^u(x))\|_2^2\Big|\\ &= \frac{1}{2}\Big|\|c_{\mathcal{E}}(x,u(x))\|_2^2 - \|c_{\mathcal{E}}(x,m_k^u(x))\|_2^2\\ &\quad + \|[c_{\mathcal{I}}(x,u(x))]_+\|_2^2 - \|[c_{\mathcal{I}}(x,m_k^u(x))]_+\|_2^2\Big|\\ &\leq \frac{1}{2}\cdot 2\gamma_{\text{b}}\Big|\|c_{\mathcal{E}}(x,u(x)) - c_{\mathcal{E}}(x,m_k^u(x))\|_2\\ &\quad + \|[c_{\mathcal{I}}(x,u(x))]_+ - [c_{\mathcal{I}}(x,m_k^u(x))]_+\|_2\Big|\end{aligned}$$

Then, case differentiation for the second term and Assumptions 4 and 1 yield directly

$$\begin{aligned} |f(x,u(x)) - f(x,m_k^u(x))| &\leq 2\gamma_{\text{b}}\gamma_{\text{l}}\|u(x) - m_k^u(x)\|_2\\ &\leq 2\gamma_{\text{b}}\gamma_{\text{l}}\kappa_{\text{gu}}\delta_k^2.\end{aligned}$$

To prove (11), we first state that

$$\nabla f(x,u(x)) = \sum_{i=1}^{p} c_{\mathcal{E}i}(x,u(x))\nabla c_{\mathcal{E}i}(x,u(x)) + \sum_{i=1}^{q}[c_{\mathcal{I}i}(x,u(x))]_+\nabla c_{\mathcal{I}i}(x,u(x)).$$

This yields

$$\begin{aligned} &\|\nabla f(x,u(x)) - \nabla f(x,m_k^u(x))\|_2\\ &\quad\leq \sum_{i=1}^{p}\|c_{\mathcal{E}i}(x,u(x))\nabla c_{\mathcal{E}i}(x,u(x)) - c_{\mathcal{E}i}(x,m_k^u(x))\nabla c_{\mathcal{E}i}(x,m_k^u(x))\|_2\\ &\quad\quad + \sum_{i=1}^{q}\|[c_{\mathcal{I}i}(x,u(x))]_+\nabla c_{\mathcal{I}i}(x,u(x)) - [c_{\mathcal{I}i}(x,m_k^u(x))]_+\nabla c_{\mathcal{I}i}(x,m_k^u(x))\|_2\\ &\quad\leq \sum_{i=1}^{p}[|c_{\mathcal{E}i}(x,u(x))|\|\nabla c_{\mathcal{E}i}(x,u(x)) - \nabla c_{\mathcal{E}i}(x,m_k^u(x))\|_2\end{aligned}$$

$$
\begin{aligned}
&+|c_{\mathcal{E}i}(x,u(x)) - c_{\mathcal{E}i}(x,m_k^u(x))| \|\nabla c_{\mathcal{E}i}(x,m_k^u(x))\|_2] \\
&+\sum_{i=1}^{q} [|[c_{\mathcal{I}i}(x,u(x))]_+| \|\nabla c_{\mathcal{I}i}(x,u(x)) - \nabla c_{\mathcal{I}i}(x,m_k^u(x))\|_2 \\
&+|[c_{\mathcal{I}i}(x,u(x))]_+ - [c_{\mathcal{I}i}(x,m_k^u(x))]_+| \|\nabla c_{\mathcal{I}i}(x,m_k^u(x))\|_2] \,.
\end{aligned}
$$

Using the definition of $\nabla c_{\mathcal{E}}(x,u(x))$ and $\nabla c_{\mathcal{I}}(x,u(x))$ and the constants in Assumption 4 as well as the triangle inequality, we obtain

$$
\begin{aligned}
&\|\nabla f(x,u(x)) - \nabla f(x,m_k^u(x))\|_2 \\
\le\;& \gamma_{\mathrm{b}} \sum_{i=1}^{p} [\|\nabla_x c_{\mathcal{E}i}(x,u(x)) - \nabla_x c_{\mathcal{E}i}(x,m_k^u(x))\|_2 \\
&+\|\nabla_u c_{\mathcal{E}i}(x,u(x))\nabla_x u(x) - \nabla_u c_{\mathcal{E}i}(x,m_k^u(x))\nabla_x m_k^u(x)\|_2 + \gamma_{\mathrm{l}}\|u(x) - m_k^u(x)\|_2] \\
&+\gamma_{\mathrm{b}} \sum_{i=1}^{q} [\|\nabla_x c_{\mathcal{I}i}(x,u(x)) - \nabla_x c_{\mathcal{I}i}(x,m_k^u(x))\|_2 \\
&+\|\nabla_u c_{\mathcal{I}i}(x,u(x))\nabla_x u(x) - \nabla_u c_{\mathcal{I}i}(x,m_k^u(x))\nabla_x m_k^u(x)\|_2 + \gamma_{\mathrm{l}}\|u(x) - m_k^u(x)\|_2] \\
\le\;& \gamma_{\mathrm{b}} \sum_{i=1}^{p} [2\gamma_{\mathrm{l}}\|u(x) - m_k^u(x)\|_2 + \|\nabla_u c_{\mathcal{E}i}(x,u(x))\nabla_x u(x) - \nabla_u c_{\mathcal{E}i}(x,u(x))\nabla_x m_k^u(x)\|_2 \\
&+\|\nabla_u c_{\mathcal{E}i}(x,u(x))\nabla_x m_k^u(x) - \nabla_u c_{\mathcal{E}i}(x,m_k^u(x))\nabla_x m_k^u(x)\|_2] \\
&\gamma_{\mathrm{b}} \sum_{i=1}^{q} [2\gamma_{\mathrm{l}}\|u(x) - m_k^u(x)\|_2 + \|\nabla_u c_{\mathcal{I}i}(x,u(x))\nabla_x u(x) - \nabla_u c_{\mathcal{I}i}(x,u(x))\nabla_x m_k^u(x)\|_2 \\
&+\|\nabla_u c_{\mathcal{I}i}(x,u(x))\nabla_x m_k^u(x) - \nabla_u c_{\mathcal{I}i}(x,m_k^u(x))\nabla_x m_k^u(x)\|_2] \\
\le\;& \gamma_{\mathrm{b}} \sum_{i=1}^{p} \left[2\gamma_{\mathrm{l}}\kappa_{\mathrm{u}}\delta_k^2 + \gamma_{\mathrm{b}}\|\nabla_x u(x) - \nabla_x m_k^u(x)\|_2 + \gamma_{\mathrm{b}}\|\nabla_u c_{\mathcal{E}i}(x,u(x)) - \nabla_u c_{\mathcal{E}i}(x,m_k^u(x))\|_2\right] \\
&+\gamma_{\mathrm{b}} \sum_{i=1}^{q} \left[2\gamma_{\mathrm{l}}\kappa_{\mathrm{u}}\delta_k^2 + \gamma_{\mathrm{b}}\|\nabla_x u(x) - \nabla_x m_k^u(x)\|_2 + \gamma_{\mathrm{b}}\|\nabla_u c_{\mathcal{I}i}(x,u(x)) - \nabla_u c_{\mathcal{I}i}(x,m_k^u(x))\|_2\right] \\
\le\;& (p+q)\gamma_{\mathrm{b}}\left(2\gamma_{\mathrm{l}}\kappa_{\mathrm{u}}\delta_k^2 + \gamma_{\mathrm{b}}\kappa_{\mathrm{gu}}\delta_k + \gamma_{\mathrm{b}}\gamma_{\mathrm{l}}\|u(x) - m_k^u(x)\|_2\right) \\
\le\;& (p+q)\gamma_{\mathrm{b}}\left(2\gamma_{\mathrm{l}}\kappa_{\mathrm{u}} + \gamma_{\mathrm{b}}\kappa_{\mathrm{gu}} + \gamma_{\mathrm{b}}\gamma_{\mathrm{l}}\kappa_{\mathrm{u}}\right)\max[\delta_k, \delta_k^2].
\end{aligned}
$$

□

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 can get stuck in a non-critical iterate.

Lemma 3 (Theorem 9.2.1 in [4]) *Suppose that Assumptions 1-7 hold. Suppose furthermore that there exists an $\varepsilon > 0$ such that $\|g_k\|_2 > \varepsilon$ for all k. Then*

$$\delta_k \ge \gamma_1^2 \min\left[\frac{\kappa_{\mathrm{mdc}\delta}\varepsilon(1-\eta_2)}{\kappa_{\mathrm{ubh}}}, \varepsilon_{\mathrm{mdc}}\right] =: \kappa_{\mathrm{lbd}}$$

for all k.

Proof: As $\|g_k\|_2 > \varepsilon$ and $\delta_k \leq \varepsilon_{\text{mdc}}$, Assumption 7 guarantees that x_k^+ exists and satisfies (7). Now, the desired result follows analogue to the proof of Theorem 9.2.1 in [4] with adapted notation. □

Now we can directly state the final convergence result from Section 9.2 in [4]. But due to the usage of the label RESTRICT, we have to ensure, in addition,

$$\|x_k^+ - x_k\|_2 \leq \kappa_\delta \delta_k \qquad \text{for all} \qquad k \geq k_0 \tag{12}$$

for some $k_0 > 0$ and some large constant $\kappa_\delta \geq 1$, see the discussion for the convergence analysis presented in Section 3 in [12] for a detailed explanation of this necessity. This condition ensures the global convergence of the algorithm as (12) plays the role of a trust-region bound in a larger trust-region with radius $\kappa_\delta \delta_k$. We thus have

Lemma 4 (Theorem 9.2.6 in [4]) *Suppose that Assumptions 1-7 as well as (12) 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 1 yields directly our final convergence result:

Theorem 1 *Suppose that Assumptions 1-7 as well as (12) 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 values 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 how to determine a trial point x_k^+ in Step 2 of Algorithm 1 that satisfies Assumption 7. In standard trust-region methods, the model that substitutes the true objective function is usually a rather simple, in many cases 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 new trial point that automatically satisfies a sufficient model decrease condition analogous to (7), see, e.g. Chapter 6.3 in [4]. For *trust-region subproblem* solvers for quadratic models see, e.g. [4, 9, 13, 25].
In our case, the trust-region subproblem (CS$_k$) is 'easy' compared to the original problem (ES) in the sense that function evaluations are fast and derivative information is accessible and therefore also the determination of the trial point is rather fast. Modeling the subproblem as general nonlinear feasibility problem (CS$_k$) as shown in Section 2 maintains all the information that is available and thus solves the problem with less expensive function evaluations, which is practically shown in Section 5. But since the problems (CS$_k$) are still arbitrarily nonlinear, (7) cannot be guaranteed in general. Even the *model minimizer*, the global solution of (5), does not 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 $\mathcal{Q}$. Thus, we try to generate

this model minimizer and apply a method in Step 2 of Algorithm 1 that shall solve (5). But due to the problems in guaranteeing (7), we have to add an additional criterion to this method. This criterion will be described in the following and is strongly connected to the solver we use for the trust-region subproblem. Depending on a the particular problem, different subproblem solvers might be reasonable, but to use again a convergent trust-region method for this solver bears a major advantage: The trial points generated within this method satisfy a sufficient model decrease condition in the model used inside this method. In the following, we will call this inner model q as it will usually be a quadratic model in practice. Our main idea now is to link this internal sufficient model decrease with the model decrease we need in Step 2 of Algorithm 1. To be able to describe this, we need some further notation. We denote the iterates generated within our trust-region subproblem solver by y_{kp} and the trial points with y_{kp}^+ with double indices kp, where k denotes the iteration in Algorithm 1 and p the current iteration within the subproblem solver. The according trust-region radii are called Δ_{kp} and the corresponding trust-region is given by

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

Now we require a sufficient model decrease in iteration kp for the trial point y_{kp}^+ in the subproblem solver in the corresponding model q_{kp}, i.e.

$$q_{kp}(y_{kp}) - q_{kp}(y_{kp}^+) \geq \kappa_{\text{mdc}} \pi(y_{kp}) \min\left(\frac{\pi(y_{kp})}{\beta_{kp}}, \Delta_{kp}\right) \tag{13}$$

where β_{kp} is an upper bound on the norm of the Hessian of q_{kp} in $\mathcal{B}(y_{kp}, \Delta_{kp})$, κ_{mdc} is a constant in $(0, 1)$ and $\pi(y_{kp})$ is some criticality measure at y_{kp}. For criticality measures see, e.g. Chapter 8 in [4]. Our criterion which decides if the outcome $\bar{y}$ of the subproblem solver is accepted as trial point x_k^+ is then as follows:

Acceptance test 1 *We accept $\bar{y}$ as new trial point x_k^+ if and only if*

$$\frac{f(x_k, m_k^u(x_k)) - f(\bar{y}, m_k^u(\bar{y}))}{q_{k0}(y_{k0}) - q_{k0}(y_{k0}^+)} \geq \kappa_{\delta\Delta} \tag{14}$$

for some $\kappa_{\delta\Delta} \in (0, \eta_2]$. Otherwise, we denote k as failure iteration, e.g. $k \in \mathcal{F}$.

Thus, (14) guarantees that the model reduction in the cheap nonlinear objective function of (5) at the trial point is at least a fraction of the predicted model decrease in the first iteration of the subproblem solver. To guarantee the desired sufficient model decrease (7), we need the following assumptions on the initialization of the subproblem solver.

Assumption 8 *(Initialization of the subproblem solver)*

- *The subproblem solver is initialized with trust-region radius $\Delta_{k0} = \delta_k$ and starting point $y_{k0} = x_k$.*
- *β_k is sufficiently large such that it is also an upper bound of the Hessian of q_{k0}, e.g. $\beta_{k0} \leq \beta_k$.*

- *For the criticality measure π holds*

$$\pi(y_{k0}) \geq \kappa_\pi \|g_k\|_2 \tag{15}$$

for all k and a constant $\kappa_\pi > 0$ independent of k.

These assumptions form a reasonable initialization of the subproblem solver and can be guaranteed very easily since usually convergence theory of trust-region methods is independent of the choice of the initial starting point and the initial trust-region radius. The requirement that (15) can be satisfied, for example, if the criticality measure is defined gradient dependent. The subproblem solver we state below satisfies this assumption.
Note that, in the context of multiscale methods, (15) is similar to a criterion that decides if it is reasonable to switch to a coarser level or not, see, e.g., [18].
Now, the following corollary shows directly the desired sufficient model decrease (7) for any point passing Acceptance test 1.

Corollary 1 *Suppose that the subproblem solver satisfies (13) in iteration kp and that Assumption 8 holds. Then, every point $\bar{y}$ that passes Acceptance test 1 satisfies (7).*

Proof: Applying sequently (14), (13) and Assumption 8 leads to

$$\begin{aligned} f(x_k, m_k^u(x_k)) - f(\bar{y}, m_k^u(\bar{y})) &\geq \kappa_{\delta\Delta}(q_{k0}(y_{k0}) - q_{k0}(y_{k0}^+)) \\ &\geq \kappa_{\delta\Delta}\kappa_{\text{mdc}}\pi(y_{k0}) \min\left(\frac{\pi(y_{k0})}{\beta_{k0}}, \Delta_{k0}\right) \\ &\geq \kappa_{\text{mdc}\delta}\|g_k\|_2 \min\left(\frac{\|g_k\|_2}{\beta_k}, \delta_k\right) \end{aligned}$$

with $\kappa_{\text{mdc}\delta} := \kappa_\pi^2 \kappa_{\delta\Delta}\kappa_{\text{mdc}}$. □

Now we can formulate our algorithm for the determination of the trial point x_k^+ in iteration k of Algorithm 1:

Algorithm 2 Trying to determine the trial point in Step 2 of Algorithm 1
Step 1: (Solving the cheap problem) *Solve Problem (5) using a trust-region method satisfying (13) in every iteration and Assumption 8. Let $\mathcal{Y} := \{y_{kp} \mid y_{kp} \neq y_{kp-1}\}$ be the set of non-identic iterates produced by this trust-region method. Save $q_{k0}(y_{k0})$ and $q_{k0}(y_{k0}^+)$.*
Step 2: (Trial point) *Choose $\bar{y} \in \mathcal{Y}$ such that Acceptance test 1 is satisfied. Denote $x_k^+ := \bar{y}$ as trial point. If the acceptance test fails for all $y_{kp} \in \mathcal{Y}$, set iteration k as failure iteration, i.e. $k \in \mathcal{F}$.*

If there are several points in $\mathcal{Y}$ satisfying Acceptance test 1, the theory does not specify here which one to choose. But in practice, a point close to the solution of the cheap problem (5) is preferred as it represents the model minimizer. To complete our discussion on how to determine the trial point, we finally have to assure that the trust-region method applied in Step 1 of Algorithm 2 generates at least one $\bar{y} \in \mathcal{Y}$ that passes Acceptance test 1 if we are in the situation of Assumption

7, i.e. the gradient g_k is still large and the trust-region radius δ_k is sufficiently small. In our case, this will follow directly from the convergence analysis of the trust-region method we use in Step 1 of Algorithm 2.

To solve the nonlinear box-constrained problems (5), we apply again a filter trust-region method as presented in [30], which is based on [16] for unconstrained problems. This method is also based on the multidimensional filter which we have described in Section 2.1 and solves the general nonlinear box-constrained problem

$$\begin{array}{cl} \min\limits_{y} & f(y) \\ s.t. & l \leq y \leq u, \end{array} \tag{16}$$

where $l = (l_1, \ldots, l_n)^\top \in \mathbb{R}^n \cup \{-\infty\}$ and $u = (u_1, \ldots, u_n)^\top \in \mathbb{R}^n \cup \{+\infty\}$ are lower and upper bounds on the variable y, respectively. Without loss of generality, we assume $l_i < u_i$ for $i = 1, \ldots, n$.
In the context of Algorithm 1, we have to distinguish between two cases. If $RESTRICT$ is $'true'$ in iteration k of Algorithm 1, the box constraints are given by $\mathcal{Q}(x_k, \delta_k)$ and we have

$$l_i = (x_k)_i - \delta_k \qquad \text{and} \qquad u_i = (x_k)_i + \delta_k$$

where $(x_k)_i$ denotes the i-th component of x_k for $i = 1, \ldots, n$. If $RESTRICT$ is $'false'$, Problem (5) is unconstrained and we set

$$l_i = -\infty \qquad \text{and} \qquad u_i = +\infty$$

for $i = 1, \ldots, n$. In the unconstrained case, the direct application of, e.g. FILTRANE, see [17], is also possible. This does not change the discussion below, but, for the sake of simplicity, we describe the use of the box-constrained method [30] for both cases here.
We first check if the method we have chosen satisfies the assumptions of Corollary 1. Important for this is the definition of the criticality measure π used in [30]:

$$\pi(y) := \|\bar{g}(y)\|_\infty,$$

where $\bar{g}(y)$ is the projected gradient (see e.g. [2, 3, 20, 23, 26]) of the objective function $f(y, m_k^u(y))$ into the feasible box of (16). $\bar{g}(y)$ is given by

$$\bar{g}(y) := y - P[y - \nabla f(y, m_k^u(y)), l, u],$$

where the projection operator $P[y, l, u]$ is defined componentwise by

$$P[y, l, u]_i = \begin{cases} l_i & \text{if} \quad y_i \leq l_i \\ y_i & \text{if} \quad l_i \leq y_i \leq u_i \\ u_i & \text{if} \quad u_i \leq y_i, \end{cases}$$

see [30]. Note that this method uses the projection operator such that all iterates y_{kp} stay feasible, i.e. $y_{kp} \in \mathcal{Q}(x_k, \delta_k)$ in our situation. If we initialize our subproblem solver with $\Delta_{k0} = \delta_k$ and $y_{k0} = x_k$ as required in Assumption 8, we get

$$\pi(y_{k0}) = \|\nabla f(y_{k0}, m_k^u(y_{k0}))\|_\infty = \|g_k\|_\infty.$$

As $\|g_k\|_\infty \geq n^{-1/2}\|g_k\|_2$, Assumption 8 is satisfied with $\kappa_\pi := n^{-1/2}$. Note that κ_π is therefore problem dependent. (13) is also assumed in the convergence analysis in [30] and can be guaranteed as described above. Therefore, the assumptions of Corollary 1 hold.
To finally show that also Assumption 7 holds, i.e. the subproblem solver produces an iterate in $\mathcal{Y}$ that passes Acceptance test 1 if $\|g_k\|_2$ is large and δ_k is sufficiently small, we use the following lemma from the convergence theory of the subproblem solver presented in [30]:

Lemma 5 (Lemma 3.3 in [30]) *Suppose that Assumptions 4 and 5 hold, there exists a constant $\kappa_{\text{umh}} > 0$ such that*

$$|f(y_{kp}^+, m_k^u(y_{kp}^+)) - q_{kp}(y_{kp}^+)| \leq \kappa_{\text{umh}} \Delta_{kp}^2 \tag{17}$$

and that $\|y_{kp} - y_{kp}^+\|_\infty \leq \Delta_{kp}$. Suppose furthermore that $\|\bar{g}(y_{kp})\|_\infty \neq 0$ and that

$$\Delta_{kp} \leq \frac{\kappa_{\text{mdc}} \pi(y_{kp})(1-\eta_2)}{\kappa_{\text{umh}}}. \tag{18}$$

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

The existence of κ_{umh} satisfying (17) follows under standard trust-region assumptions, see the Assumptions A1-A3 and Lemma 3.2 in [30]. This is the analogous statement for the subproblem solver like Lemma 2 in our framework. The following theorem guarantees now that Algorithm 2 satisfies Assumption 7 if we use the described subproblem solver:

Theorem 2 *Suppose that Assumption 4, 5 and 8 hold. Suppose that we apply the algorithm in [30] as subproblem solver in Algorithm 2 and that (17) and (13) are satisfied in iteration kp. Suppose furthermore that $\|g_k\|_2 > \varepsilon$ and*

$$\delta_k \leq \frac{\kappa_{\text{mdc}} \varepsilon (1-\eta_2)}{\sqrt{n}\kappa_{\text{umh}}} =: \varepsilon_{\text{mdc}}.$$

Then Assumption 7 is satisfied for iteration k.

Proof: By the assumptions, the assumptions of Lemma 5 are satisfied in iteration $k0$ as

$$\|\bar{g}_{k0}\|_\infty = \|g_{k0}\|_\infty \geq n^{-1/2}\|g_{k0}\|_2 \geq n^{-1/2}\varepsilon > 0$$

and

$$\Delta_{k0} = \delta_k \leq \frac{\kappa_{\text{mdc}} \varepsilon (1-\eta_2)}{\sqrt{n}\kappa_{\text{umh}}} \leq \frac{\kappa_{\text{mdc}} \|g_k\|_2 (1-\eta_2)}{\sqrt{n}\kappa_{\text{umh}}} \leq \frac{\kappa_{\text{mdc}} \|g_k\|_\infty (1-\eta_2)}{\kappa_{\text{umh}}} \leq \frac{\kappa_{\text{mdc}} \pi(y_{k0})(1-\eta_2)}{\kappa_{\text{umh}}}.$$

Therefore, iteration $k0$ is very successful, we accept y_{k0}^+ setting $y_{k1} := y_{k0}^+ \in \mathcal{Y}$, and y_{k1} passes Acceptance test 1, as $\rho_{k0} \geq \eta_2$. Thus, Algorithm 2 produces a trial point x_k^+. As our assumptions satisfy in addition the assumptions of Corollary 1, x_k^+ also satisfies (7). □

5 Numerical results

5.1 Set-up and test properties

In this section, we test the practical behavior of Algorithm 1. We have implemented our algorithm in Matlab and call this implementation EFNES[1] in the following. For a comparison to existing algorithms, we apply EFNES to a set of 54 test problems, which were originally a subset of the CUTEr collection [14], but had to be modified to simulate the existence of one or more expensive functions. Therefore, we define some parts of the nonlinear systems to be expensive. We made these definitions in various and quite arbitrary ways.

Example	Var.	Eq.	Ineq.	Ex.func.	Example	Var.	Eq.	Ineq.	Ex.func.
Aircrfta	5	5	0	4-P-O	Eigena*	110	55	55	55-P-O
Argauss	3	15	0	15-E-A	Eigenb*	6	3	3	3-P-O
Arglale*	10	10	10	20-P-C	Eigenc	30	30	0	1-P-O
Arglble*	50	1	99	100-P-C	Gottfr	2	2	0	2-P-E
Arglcle	10	19	0	18-P-C	Growth*	3	1	11	2-L-O
Argtrig*	10	10	0	10-T-A	Hatfldf	3	3	0	3-EP-E
Artif*	10	0	10	10-T-A	Hatfldg	25	25	0	10-P-E
Arwhdne*	10	9	9	9-P-E	Himmelba	2	2	0	2-P-C
Bdvalue	10	10	0	6-P-O	Himmelbc	2	2	0	2-P-O
Bdvalues	10	10	0	3-P-O	Himmelbd	2	2	0	2-P-O
Booth	2	2	0	2-P-C	Himmelbe	3	3	0	2-P-C
Bratu2d	25	25	0	25-E-A	Hypcir	2	2	0	2-P-C
Bratu2dt	25	25	0	25-E-A	Integreq	10	10	0	3-P-O
Bratu3d	27	27	0	27-E-A	Msqrta	4	4	0	4-P-O
Brownale	10	10	0	1-P-E	Msqrtb	9	9	0	9-P-O
Broydn3d	10	10	0	8-P-O	Pfit1*	3	2	1	1-X-O
Broydnbd	10	10	0	3-P-O	Pfit2	3	3	0	1-X-O
Cbratu2d	8	8	0	4-ET-O	Pfit3	3	3	0	1-P-O
Cbratu3d	16	16	0	8-ET-O	Pfit4	3	3	0	1-X-O
Chandheq	10	10	0	2-P-E	Powellbs	2	2	0	2-PX-C
Chemrcta	10	10	0	4-P-E	Powellsq	2	2	0	2-PH-C
Chemrctb*	10	6	4	8-E-O	Recipe	3	3	0	3-PH-C
Chnrsbne	10	18	0	5-P-E	Rsnbrne	2	2	0	2-P-C
Cluster	2	2	0	2-PT-E	Semicon1	10	10	0	9-E-O
Coolhans	9	9	0	5-P-E	Semicon2	10	10	0	9-E-O
Cubene	2	2	0	2-P-C	Sinvalne	2	2	0	2-PT-C
Deconvne*	61	10	30	5-P-O	Zangwil3	3	3	0	3-P-C

Table 1: Characterization of the test problems

[1] solver for Expensive Function based Nonlinear Equation Systems

In addition, none of the available test problems in the CUTEr collection that has a suitable problem size contains inequalities. Thus, we redefine some equations as inequalities to obtain a well-balanced and significant test set for problems of the form (ES).
In Table 1, we specify the test problems which we have taken over from the CUTEr collection and the corresponding modifications. We label the problems where we have replaced at least one equation by an inequality by *. The table also contains information about the specific choice for the expensive function(s) we use in the corresponding problems.
To denote the modifications we made due to the artificially introduction of expensive function(s) on the test problems, we use a classification with three positions $\cdot - \cdot - \cdot$.
The first position contains the number of expensive functions included in the particular problem. The next position specifies the type of the expensive function. The expensive function can be polynomial (P), exponential (E), trigonometrical (T), logarithmical (L), hyperbolical (H) or of the type x^x (X). If more than one of these types is involved, all of them are mentioned.
The last position yields information if the expensive function satisfies one of the following conditions: The complete system of nonlinear equations and inequalities is considered as expensive (C), entire equations or inequalities of the system are considered as expensive function (E), or all of the equations and inequalities of the system contain at least one part that is defined as expensive function (A). If none of these conditions is satisfied, i.e, the expensive functions appear only in some of the equations or inequalities, the letter (O) is added.

5.2 Using available information

The nested function formulation of the feasibility problems (ES) yields a lot of freedom and gives the possibility to model available information in the outer functions $c_{\mathcal{I}}$ and $c_{\mathcal{E}}$. Such information is given in practical applications, for example, in the post-processing process of numerical simulations. In general, incorporating as much information as possible in the outer functions leads to better results, i.e. fewer expensive function evaluations are necessary. The reason is that the more information is modeled in the outer functions instead of inside the expensive function, the more accurate is, in average, the local model (CS$_k$). This modeling approach is, together with the use of the local models for the expensive function, the main reason for the effectiveness of EFNES.
The following example shows the effect of modeling available information. Let us consider the example *Argauss*, see Table 1, which is the following nonlinear system of equations (without any expensive function):

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

for $i = 1, \ldots, 15$, where $\alpha_i = 4 - \frac{i}{2}$ and $\beta_i \in \mathbb{R}$ are small constants. For this system, we artificially established 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 without the time for the evaluation of the expensive function are given in Table 2. This illustrates the benefit that arises from taking as much information into account as possible.

	$e_i = (\alpha - x(3))^2$	$e_i = \frac{x_2}{2}(\alpha_i - x_3)^2$	$e_i = e^{\frac{x_2}{2}(\alpha_i - x_3)^2}$	$e_i = x_1 e^{\frac{x_2}{2}(\alpha_i - x_3)^2} - \beta_i$
CPU time	0.6362	1.7265	2.1693	3.3095
evaluations	8	17	19	24

Table 2: Effect of different definitions of expensive functions in problem Argauss

5.3 Comparison to other methods

We compare our method to four other solvers for nonlinear feasibility problems. As EFNES is primary based on the filter trust-region method FILTRANE [17], we have chosen FILTRANE as part of the GALAHAD package [15] for comparison. Moreover, we consider TRESNEI [27], 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.

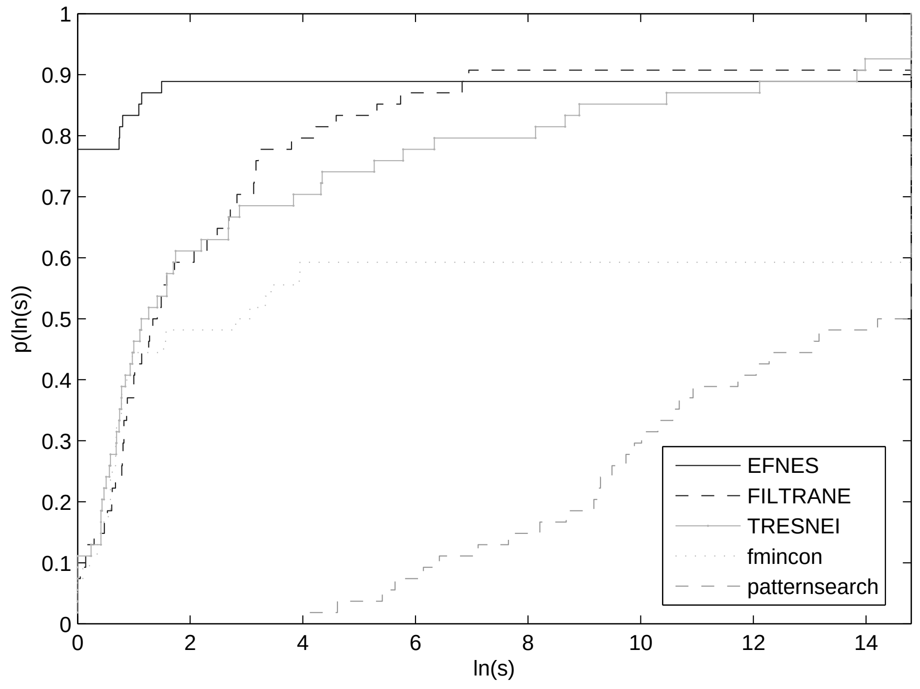


Figure 2: Performance profile counting the number of expensive function evaluation

All algorithms are compared in the number of evaluations of the expensive function and in the required computation time. To measure the computation time, we use the tools provided by Matlab (or 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. Also TRESNEI provides information about the number of evaluations, but without consideration of the additional function evaluations necessary for building the Jacobian. Thus, we added these evaluations to obtain the total number of expensive function evaluations. We run FILTRANE

with forward differentiation to obtain the derivative, i.e., the gradient and the hessian that are normally used were not regarded to simulate that expensive functions generally do not provide derivative information.
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 2 presents a performance profile [8] comparing all codes in the number of evaluations of the expensive function. For every $s \geq 1$, a performance profile shows the fraction $p(\ln(s))$ of test problems, 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 logarithmical here.
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.

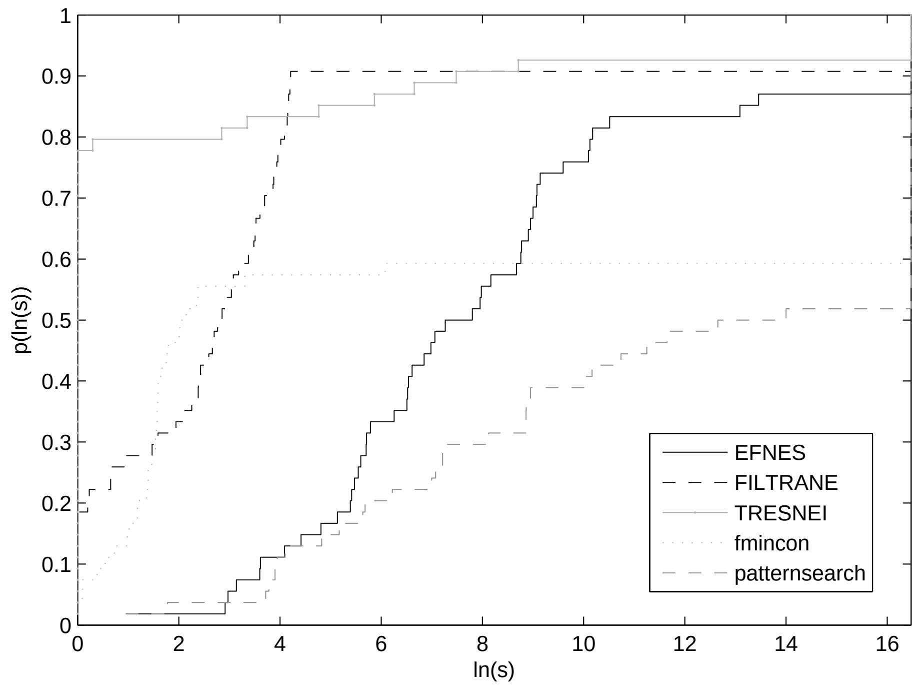


Figure 3: Performance profile comparing the computation time

Figure 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 as, in every iteration, a whole feasibility problem has to be solved as subproblem (CS_k). 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 0.56 seconds (respectively 0.18 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 34.74 (respectively 10.06) seconds.

In many practical applications, numerical simulations have a much longer execution time. Thus, our method significantly reduces, in general, the total cost of solving feasibility problems that involve expensive function evaluations.

6 Conclusion

We have introduced a new algorithm for solving feasibility problems that involve expensive function evaluations like, e.g., numerical simulations. This algorithm is based on the trust-region idea. In addition, it is combined with the filter of Fletcher and Leyffer [12] to achieve feasibility and it applies conditional models for the expensive function to keep, on average, the number of expensive function evaluations low. Moreover, a nested function approach for the problem formulation yields a lot of flexibility to incorporate as much information as available. We show that this problem formulation can also reduce the number of required expensive function evaluations. For our algorithm, we prove convergence to a first-order critical point of a least-squares reformulation of the feasibility problem from an arbitrary starting point under common assumptions. Numerical results on a set of 54 test problems show, in comparison to four other algorithms for feasibility problems, a considerable reduction of expensive function evaluations. If the functions are expensive in terms of evaluation time, our algorithm turns out to be, on average, more effective if a single function evaluation takes longer than some seconds. Consequently, as numerical simulations often take minutes or even hours in practical applications, our algorithm can provide a significant improvement in the total solution time of simulation based feasibility problems.

7 Acknowledgement

The authors would like to thank Prof. Dr. Philippe Toint and Prof. Dr. Kathrin Klamroth for the fruitful discussions and helpful comments.
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, 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] 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.

[21] 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.

[22] 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.

[23] 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.

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

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

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

[27] 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, 2009.

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

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

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

[31] 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.