Line-search based optimization using function approximations with tunable accuracy

Dane S. Grundvig & Matthias Heinkenschloss

To cite this article: Dane S. Grundvig & Matthias Heinkenschloss (09 Dec 2024): Line-search based optimization using function approximations with tunable accuracy, Optimization Methods and Software, DOI: 10.1080/10556788.2024.2436192

To link to this article: https://doi.org/10.1080/10556788.2024.2436192

Published online: 09 Dec 2024.

OPTIMIZATION METHODS & SOFTWARE
https://doi.org/10.1080/10556788.2024.2436192

Line-search based optimization using function approximations with tunable accuracy

Dane S. Grundvig and Matthias Heinkenschloss

Department of Computational Applied Mathematics and Operations Research, MS-134, Rice University, Houston, TX, USA

ABSTRACT
This paper develops a line-search algorithm that uses objective function models with tunable accuracy to solve smooth optimization problems with convex constraints. The evaluation of objective function and its gradient is potentially computationally expensive, but it is assumed that one can construct effective, computationally inexpensive models. This paper specifies how these models can be used to generate new iterates. At each iteration, the model has to satisfy function error and relative gradient error tolerances determined by the algorithm based on its progress. Moreover, a bound for the model error is used to explore regions where the model is sufficiently accurate. The algorithm has the same first-order global convergence properties as standard line-search methods, but only uses the models and the model error bounds. The algorithm is applied to problems where the evaluation of the objective requires the solution of a large-scale system of nonlinear equations. The models are constructed from reduced order models of this system. Numerical results for partial differential equation constrained optimization problems show the benefits of the proposed algorithm.

ARTICLE HISTORY
Received 5 June 2024
Accepted 26 November 2024

KEYWORDS
Optimization; line-search; inexact function; inexact gradient; reduced order model

1. Introduction

We develop a line-search algorithm that uses objective function models with tunable accuracy to solve smooth optimization problems with convex constraints. Given Hilbert space $(X, \langle\cdot,\cdot\rangle_X)$, a closed convex set $C \subset X$ and a smooth function $f : C \to \mathbb{R}$, the optimization problem is

$$\min_{x \in C} f(x). \tag{1}$$

Let $\Pi_C : X \to C$ be the projection onto C. We assume that the evaluation of f and its gradient is computationally expensive, e.g. because the evaluation of f at x requires an expensive simulation, but that one can compute differentiable, effective, computationally inexpensive to evaluate models m_k of the objective function around the current iterate x_k. Instead of traditional line-search methods that use Taylor expansions at the current iterate to build a convex quadratic model and then construct a new iterate using an approximate minimizer

CONTACT Matthias Heinkenschloss heinken@rice.edu Department of Computational Applied Mathematics and Operations Research, MS-134, Rice University, 6100 Main Street, Houston, TX 77005-1892, USA

of this quadratic model (see, e.g. [14, Section 6.3], [36, Ch. 3], [28]), we want a line-search method that uses m_k. In this paper, we specify what approximation properties are required of these models at the current iterate x_k, and how these models can be used to generate a new iterate in the context of line-search methods. While once generated, a model m_k and its gradient are computationally inexpensive to evaluate, the computation of this model, however, carries a computational cost. Therefore, it is desirable to use a current model as much as possible. Motivated by [51], we develop criteria based on error bounds between the current model m_k and the true objective to specify regions in parameter space in which the model can be used.

Our algorithm determines function and gradient tolerances τ_k^f, τ_k^g and requires that

$$|m_k(x_k) - f(x_k)| \le \tau_k^f \tag{2a}$$

and

$$\begin{aligned} &\|\Pi_C(x_k - \alpha_k^{(0)} \nabla m_k(x_k)) - \Pi_C(x_k - \alpha_k^{(0)} \nabla f(x_k))\|_X \\ &\quad \le \|x_k - \Pi_C(x_k - \alpha_k^{(0)} \tau_k^g \nabla m_k(x_k))\|_X, \end{aligned} \tag{2b}$$

where $\alpha_k^{(0)}$ is the initial trial step-size in iteration k. Moreover, our algorithm assumes that a bound $e_k(x)$ for the model error $|m_k(x) - f(x)|$ is available and uses this bound to explore the model in regions where it is sufficiently accurate to compute the new iterate. Our algorithm has the same first-order global convergence properties of standard line-search methods, but our algorithm only uses the models m_k and the error functions e_k and never directly accesses the original objective function.

The use of a Hilbert space $(X, \langle \cdot, \cdot \rangle_X)$ setting does not significantly change the algorithmic setup and theory compared to the specific case $X = \mathbb{R}^n$ and $\langle x, y \rangle_X = x^T y$. However, in many cases the problem (1) with $X = \mathbb{R}^n$ is obtained as the discretization of an infinite dimensional problem and the inner product is a weighted Euclidean inner product $\langle x, y \rangle_X = x^T M y$ with a symmetric positive definite matrix $M = \mathbb{R}^{n \times n}$. Posing the problem in a Hilbert space setting accommodates this and other problem settings.

Our algorithm and theory are agnostic about model construction. Of special interest, both in terms of applications and in terms of algorithmic development, are an important class of problems where the kth model m_k is computed using projection-based reduced order models (ROMs) of systems that underlay the definition of the objective function. We specify theoretically and illustrate numerically how these models can be used with our proposed optimization algorithm. Specifically, we consider objective functions given by $f(x) = \widetilde{f}(x, y(x))$, where $y(x) \in \mathbb{R}^{n_y}$, $n_y \gg 1$, solves a large-scale system of (nonlinear) equations $R(x, y) = 0$. These objective functions arise in applications where x represents the input (e.g. a control or a vector of design parameters) into a system, which is modelled by $R(x, y) = 0$, and the state of the system $y(x) \in \mathbb{R}^{n_y}$. The state $y(x)$ together with x are used to specify the objective, which quantifies how well the system performs. Specific examples will be discussed in Section 4. To compute a model in the kth iteration, a vector $\bar{y}_k \in \mathbb{R}^{n_y}$, matrices $V_k, W_k \in \mathbb{R}^{n_y \times r_k}$, $r_k \ll n_y$ and a small-scale ROM $W_k^T R(x, \bar{y}_k + V_k \widehat{y}_k(x)) = 0$ are constructed such that $y(x) \approx \bar{y}_k + V_k \widehat{y}_k(x)$ for x around x_k. The objective function model is $m_k(x) = \widetilde{f}(x, \bar{y}_k + V_k \widehat{y}_k(x))$. The quality of the model can be tuned by adjusting the generation of $\bar{y}_k$, V_k, W_k. Problems of this form arise in many important applications, see, e.g.

[2,17,32,34,35,41,45,47,49,52], and the rigorous and efficient integration of ROM-based objective function models enables faster solution of these problems.

The use of models m_k for the minimization of a smooth function f is related to the question of how much error in the function f and its gradient can be tolerated. These issues have been analysed for trust-region methods and, often in different problem contexts, for line-search based methods. Next, we will review previous approaches and contrast them with the one in this paper.

The use of inexact function and gradient information in trust-region methods was first studied in [11,12], where conditions on the size objective function and gradient errors were developed to ensure convergence of trust-region methods. See also [13, Sections 8.4,10.6]. The paper [1] explored the use of more general smooth models instead of traditional Taylor expansion based models in trust-region methods. They require that the error between the models and the original objective function, as well as the error between their gradients satisfy the conditions in [11,12]. The conditions on function and gradient error in [11,12], [13, Sections 8.4,10.6] require rather precise error bounds which are difficult to implement in practice. The papers [23], [30, Section 4] relaxed the gradient conditions in [11] to make them implementable when only asymptotic error bounds are available. The paper [31, Section 4] additionally incorporated inexactness in function evaluations and developed conditions that are implementable when only asymptotic error bounds are available. See also the survey [29]. In [30,31] the theory of trust-region methods with inexact function and gradient evaluations was applied to optimization under uncertainty, and errors in objective and gradient evaluation were due to sampling / approximation errors for the expected value. Related trust-region theory was used in [44,54] for multilevel partial differential equation (PDE) constrained optimization. The trust-region theory of [11] was used in [15] to manage the construction of Proper Orthogonal Decomposition (POD) based ROMs for PDE constrained optimization. The improved trust-region theory of [30,31] was used, e.g. in [45,48,52,53] to manage the construction of ROMs for PDE constrained optimization. The trust-region convergence theory is quite powerful, but present trust-region methods have a potential disadvantage. If the ratio between predicted (by the model) decrease and actual decrease is small, the computed trial step is rejected, the trust-region radius is reduced, and the model is refined. The trust-region model is then increased in subsequent successful iterations. Thus, a poorly constructed temporary model may result in small trust-region radius. Even if the model refinement vastly improves the fidelity of the model, current trust-region algorithms still use the reduced trust-region radius and only allow it to increase relatively slowly. This motivates the use of line-search methods, which compute steps and step-size based on the current model, but do not base these computations on past step and step-size selections.

In contrast to papers on trust-region methods for problems with inexact function and gradient information, there are few papers on this issue for line-search methods. The report [18] provides conditions on gradient errors that can be permitted while still guaranteeing global convergence. Most of the recent line-search papers focus on problems with stochastic noise in function or gradient evaluation, which is a different problem setting than the one considered in this paper. For example, [8,50] consider line-search methods for stochastic optimization problems and propose adaptive methods for choosing sample sizes which construct approximate gradients in this context. Similar results are presented in [9], where a criterion is developed for deciding sample sizes in a line-search based scheme for solving a machine learning optimization problem. In [37], convergence of a line-search method with random gradients is analysed, and this analysis is extended to also include noisy functions in [5]. The paper [43] studies proximal gradient methods for convex optimization with errors in the gradient of the smooth component of the function and in the proximity operator. These errors enter the bounds for the convergence of function values, but are not adjusted to achieve convergence to the solution of the original problem. As we have mentioned earlier, the above line search papers address a different problem setting motivated by stochastic optimization. In contract, we assume that given a current iterate x_k and tolerances τ_k^f and τ_k^g, which will be determined by our optimization algorithm, one can generate a model m_k such that (2a) holds and that a bound $e_k(x)$ for the model error $|m_k(x) - f(x)|$ is available. The previously mentioned line-search approaches are not applicable in this context.

Our line-search algorithm is motivated by [51]. While the algorithm in [51] has flavours of both line-search and trust-region, the line-search algorithm elements are responsible for global convergence and we therefore classify [51] as a line-search algorithm. The algorithm in [51] was used in [39] and in [27] to manage the construction of reduced basis ROMs for specific optimization problems governed by linear PDEs. The algorithm in [51] and the one proposed in this paper compute a so-called generalized Cauchy point using a descent direction of the current model and then compute a new iterate that further reduces the model compared to the generalized Cauchy point. The iterate is obtained by approximately minimizing the current model m_k subject to constraints involving the bound $e_k(x)$ for the model error $|m_k(x) - f(x)|$. To ensure well-posedness of the algorithm and global convergence results, the model m_k may need to be refined. The algorithms in [51] and the one proposed in this paper differ in when and how this refinement is applied, and in the formulation of the minimization subproblem. Our proposed algorithm has fewer conditions on substeps, and it detects early in the iteration whether the current model needs to be refined. In addition we relax several conditions imposed in [51] and present a simplified convergence analysis. The basic convergence theories in [51] and in this paper both assume that the number of model refinements needed in every iteration is uniformly bounded. Because the optimization algorithms are agnostic about model construction, this assumption can only be verified when a specific model construction is chosen. In [51] this assumption is numerically observed to be valid for the examples considered. We prove that in our setting this assumption is satisfied for a broad class of ROM based models. This is possible because the need for model refinement is detected earlier in our algorithm and our algorithm has fewer substeps. Finally, we allow convex constraints in (1). However, this is a fairly straightforward extension.

This paper is organized as follows. In Section 2.1 we present the basic line-search algorithm and a basic convergence result. The initial convergence theory assumes a condition that involves $m_{k+1}(x_{k+1})$ (see condition (8) in the next section). Since x_{k+1} is computed iteratively, naive application of this condition requires construction of a new model m_{k+1} at every trial iterate x_{k+1}. Section 2.2 develops conditions that allow the computation of x_{k+1} based on the current model m_k and on the bound $e_k(x)$ for the model error $|m_k(x) - f(x)|$. These conditions allow one to first compute the new iterate x_{k+1} and then construct the new model. The approach of [51] is summarized in Section 2.3 and compared with the proposed approach. As we mentioned earlier, the conditions developed in Section 2.2 that allow the computation of x_{k+1} without constructing a new model may require a refinement of the current model m_k. In Section 3 we consider models computed using a broad class of ROMs and we prove that a finite number of model refinements are sufficient. Section 4 applies our algorithm to model problems from [27,39], and [45].

2. Line-search with inexact function evaluations

2.1. General idea

Our algorithm seeks to find x_* that solves (1), such x_* will be in C and satisfy

$$x_* = \Pi_C(x_* - \alpha \nabla f(x_*)) \quad \forall\, \alpha > 0. \tag{3}$$

A point x_* that satisfies (3) is called *stationary*.

Assume that the original function f is Fréchet differentiable. Suppose that at every step k of the algorithm there is a continuously Fréchet differentiable approximation m_k of the true objective f. Furthermore, assume that for a given approximation there is a computable error bound function e_k such that

$$|m_k(x) - f(x)| \le e_k(x) \quad \forall\, x \in \mathbb{R}^n. \tag{4}$$

This error bound function e_k may be large, and could even take the value ∞ away from the current iterate x_k at which the current model m_k is built. However, we assume there is a method to generate an approximation m_k such that the error bound at x_k is below a tolerance τ_k^f,

$$e_k(x_k) \le \tau_k^f. \tag{5}$$

In addition, a relative error τ_k^g on the projected gradient will be described below. The tolerances τ_k^f and τ_k^g will ultimately be specified by our optimization algorithm. When objective function models are computed using ROMs of the underlying simulation, error bounds which satisfy (5) and (4) can be computed; see Section 3 for additional details.

We will be using models m_k of the true objective f to find critical points of (1). For line-search in the unconstrained case $C = X$, a general descent direction s_k with $\langle \nabla m_k(x_k), s_k\rangle_X < 0$ is used. In the constrained case, however, it is not guaranteed that $m_k(\Pi_C(x_k + \alpha s_k)) \le m_k(x_k)$ for sufficiently small $\alpha > 0$. See, e.g. [6, p. 225], [7, pp. 224–225]. Therefore search directions s_k need to be adjusted to C and x_k, e.g. [6, p. 225], [16]. More general search direction can be incorporated, but to focus the paper, we use $s_k = -\nabla m_k(x_k)$ and the projected gradient $\Pi_C(x_k - \alpha_k \nabla m_k(x_k))$. Specifically, we choose a step-size α_k such that the so-called generalized Cauchy point

$$x_k^C = \Pi_C(x_k - \alpha_k \nabla m_k(x_k)), \tag{6}$$

satisfies a sufficient decrease condition

$$m_k(x_k^C) \le m_k(x_k) - \frac{c_1}{\alpha_k}\|x_k - x_k^C\|_X^2. \tag{7}$$

In the unconstrained case $C = X$, the sufficient decrease condition (7) is identical to the traditional sufficient decrease condition with $s_k = -\nabla m_k(x_k)$.

In addition to (7) we will also require a lower bound on the step-size α_k. Under suitable conditions on the model m_k, which we will specify in Lemmas 2.1 and 2.2 below, there exist step-sizes α_k that satisfy (7) and there are several algorithms to compute these. See, e.g. [14, Section 6.3], [36, Ch. 3] for the case $X = \mathbb{R}^n$. One possible algorithm is the Backtracking Algorithm 1 below. In the simplest setting Algorithm 1 uses $\beta_1 = \beta_2$, which reduces the trial step-size $\alpha_k^{(\ell)}$ by a constant factor β_1. The choice $0 < \beta_1 \le \beta_2 < 1$ allows one to use polynomial models of $\alpha \mapsto m_k(x_k + \alpha s_k)$ to potentially find α_k faster. See, e.g. [14, Section 6.3.2].

Algorithm 1: Backtracking Line-Search

```
Require: $x_k, s_k \in X, c_1 \in (0,1), 0 < \beta_1 \le \beta_2 < 1, \alpha_k^{(0)} > 0$.
Ensure: Step-size $\alpha_k$.
1: for $i = 0, 1, 2, \dots$ do
2:     If (7) is satisfied by $\alpha_k^{(i)}$; return with $\alpha_k = \alpha_k^{(i)}$.
3:     Compute $\alpha_k^{(i+1)} \in [\beta_1 \alpha_k^{(i)}, \beta_2 \alpha_k^{(i)}]$.
4: end for
```

Lemma 2.1: *Let m_k be continuously Fréchet differentiable in an open set $\mathcal{D}_k$ containing $x_k \in C$. If the gradient of m_k is Lipschitz continuous in the set $\mathcal{D}_k$ with Lipschitz constant L_k, then the sufficient decrease condition (7) is satisfied for all $\alpha_k \in (0, 2(1-c_1)/L_k)$.*

For a proof, see e.g. [28, pp. 94]. This reference considers the case $X = \mathbb{R}^n$ and C given by box constraints, but the arguments can be directly extended to our case.

Since the backtracking line-search Algorithm 1 reduces the trial step-size at least by a factor $\beta_2 < 1$, it will find $\alpha_k = \alpha_k^{(i)}$ such that the sufficient decrease condition (7) is satisfied after a finite number of reductions.

Lemma 2.2: *If the assumptions of Lemma 2.1 hold, if the initial step-size of Algorithm 1 satisfies $\alpha_k^{(0)} \ge 2(1-c_1)/L_k$, and if $x_k + \alpha s_k \in \mathcal{D}_k$ for all $\alpha \in [0, \alpha_k^{(0)}]$, then the step-size computed by Algorithm 1 satisfies $\alpha_k \ge 2\beta_1(1-c_1)/L_k$.*

Proof: If Algorithm 1 returns $\alpha_k = \alpha_k^{(0)}$, the lower bounds follows from the assumption on $\alpha_k^{(0)}$ and $\beta_1 \in (0,1)$. If $\alpha_k = \alpha_k^{(i)}$ for $i \ge 1$, then $\alpha_k^{(i-1)}$ did not satisfy the sufficient decrease condition (7), i.e.

$$\begin{aligned} m_k(x_k - \alpha_k^{(i-1)} \nabla m_k(x_k)) - m_k(x_k) &> -\frac{c_1}{\alpha_k^{(i-1)}} \| x_k - \Pi_C(x_k - \alpha_k^{(i-1)} \nabla m_k(x_k)) \|_X^2 \\ &\ge -c_1 \alpha_k^{(i-1)} \| \nabla m_k(x_k) \|_X^2, \end{aligned}$$

where we used $x_k = \Pi_C(x_k)$ and the fact that projections obey $\|\Pi_C(x) - \Pi_C(y)\|_X \le \|x - y\|_X$ for all $x, y \in X$. This implies

$$m_k(x_k - \alpha_k^{(i-1)} \nabla m_k(x_k)) - m_k(x_k) + \alpha_k^{(i-1)} \| \nabla m_k(x_k) \|_X^2 \ge (1-c_1) \alpha_k^{(i-1)} \| \nabla m_k(x_k) \|_X^2.$$

Moreover,

$$\begin{aligned} & m_k(x_k - \alpha_k^{(i-1)} \nabla m_k(x_k)) - m_k(x_k) + \alpha_k^{(i-1)} \|\nabla m_k(x_k)\|_X^2 \\ & = \alpha_k^{(i-1)} \int_0^1 \langle \nabla m_k(x_k) - \nabla m_k(x_k - t\alpha_k^{(i-1)} \nabla m_k(x_k)), \nabla m_k(x_k)\rangle_X dt \\ & \leq (\alpha_k^{(i-1)})^2 \frac{L_k}{2} \|\nabla m_k(x_k)\|_X^2. \end{aligned}$$

Combining the previous two inequalities gives $\alpha_k^{(i-1)} \geq 2(1-c_1)/L_k$. The desired bound follows from $\alpha_k = \alpha_k^{(i)} \geq \beta_1 \alpha_k^{(i-1)}$. ■

Corollary 2.3: *If the assumptions of Lemma 2.2 hold and the Lipschitz constants are uniformly bounded, $L_k \leq L$, $k \in \mathbb{N}$, then $\alpha_k \geq 2\beta_1(1-c_1)/L$.*

In traditional line-search methods the new iterate is $x_{k+1} = x_k^C$, but we allow $x_{k+1} \neq x_k^C$. In our algorithm, the generalized Cauchy point (6) will ensure convergence. The new iterate x_{k+1} will be computed by exploring the model m_k more fully to potentially accelerate the convergence.

The proof of global convergence of a line-search method with exact function information uses a telescoping sum argument, see, e.g. [14, pp. 121–123] or [36, pp. 38–39]. This argument can be easily extended to the inexact case (see Theorem 2.5 below) if the models satisfy

$$m_k(x_k) - m_{k+1}(x_{k+1}) \geq a_1 \left(m_k(x_k) - m_k(x_k^C) \right) \tag{8}$$

for some $a_1 \in (0,1]$. The closer a_1 is to zero, the easier it is for (8) to be satisfied. However, for a_1 closer to zero less decrease is enforced and the algorithm may converge slower. Note that if $m_{k+1}(x_{k+1}) \leq m_k(x_k^C)$, then (8) is satisfied for any $a_1 \in (0,1]$ because $m_k(x_k^C) < m_k(x_k)$ by the sufficient decrease condition (7).

It is non-trivial how to satisfy condition (8) in a computationally efficient way, see our discussion at the end of this section. Therefore, much of this paper will centre around computationally efficient ways of ensuring (8). If we accept (8) for now, then the basic line-search algorithm with inexact functions is summarized in Algorithm 2 below.

Algorithm 2: Line-Search Algorithm with Inexact Function Information

```
Require: $c_1 \in (0,1)$, $x_0 \in C$, stopping tolerance tol $> 0$
Ensure: Point $x_K$ where $\|x_K - \Pi_C(x_K - \alpha_K^{(0)} \nabla m_k(x_K))\|_X \leq$ tol.
1: Generate initial model $m_0$.
2: for $k = 0, 1, 2, \ldots$ do
3:     If $\|x_k - \Pi_C(x_k - \alpha_k^{(0)} \nabla m_k(x_k))\|_X <$ tol, then return $x_k$.
4:     Use Backtracking Line-Search Algorithm 1 to compute $\alpha_k$ such that
       $x_k^C = \Pi_C(x_k - \alpha_k \nabla m_k(x_k))$ satisfies the sufficient decrease condition (7).
5:     Compute $x_{k+1} \in C$ and a new model $m_{k+1}$ such that (8) is satisfied.
6: end for
```

Note, that Algorithm 2 says nothing about constructing m_k and does not consider the difficulty of finding x_{k+1} and the new model m_{k+1} such that (8) is satisfied. These issues will be the focus of the following sections. For convergence, we introduce an additional condition on the relative gradient error,

$$\frac{\|\Pi_C(x_k - \alpha_k^{(0)}\nabla m_k(x_k)) - \Pi_C(x_k - \alpha_k^{(0)}\nabla f(x_k))\|_X}{\|x_k - \Pi_C(x_k - \alpha_k^{(0)}\nabla m_k(x_k))\|_X} \le \tau_k^g \le \tau_g, \tag{9}$$

where $\alpha_k^{(0)}$ is the initial step-size in the Backtracking Algorithm 1. Note that (9) is implied by

$$\frac{\alpha_k^{(0)}\|\nabla m_k(x_k) - \nabla f(x_k)\|_X}{\|x_k - \Pi_C(x_k - \alpha_k^{(0)}\nabla m_k(x_k))\|_X} \le \tau_k^g \le \tau_g. \tag{10}$$

Often $\alpha_k^{(0)} = 1$, but since (projected) gradient methods are scaling dependent, the initial step size should be adjusted to the problem scaling. In the unconstrained case $C = X$, (9) and (10) reduce to the standard relative gradient tolerance $\|\nabla m_k(x_k) - \nabla f(x_k)\|_X / \|\nabla m_k(x_k)\|_X \le \tau_k^g \le \tau_g$.

The convergence result in Theorem 2.5 requires the following lemma.

Lemma 2.4: *For every $x \in C$ and $z \in X$, the function $g : (0, \infty) \to \mathbb{R}$ defined by*

$$g(\alpha) = \frac{1}{\alpha}\|\Pi_C(x + \alpha z) - x\|_X$$

is monotonically nonincreasing, i.e. $g(\alpha) \le g(\tilde{\alpha})$ for $\alpha > \tilde{\alpha}$.

For a proof see, e.g. [16, Lemma 1].

If Algorithm 2 generates a sequence of iterates $\{x_k\}$ which satisfy the relative gradient condition (9), then we can prove the convergence result in Theorem 2.5. This result is a generalization of the well-known line-search convergence result, see, e.g. [14, Theorem 6.3.3] or [36, pp. 38–39].

Theorem 2.5: *If the objective function models m_k are continuously Fréchet differentiable with Lipschitz continuous gradients ∇m_k, if the Lipschitz constants are uniformly bounded, if $m_k(x_k) \ge M$ for all k, if the initial step-sizes in the Backtracking Algorithm 1 satisfy $2(1 - c_1)/L_k \le \alpha_k^{(0)} \le \alpha^{(0)}$, and if the Line-Search Algorithm 2 generates a sequence $\{x_k\}$ of iterates, then*

$$\lim_{k\to\infty} \|x_k - \Pi_C(x_k - \alpha^{(0)}\nabla m_k(x_k))\|_X = 0. \tag{11}$$

If, in addition, the objective function models m_k have bounded relative projected gradient error (9), then

$$\lim_{k\to\infty} \|x_k - \Pi_C(x_k - \alpha^{(0)}\nabla f(x_k))\|_X = 0. \tag{12}$$

Proof: Since x_{k+1} satisfies (8) and $\{m_k(x_k)\}$ is bounded from below,

$$\begin{aligned}\infty > m_0(x_0) - m_{K+1}(x_{K+1}) &= \sum_{k=0}^{K} m_k(x_k) - m_{k+1}(x_{k+1}) \\ &\geq \sum_{k=0}^{K} a_1 \left(m_k(x_k) - m_k(x_k^C) \right) > 0\end{aligned}$$

for all K. Applying the sufficient decrease (7) gives

$$\infty > \sum_{k=0}^{\infty} m_k(x_k) - m_k(x_k^C) \geq \sum_{k=0}^{\infty} c_1 \alpha_k \frac{\|x_k - \Pi_C(x_k - \alpha_k \nabla m_k(x_k))\|_X^2}{\alpha_k^2}. \tag{13}$$

Because the Lipschitz constants L_k of ∇m_k are uniformly bounded, $L_k \leq L$, Corollary 2.3 implies the step-size condition

$$\alpha_k \geq 2\beta_1(1-c_1)/L. \tag{14}$$

Moreover, since the step-sizes generated by the Backtracking Algorithm 1 satisfy $\alpha_k \leq \alpha_k^{(0)}$, the monotonicity result in Lemma 2.4 implies

$$\frac{\|x_k - \Pi_C(x_k - \alpha_k \nabla m_k(x_k))\|_X}{\alpha_k} \geq \frac{\|x_k - \Pi_C(x_k - \alpha_k^{(0)} \nabla m_k(x_k))\|_X}{\alpha_k^{(0)}}. \tag{15}$$

Inserting (14), (15) into (13) gives

$$\infty > \sum_{k=0}^{\infty} \left(\frac{1}{\alpha_k^{(0)}} \|x_k - \Pi_C(x_k - \alpha_k^{(0)} \nabla m_k(x_k))\|_X \right)^2. \tag{16}$$

Since the initial step-sizes in the Backtracking Algorithm 1 satisfy $\alpha_k^{(0)} \leq \alpha^{(0)}$, (16) and the monotonicity result in Lemma 2.4 imply

$$0 = \lim_{k\to\infty} \frac{\|x_k - \Pi_C(x_k - \alpha_k^{(0)} \nabla m_k(x_k))\|_X}{\alpha_k^{(0)}} = \lim_{k\to\infty} \frac{\|x_k - \Pi_C(x_k - \alpha^{(0)} \nabla m_k(x_k))\|_X}{\alpha^{(0)}},$$

which is (11).

The projected gradient error condition (9) and (16) imply

$$\frac{\|x_k - \Pi_C(x_k - \alpha_k^{(0)} \nabla f(x_k))\|_X}{\alpha_k^{(0)}} \leq (1+\tau_g) \frac{\|x_k - \Pi_C(x_k - \alpha_k^{(0)} \nabla m_k(x_k))\|_X}{\alpha_k^{(0)}} \to 0$$

as $k \to \infty$. Since initial step-sizes in the Backtracking Algorithm 1 satisfy $\alpha_k^{(0)} \leq \alpha^{(0)}$, the monotonicity result in Lemma 2.4 implies

$$0 = \lim_{k\to\infty} \frac{\|x_k - \Pi_C(x_k - \alpha_k^{(0)} \nabla f(x_k))\|_X}{\alpha_k^{(0)}} = \lim_{k\to\infty} \frac{\|x_k - \Pi_C(x_k - \alpha^{(0)} \nabla f(x_k))\|_X}{\alpha^{(0)}},$$

which is (12). ■

Versions of Algorithm 2 and Theorem 2.5 (as well as the algorithms and theorems in the following sections) exist for the unconstrained case $C = X$, see [20, Ch. 2]. In the unconstrained case, a more general search direction s_k is allowed. More general search directions s_k can be used in the constrained case as well as long as results analogous to Lemmas 2.1, 2.2 hold.

Theorem 2.5 guarantees convergence of the Line-Search Algorithm 2 provided that it is well-posed, i.e. that there actually is a sequence $\{x_k\}$ of iterates. The critical condition is (8). The model m_k is used to compute x_k^C and x_{k+1}, and it is relatively easy to generate these quantities to ensure (7) and $m_k(x_{k+1}) \le m_k(x_k^C)$. Additionally, the alternate condition

$$m_k(x_k) - m_k(x_{k+1}) \ge a_1 \left(m_k(x_k) - m_k(x_k^C) \right), \tag{17}$$

which only uses the current model m_k is relatively easy to satisfy. However, (8) replaces $m_k(x_{k+1})$ in (17) by the evaluation at the new model $m_{k+1}(x_{k+1})$. This makes (8) difficult to work with in general, because x_{k+1} is typically computed iteratively, and naive incorporation of (8) requires that a model m_{k+1} is constructed at every trial iterate for x_{k+1}. This is computationally inefficient because generation of a model m_{k+1} is typically computationally expensive. Thus, for a version of Line-Search Algorithm 2 that is applicable in practice it is important to separate computation of x_{k+1} from the generation of m_{k+1} as much as possible, i.e. to allow computation of x_{k+1} using the current model m_k and generate a new m_{k+1} afterwards. This is done by expanding on an idea of [51]. By definition (4) of the function error,

$$\begin{aligned} & m_k(x_k) - m_{k+1}(x_{k+1}) \\ &\quad = m_k(x_k) - m_k(x_{k+1}) + m_k(x_{k+1}) - f(x_{k+1}) + f(x_{k+1}) - m_{k+1}(x_{k+1}) \\ &\quad \ge m_k(x_k) - m_k(x_{k+1}) - e_k(x_{k+1}) - e_{k+1}(x_{k+1}). \end{aligned}$$

Therefore, (8) is implied by

$$-e_{k+1}(x_{k+1}) - e_k(x_{k+1}) + m_k(x_k) - m_k(x_{k+1}) \ge a_1 \left(m_k(x_k) - m_k(x_k^C) \right). \tag{18}$$

The condition (18) works with error estimators and will next be used to devise a practical algorithm to compute x_{k+1} and update the function model m_{k+1}.

2.2. Computation of the new iterate and model adjustment

The new iterate x_{k+1} must satisfy (18), which implicitly still involves the new model m_{k+1} through the error bound e_{k+1}. Next, we will untangle this dependency.

First, equivalently write (18) as follows

$$e_{k+1}(x_{k+1}) \le -e_k(x_{k+1}) - m_k(x_{k+1}) + m_k(x_k^C) + (1 - a_1) \left(m_k(x_k) - m_k(x_k^C) \right). \tag{19}$$

Condition (19) implies (8). As before, the condition (19) is not verifiable without the construction of a new model m_{k+1}. Therefore, (19) is not used directly but will be the impetus to find other, implementable, conditions. If x_{k+1} can be computed so that the right hand side in (19) is strictly positive, then (19) becomes a condition on the new model m_{k+1}: The new model m_{k+1} must be computed so that its error $e_{k+1}(x_{k+1})$ at the new iterate x_{k+1} satisfies (19). Together, the following two conditions ensure the positivity of the right hand side,

$$m_k(x_{k+1}) \le m_k(x_k^C) \tag{20}$$

and

$$e_k(x_{k+1}) \le a_2(1-a_1)\left(m_k(x_k) - m_k(x_k^C)\right), \tag{21}$$

where $a_2 \in (0,1)$. Note that a_2 provides some slack in the sense that the smaller a_2 is, the more positive the right hand side of (19).

In summary, if a generalized Cauchy point $x_k^C = \Pi_C(x_k - \alpha_k \nabla m_k(x_k))$ and a new iterate $x_{k+1} \in C$ can be constructed so that (20) and (21) hold (and the sufficient decrease condition (7) for generalized Cauchy point holds), then condition (19) becomes a condition on the error $e_{k+1}(x_{k+1})$ of the new model e_{k+1} at x_{k+1}. A new model m_{k+1} that satisfies (19) can be constructed whenever the right hand side in (19) is positive. The conditions (20), (21), and (19) imply (18) and (8), and therefore Theorem 2.5 guarantees convergence.

The condition (21) motivates the choice of constraint in the following minimization problem for computing the new iterate x_{k+1}. Given x_k^C, find x_{k+1} as the approximate solution of

$$\min \quad m_k(x) \tag{22a}$$

$$\text{s.t.} \quad x \in C, \quad e_k(x) \le a_2(1-a_1)\left(m_k(x_k) - m_k(x_k^C)\right). \tag{22b}$$

Clearly, any feasible point x_{k+1} of (22a) satisfies (21). This leaves the condition (20). If the Cauchy point is feasible for (22a), i.e. if $x_k^C \in C$ and

$$e_k(x_k^C) \le a_2(1-a_1)\left(m_k(x_k) - m_k(x_k^C)\right), \tag{23}$$

then any feasible point x_{k+1} of (22a) with a model function value lower than $m_k(x_k^C)$ qualifies as a new iterate. Hence, it is not necessary to compute a minimizer x_{k+1} of (22a), but any feasible point x_{k+1} of (22a) that satisfies (20) can be used.

Example 2.6: The model m_k and the error bound function e_k can be used to explore the model more globally via (22a). We illustrate (22a) using a simple two-parameter version, $X = \mathbb{R}^2$, of the thermal fin problem described in Section 4.1. To reduce the problem in Section 4.1 to a two-dimensional one, we assume heat conductivities $\kappa_0 = \ldots = \kappa_4 = \kappa \in [0.1, 10]$ and Biot number $\mathrm{Bi} \in [0.01, 1]$, and use the two variables $x = (\kappa, \mathrm{Bi})^T$. For more details on the problem setup, construction of m_k, etc., see Section 4.1. Figure 1 shows zoomed-in representations of the contours of the objective model m_k and of the feasible region for (22a) for the first three iterations. In each plot, the thick line indicates the boundary of the feasible set.

As we have stated before, if (23) is satisfied, then any feasible point x_{k+1} of (22a) with a model function value lower than $m_k(x_k^C)$ qualifies as a new iterate. Thus, the remaining

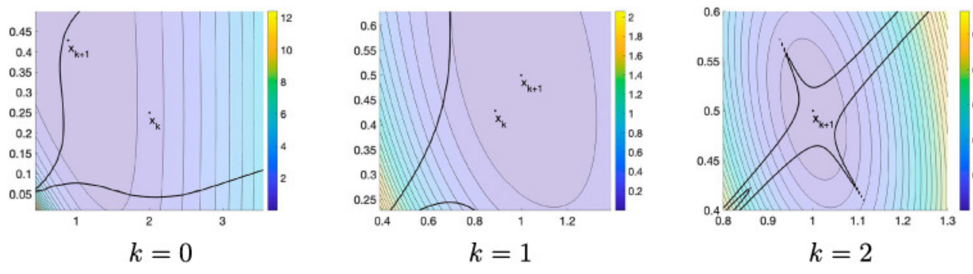

Figure 1. Contours (thin lines) of the objective model m_k and boundary (thick line) of the feasible region for (22a) for the first three iterations of a two dimensional version of the thermal fin problem in Section 4.1. Each plot zooms into regions around x_k and x_{k+1}. For iteration $k = 2$, $x_k \approx x_{k+1}$ and only x_{k+1} is shown.

question is whether (23) can be satisfied. In general, given a model m_k, a Cauchy point x_k^C that satisfies the sufficient decrease (7) condition will not necessarily satisfy (23). The condition (23) becomes a condition on the current model m_k. The model m_k and its gradient must not only approximate f and its gradient sufficiently well at x_k, the model m_k must also be sufficiently accurate in the sense that (23) holds at the generalized Cauchy point. If (23) is not satisfied, then there are two options: 1) Backtrack the generalized Cauchy point, or 2) refine the model m_k. We will discuss these next.

The idea behind backtracking is simple. If the model m_k sufficiently approximates the function f at x_k, then we can try to compute a generalized Cauchy point closer to x_k that satisfies (23) and (7). This backtracking continues to use the current model and avoids or at least delays a costly model update. Let $0 < \widetilde{\beta}_1 \le \widetilde{\beta}_2 < 1$. If $\alpha_k^{\text{old}} := \alpha_k$ satisfies the sufficient decrease condition (7), but (23) is violated, then we can try to find $\alpha_k \in [\widetilde{\beta}_1 \alpha_k^{\text{old}}, \widetilde{\beta}_2 \alpha_k^{\text{old}}]$ so that α_k satisfies the sufficient decrease condition (7) and $x_k^C = x_k + \alpha_k s_k$ satisfies (23). Note that since α_k^{old} satisfies $\alpha_k^{\text{old}} \ge 2\beta_1(1 - c_1)/L$, the step-size α_k satisfies $\alpha_k^{\text{old}} \ge \widetilde{\beta}_1 2\beta_1(1 - c_1)/L$. The specific construction of $\alpha_k \in [\widetilde{\beta}_1 \alpha_k^{\text{old}}, \widetilde{\beta}_2 \alpha_k^{\text{old}}]$ can depend on the properties of the model. If with this backtracked α_k the point $x_k^C = x_k + \alpha_k s_k$ satisfies (23) and (7), then we accept $x_k^C = x_k + \alpha_k s_k$. Otherwise we set $\alpha_k = \alpha_k^{\text{old}}$ and refine the model.

If the generalized Cauchy point (with additional backtracking or not) does not satisfy (23), we refine the current model m_k so that the refined model satisfies the function and gradient approximation properties at x_k, and ultimately also (23). This refinement process may have to be repeated. Model m_k construction is application specific. In Section 3 we will consider the important case of objective functions f whose evaluation requires computationally expensive simulation and of models m_k that are generated from ROMs of this simulation. We will show that in this case models m_k can be constructed so that the condition (23) is satisfied.

A summary of our line-search algorithm with inexact function information is given in Algorithm 3 below. Its convergence is summarized in Corollary 2.7 below.

Corollary 2.7: *Let the assumptions of Theorem 2.5 hold. If the number of sub-iterations in the model construction loop in line 4 of Algorithm 3 is finite and uniformly bounded, then the iterates calculated by the Line-Search Algorithm 3 satisfy* $\lim_{k\to\infty} \|x_k - \Pi_C(x_k - \alpha^{(0)} \nabla f(x_k))\|_X = 0$.

Algorithm 3: Line-Search with Inexact Function Information

```
Require: $c_1 \in (0,1), a_1 \in (0,1), a_2 \in (0,1], 0 < \beta_1 \le \beta_2 < 1, 0 < \widetilde{\beta}_1 \le \widetilde{\beta}_2 \le 1, \tau_0^f > 0$,
    $\tau_0^g \in (0,1)$, tolerance tol $> 0$, $x_0 \in X$.
Ensure: Point $x_K$ where $\|x_K - \Pi_C(x_K - \nabla m_K(x_K))\|_X <$ tol.
 1: Generate initial model $m_0$.
 2: for $k = 0, 1, 2, \dots$ do
 3:   If $\|x_k - \Pi_C(x_k - \nabla m_k(x_k))\|_X <$ tol then stop.
 4:   for $i = 0, 1, \dots$ do
 5:     if $i = 0$ then
 6:       Construct model $m_k$ that satisfies $e_k(x_k) \le \tau_k^f$ and (9).
 7:     else
 8:       Construct refined model $m_k$ that satisfies $e_k(x_k) \le \tau_k^f$, (23), and (9).
 9:     end if
10:     Find $\alpha_k$ that satisfies the sufficient decrease condition.
11:     Set the Cauchy point $x_k^C = \Pi_C(x_k - \alpha_k \nabla m_k(x_k))$.
12:     If $x_k^C = x_k + \alpha_k s_k$ satisfies (23) goto line 23.
13:     if $\widetilde{\beta}_2 < 1$ then
14:       Set $\alpha_k^{\text{old}} := \alpha_k$.
15:       Try to find $\alpha_k \in [\widetilde{\beta}_1 \alpha_k^{\text{old}}, \widetilde{\beta}_2 \alpha_k^{\text{old}}]$ that satisfies the sufficient decrease
          condition (7) and $x_k^C = x_k + \alpha_k s_k$ satisfies (23).
16:       if $x_k^C = x_k + \alpha_k s_k$ satisfies (23) then
17:         goto line 23.
18:       else
19:         Set $\alpha_k = \alpha_k^{\text{old}}$.
20:       end if
21:     end if
22:   end for
23:   Compute approximate solution $x_{k+1}$ of (22) that satisfies (20), (22b).
24:   Set $\tau_{k+1}^f = -e_k(x_{k+1}) - m_k(x_{k+1}) + m_k(x_k^C) + (1 - a_1)\left(m_k(x_k) - m_k(x_k^C)\right)$.
25: end for
```

Proof: The iterates x_k generated by Algorithm 3 satisfy the conditions (20) and (21), which imply (19), and therefore, (8). Thus, the iterates generated by Algorithm 3 satisfy Algorithm 2 and the desired result follows from Theorem 2.5. ■

The critical assumption in Corollary 2.7 is that the model construction loop in line 4 of Algorithm 3 is performed a finite and bounded number of times at each outer iteration k. In general, verification of this assumption depends on how the models m_k are constructed and refined. In the special case where the function error bound e_k is also efficient, i.e. is essentially proportional to the error, one can show that no refinement is needed, provided the gradient error is sufficiently small. Unfortunately, this condition on the error is very restrictive and we therefore omit the details. As mentioned before, in Section 3 we will consider the important case of objective functions governed by expensive simulations and models computed using ROMs of the expensive systems, and we will show that in this case the model construction loop in line 4 of Algorithm 3 is performed a finite number of times at each outer iteration.

2.3. Yue-Meerbergen algorithm

We briefly compare our algorithm with the one from Yue and Meerbergen [51]. A detailed treatment of the main components of the algorithm in [51] in the notation of our paper can be found in [20, Section 2.6]. The paper [51] considers $C = X = \mathbb{R}^n$. The paper [27] introduces an extension of [51] to the convex constrained case.

Given

$$\epsilon_L \in (0, 1),$$

which may be adjusted during the iteration, the ETR method in [51] and [27] consider the minimization subproblem

$$\min \quad m_k(x) \tag{24a}$$

$$\text{s.t.} \quad x \in C, \quad e_k(x) \leq \epsilon_L |m_k(x)|. \tag{24b}$$

($C = X = \mathbb{R}^n$ in [51]) instead of (22a) to compute a trial iterate x_{k+1}.

The convergence result in [51, Th. 3.3] is for the case when

$$m_{k+1}(x_{k+1}) \leq m_k(x_k^C), \tag{25}$$

called the 'ideal case' in [51, Section 3.2.2]. Note that (25) is (8) with $a_1 = 1$, this means that the condition (8) used in Algorithm 3 is an easier to satisfy relaxation of (25). The proof in [51, Th. 3.3], like our proof of Theorem 2.5, is a generalization of the well-known line-search convergence result. The algorithm developments in Sections 3.2.3, 3.3 and 4 of [51] are there to devise a practical way to compute an x_{k+1} and new model m_{k+1} such that (25) is satisfied, just like our development in Section 2.2 are there to devise a practical way to compute an x_{k+1} and new model m_{k+1} such that (8) is satisfied. Our development in Section 2.2 involves fewer steps and decisions, and in our case the decision to refine the model m_k is made based on the current generalized Cauchy point (see Line 8 in Algorithm 3), whereas in [51] the decision to refine the model is made after additional computations with the model m_k have been performed. Convergence of the algorithm from [51] requires that after a finite, uniformly bounded number of sub-steps/model refinements, an iterate x_{k+1} and new model m_{k+1} can be computed such that (25) holds. While in the numerical examples, convergence is observed, this is not proven for the set-up in [51]. Our convergence result, Corollary 2.7 also assumes that a finite and bounded number of model refinements are performed in each outer iteration. However, we will prove in Section 3 that this is the case for an important class of applications in which models are computed via ROMs.

The paper [27] extends the algorithm in [51] to constrained problems (1). As we argued earlier, the algorithm in [51] and its extension in [27] has flavours of both line-search and trust-region, but the line-search algorithm elements are responsible for global convergence. Therefore, we classify [51] and [27] as line-search algorithms. In addition to incorporating box constraints C, [27] also introduce a condition that allows the parameter ϵ_L, which controls the size of the constraint region in the subminimization problem (24a) to be increased. In [51], ϵ_L does not drive the convergence of the algorithm (the generalized Cauchy point and (25), called the 'ideal case' in [51, Section 3.2.2] drive convergence) but instead acts as a heuristic constraint on the subminimization problem (24a). Similarly, in [27], the convergence guarantees are similarly not influenced by ϵ_L so the additional possibility of increasing ϵ_L which they introduce is merely a heuristic that may improve the performance but does not change the convergence theory. Like the convergence proof in [51], the convergence result in [27, Thm. 4.5] requires that at most a finite and uniformly bounded number of model refinements are needed to compute an acceptable next iterate x_{k+1}. Moreover, the proof of [27, Thm. 4.5] uses a special feature of their model construction, which is a special case of the setting in Section 3. They consider a problem setting where objective evaluations $f(x)$ are dependent on the solution of an n_y dimensional system. They continue to enrich the ROM spaces and after n_y enrichments, the ROM objectives m_k are equal to the true objective f. Once this happens the constraint (24b) is never active. Unfortunately, this setting is not interesting in practice, since ultimately no ROM objectives m_k, but only different representations of f, are used. Our proof of our convergence Theorem 2.5 requires no such assumptions. Moreover, in the setting of Section 3.2 we can prove that the number of sub-iterations in the model construction loop in line 4 of Algorithm 3 is finite and uniformly bounded.

3. ROMs

As we have mentioned before, the critical assumption in Corollary 2.7 is that the model construction loop in line 4 of Algorithm 3 is performed a finite number of times at each outer iteration. Verification of this assumption depends on how the models $m_{k,i}$ are constructed and refined. In this section we will consider the important class of problems where the evaluation of the objective f requires a complex simulation, represented by a system of nonlinear equations, and the model $m_{k,i}$ is computed using a ROM of the system of nonlinear equations (see, e.g. [3] and [4]), and we will show that a finite number of model refinements is sufficient.

3.1. Optimization problem and model construction

The problem set-up is as follows. Let $\widetilde{f} : C \times \mathbb{R}^{n_y} \to \mathbb{R}$, $R : C \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_y}$ be continuously Fréchet differentiable functions. We assume that for every $x \in C$ there exists a unique solution $y = y(x) \in \mathbb{R}^{n_y}$ of

$$R(x, y(x)) = 0. \tag{26}$$

The system (26) is also referred to as the full order model (FOM) and in the applications we have in mind it is obtained from a large-scale high-fidelity approximation of a system of partial differential equations, i.e. $n_y \gg 1$. The objective function is

$$f(x) = \widetilde{f}(x, y(x)), \tag{27}$$

where $y = y(x) \in \mathbb{R}^{n_y}$ is the solution of (26). We refer to (27) as the FOM objective.

A model of (27) is computed using a Petrov-Galerkin ROM. That is, given $V, W \in \mathbb{R}^{n_y \times r}$, $r \ll n_y$, such that $y(x)$ is approximately contained in the range of V, the ROM of (26) is given by

$$W^T R(x, V\widehat{y}(x)) = 0. \tag{28}$$

We assume that $V, W \in \mathbb{R}^{n_y \times r}$ are constructed so that for every $x \in C$ there exists a unique solution $\widehat{y} = \widehat{y}(x) \in \mathbb{R}^r$ of (28). The model is given by

$$m(x) = \widetilde{f}(x, V\widehat{y}(x)), \tag{29}$$

where $\widehat{y} = \widehat{y}(x) \in \mathbb{R}^r$ is the solution of (28). We refer to (29) as the ROM objective.

We use the adjoint method (see, e.g. [26, Section 1.6]) to compute the gradient of both the FOM and ROM objectives. Let $\nabla_x \widetilde{f}(x, y) \in X$ and $\nabla_y \widetilde{f}(x, y) \in \mathbb{R}^{n_y}$ denote the partial gradients of $\widetilde{f}$, and let $R_x(x, y) \in L(X, \mathbb{R}^{n_y})$ and $R_y(x, y) \in \mathbb{R}^{n_y \times n_y}$ denote the partial Jacobians of R.

Given $x \in C$ and the corresponding solution $y(x) \in \mathbb{R}^{n_y}$ of (26), assume that $R_y(x, y(x))$ is invertible. The gradient of the FOM objective function (27) is

$$\nabla f(x) = R_x(x, y(x))^* p(x) + \nabla_x \widetilde{f}(x, y(x)), \tag{30}$$

where $p(x) \in \mathbb{R}^{n_y}$ solves the adjoint equation

$$R_y(x, y(x))^T p(x) = -\nabla_y \widetilde{f}(x, y(x)). \tag{31}$$

The gradient of the model m is computed analogously to the gradient $\nabla f(x)$. Given $x \in C$ and the corresponding solution $\widehat{y}(x) \in \mathbb{R}^r$ of (28), assume that $W^T R_y(x, V\widehat{y}(x))V$ is invertible. The gradient of the ROM objective (32) is given by

$$\nabla m(x) = R_x(x, V\widehat{y}(x))^* W\widehat{p}(x) + \nabla_x \widetilde{f}(x, V\widehat{y}(x)), \tag{32}$$

where $\widehat{p}(x) \in \mathbb{R}^r$ solves the ROM adjoint equation

$$V^T R_y(x, V\widehat{y}(x))^T W\widehat{p}(x) = -V^T \nabla_y \widetilde{f}(x, V\widehat{y}(x)). \tag{33}$$

Lemma 3.1: *Assume that for $x \in C$ the FOM (26) has a unique solution $y(x) \in \mathbb{R}^{n_y}$ and the ROM (28) has a unique solution $\widehat{y}(x) \in \mathbb{R}^r$. If $y(x) \in$ range(V), then*

$$f(x) = m(x).$$

If, in addition, $R_y(x, y(x))$ and $W^T R_y(x, V\widehat{y}(x))V$ are invertible and the solution $p(x)$ of (31) satisfies $p(x) \in$ range(W), then

$$\nabla f(x) = \nabla m(x).$$

Proof: If $y(x) \in \text{range}(V)$, then $y(x) = V\widetilde{y}(x)$ for some $\widetilde{y}(x) \in \mathbb{R}^r$ and $0 = R(x, y(x)) = R(x, V\widetilde{y}(x))$ implies $0 = W^T R(x, V\widetilde{y}(x))$. Since the ROM (28) has a unique solution, $\widehat{y}(x) = \widetilde{y}(x)$ and $y(x) = V\widehat{y}(x)$. Hence

$$f(x) = \widetilde{f}(x, y(x)) = \widetilde{f}(x, V\widehat{y}(x)) = m(x).$$

If, in addition, $p(x) \in \text{range}(W)$, then $y(x) = V\widehat{y}(x)$ and the invertibility of $R_y(x, y(x))$ implies that $p(x) = W\widehat{p}(x)$. Hence,

$$\nabla f(x) = R_x(x, y(x))^* p(x) + \nabla_x \widetilde{f}(x, y(x))$$

$$= R_x(x, V\widehat{y}(x))^* W\widehat{p}(x) + \nabla_x \widetilde{f}(x, V\widehat{y}(x)) = \nabla m(x). \qquad \blacksquare$$

Lemma 3.1 states basic results about the exactness of the ROM generated model and its gradient at a single point. In addition, the ROM literature provides bounds for the error $y(x) - V\widehat{y}(x)$ between the solutions of (26) and of (28). See, e.g. the papers and books [10,19,21,25,38,40,42,46]. These can be used to construct estimates (4) for the error between the objective function and its model. Similarly, one can construct bounds for the error between the solution $p(x)$ of the adjoint Equation (31) and the solution $W\widehat{p}(x)$ of the ROM adjoint (33), and use these to construct estimates for the error in the projected gradient (9), (10). Again, for several example problems, such objective function error and gradient error bounds are developed, e.g. in [27,39,45,51].

3.2. Model update

The setting is as in the previous Section 3.1. Now we consider the use of ROM objectives as models in Algorithm 3. The kth iteration of Algorithm 3 contains an inner iteration indexed by i. The model may be refined in the inner iteration and to keep track of these changes we use $m_{k,i}$ to denote the model in the ith inner iteration of the kth outer iteration. Similarly, we will use $s_{k,i}$, $\alpha_{k,i}$, and $x^C_{k,i}$ to denote the step, the step-size and the generalized Cauchy point in the ith inner iteration of the kth outer iteration.

In the kth iteration we have a current iterate $x_k \in C$. Assume that in the ith inner iteration of outer iteration k we have a ROM objective $m_{k,i}$. The ROM objective $m_{k,i}$ is computed with

$$V_{k,i}, W_{k,i} \in \mathbb{R}^{n_y \times r_{k,i}}, \quad r_{k,i} \ll n_y,$$

which satisfy the following conditions. Let $y(x_k) \in \mathbb{R}^{n_y}$ be the solution of (26) with x replaced by x_k, assume that $R_y(x_k, y(x_k))$ is invertible, and let $p(x_k) \in \mathbb{R}^{n_y}$ be the solution of (31) with x replaced by x_k. Assume that the ROM matrices are constructed so that

$$y(x_k) \in \operatorname{range}(V_{k,i}), \quad p(x_k) \in \operatorname{range}(W_{k,i}). \tag{34}$$

Furthermore, assume that $W^T_{k,i} R(x_k, V_{k,i}\widehat{y}_{k,i}) = 0$ has a unique solution $\widehat{y}_{k,i} \in \mathbb{R}^{r_{k,i}}$ and that $W^T_{k,i} R_y(x_k, V_{k,i}\widehat{y}_{k,i}) V_{k,i}$ is invertible. The ROM objective (29) computed with V, W replaced by $V_{k,i}$, $W_{k,i}$ is denoted by $m_{k,i}$. By (34) and Lemma 3.1,

$$f(x_k) = m_{k,i}(x_k), \quad \nabla f(x_k) = \nabla m_{k,i}(x_k). \tag{35}$$

If the ROM matrices satisfy (34), then the direction used to compute the trial generalized Cauchy points in outer iteration k is the same, only the step-sizes $\alpha_{k,i}$ vary.

We use the Backtracking Algorithm 1 to compute a step-size $\alpha_{k,i}$ and

$$x^C_{k,i} = \Pi_C(x_k - \alpha_{k,i} \nabla m_{k,i}(x_k)), \tag{36}$$

such that the generalized Cauchy point (36) satisfies the sufficient decrease condition (cf. (7))

$$m_{k,i}(x^C_{k,i}) \leq m_{k,i}(x_k) - \frac{c_1}{\alpha_{k,i}} \|x_k - x^C_{k,i}\|^2_X. \tag{37}$$

If the generalized Cauchy point (36) satisfies (23), then we use the model $m_k = m_{k,i}$ and $x_k^C = x_{k,i}^C$ to compute an approximate solution x_{k+1} of (22a) that satisfies (20). See Steps 12 and 23 of Algorithm 3.

If the generalized Cauchy point (36) does not satisfy (23), then we need to refine the model $m_{k,i}$. We do this by solving the FOM (26) with $x = x_{k,i}^C$ to compute $y(x_{k,i}^C) \in \mathbb{R}^{n_y}$. Then we update

$$V_{k,i+1} = \text{orth}\left([V_{k,i}, y(x_{k,i}^C)]\right), \tag{38a}$$

i.e, $V_{k,i+1}$ is a matrix with orthonormal columns such that range$(V_{k,i+1})$ = range$([V_{k,i}, y(x_{k,i}^C)])$, and we update $W_{k,i+1}$ such that

$$\text{range}(W_{k,i}) \subset \text{range}(W_{k,i+1}), \tag{38b}$$

$W_{k,i+1}^T R(x_k, V_{k,i+1}\widehat{y}_{k,i+1}) = 0$ has a unique solution $\widehat{y}_{k,i+1} \in \mathbb{R}^{r_{k,i+1}}$, and that $W_{k,i+1}^T R_y(x_k, V_{k,i+1}\widehat{y}_{k,i+1}) V_{k,i+1}$ is invertible. Then we set $i \leftarrow i+1$ and repeat the process.

The construction of the ROMs with properties (34) and (38a), and Lemma 3.1 imply the following result.

Lemma 3.2: *Under the assumptions made in this subsection,*

$$y(x_{k,j}^C) \in \mathit{range}(V_{k,j+1}) \subset \mathit{range}(V_{k,i}) \quad \textit{for all } 0 \le j < i, \tag{39a}$$

$$y(x_k) \in \mathit{range}(V_{k,j}) \subset \mathit{range}(V_{k,i}) \quad \textit{for all } 0 \le j \le i, \tag{39b}$$

$$p(x_k) \in \mathit{range}(W_{k,j}) \subset \mathit{range}(W_{k,i}) \quad \textit{for all } 0 \le j \le i, \tag{39c}$$

and

$$f(x_k) = m_{k,i}(x_k), \quad \nabla f(x_k) = \nabla m_{k,i}(x_k) \quad \textit{for all } 0 \le i, \tag{40a}$$

$$f(x_{k,j}^C) = m_{k,i}(x_{k,j}^C), \quad \textit{for all } 0 \le j < i. \tag{40b}$$

Proof: The inclusions (39a) follow from (34) and (38a). The equalities (40a) follow from (39a) and Lemma 3.1. ■

Since the search direction is the negative gradient, (40a) implies that the search direction remains the same throughout iteration k. Only the step-size $\alpha_{k,i}$ changes. In outer iteration k, at the trial generalized Cauchy points $x_{k,j}^C$ the current and new models $m_{k,i}$, $i \ge j$, agree with the function f. Moreover, the gradients of $m_{k,i}$, $i \ge j$, agree with the gradient of f. Thus, the sufficient decrease condition (37) for the model $m_{k,i}$ effectively becomes a sufficient decrease condition for the function f. This will allow us to prove that a finite number of model refinements are sufficient to compute a generalized Cauchy point $x_{k,i}^C$ that satisfies (23).

First we examine sufficient decrease condition for f,

$$f(x_k^C) \le f(x_k) - \frac{c_1}{\alpha_k}\|x_k - x_k^C\|_X^2 \tag{41}$$

where $0 < c_1 < 1$ cf. (7). We use the Backtracking Algorithm 1 applied to f and (41) to compute α_k. Lemma 2.1 applied with f instead of m_k guarantee the existence of $\alpha_k > 0$ such that (41) holds. Lemma 2.2 applied with f implies the step-size is uniformly bounded away from zero, $\alpha_k \geq \alpha_{\min} := 2\beta_1(1-c_1)/L_f > 0$ for all k where L_f is the Lipshitz constant for f. This implies that the number of iterations needed by the Backtracking Algorithm 1 to find the step-size is uniformly bounded.

For the next result, we need that the initial step size $\alpha_k^{(0)}$ in the Backtracking Algorithm 1 can be chosen independently of k.

Theorem 3.3: *If the assumptions on the model refinement made in this subsection hold, if the assumptions on f in Lemmas 2.1 and 2.2 hold, and if $\alpha_{k,i}$ with (36), (37) is computed using the Backtracking Algorithm 1 with initial step-sizes $\alpha_{k,0}^{(0)} := \alpha^{(0)}$ and $\alpha_{k,i}^{(0)} := \alpha_{k,i-1}$, $i > 0$, then there exists an integer $i_{\max}$ independent of k such that at most $i_{\max}$ model refinements are needed in the kth iteration of Algorithm 3 to compute a generalized Cauchy point that satisfies (23).*

Proof: Under the assumptions on f in Lemmas 2.1 and 2.2, the Backtracking Algorithm 1 with initial step-size $\alpha^{(0)}$ applied to f and (41) requires at most $\ell_{\max}$ iterations to find a step-size α_k that satisfies (41), where $\ell_{\max}$ is the smallest integer with $\beta_2^{\ell_{\max}}\alpha^{(0)} \leq \alpha_{\min}$.

In the ith inner iteration of the kth outer iteration of Algorithm 3 the model $m_{k,i}$ is used to compute a step-size $\alpha_{k,i}$, and a corresponding generalized Cauchy point $x_{k,i}^C = x_k + \alpha_{k,i}s_k$. By choice of the initial $\alpha_{k,i}^{(0)} := \alpha_{k,i-1}$, $i > 0$, the step-sizes satisfy $\alpha_{k,i} \leq \alpha_{k,i-1}$.

If $\alpha_{k,i} = \alpha_{k,j}$ for some $j < i$ then $x_{k,i}^C = x_{k,j}^C$ and, by Lemma 3.2,

$$f(x_{k,i}) = m_{k,i}(x_{k,i}^C), \quad f(x_k) = m_{k,i}(x_k)$$

which implies

$$0 = e_k(x_{k,i}^C) < a_2(1-a_1)\left(m_{k,i}(x_k) - m_{k,i}(x_{k,i}^C)\right), \tag{42}$$

i.e. the generalized Cauchy point $x_{k,i}^C = x_{k,j}^C$ satisfies (23) and we are done.

This implies that Cauchy points, i.e. corresponding step-sizes of unaccepted models are strictly decreasing. Moreover, by the Backtracking Algorithm 1 and our choice of initial step-size $\alpha_{k,i}^{(0)}$, step-sizes of unaccepted models decrease by at least $\beta_2 \in (0,1)$, $\alpha_{k,i} \leq \beta_2\alpha_{k,i-1}$, $i > 0$. Thus if the model $m_{k,i}$ did not generate a generalized Cauchy point $x_{k,i}^C$ that satisfies (23), then

$$\alpha_{k,i} \leq \beta_2\alpha_{k,i-1} \leq \beta_2^i\alpha_{k,0} \leq \beta_2^i\alpha^{(0)}$$

and, by Lemma 2.2,

$$f(x_{k,i}^C) = m_{k,i+1}(x_{k,i}^C), \quad f(x_k) = m_{k,i+1}(x_k), \quad \nabla f(x_k) = \nabla m_{k,i+1}(x_k).$$

After at most $i = \ell_{\max}$ iterations, $\alpha_{k,i} \leq \beta_2^i\alpha^{(0)}$ and $\alpha_{k,i}$, $x_{k,i}^C$ satisfy (41), the condition (7) with m_k replaced by $m_{k,i+1}$ and x_k^C replaced by $x_{k,i}^C$, as well as $0 = e_{k,i+1}(x_{k,i}^C) < a_2(1-a_1)(m_{k,i+1}(x_k) - m_{k,i+1}(x_{k,i}^C))$.

Thus at most $\ell_{\max} + 2$ models are generated in the kth iteration of Algorithm 3 to compute a generalized Cauchy point that satisfies (23). ■

Corollary 3.4: *Let the assumptions of Theorems 2.5 and 3.3 hold. The iterates calculated by the Line-Search Algorithm 3 with models given by the ROM objectives computed as in Section 3.2 satisfy* $\lim_{k\to\infty} \|x_k - \Pi_C(x_k - \alpha^{(0)}\nabla f(x_k))\|_X = 0$.

Proof: The result follows immediately from Theorems 2.5 and 3.3, since a finite and bounded number of model refinements are sufficient in the kth iteration of Algorithm 3 to compute a generalized Cauchy point that satisfies (23). ■

4. Numerical experiments

We apply Algorithm 3 and the Yue and Meerbergen [51] algorithm to model problems from [27,39], and [45]. Our Algorithm 3 will also be referred to as 'New', whereas the Yue and Meerbergen [51] algorithm with detailed algorithm specifications given in [20, Section 2.6] will be referred to as 'YM'.

Although some problems have box constraints, these turn out to be inactive most of the time and therefore the applied algorithms essentially consider $C = \mathbb{R}^n$. In our numerics, we use a line-search Newton Conjugate Gradient (NCG) method or a limited-memory BFGS (LBFGS) method (instead of the negative gradient) to compute a generalized Cauchy point (6) and to compute an approximate solution of subproblem (22a). We apply one step of NCG or LBFGS to find the generalized Cauchy point (see e.g. [36, pp. 168-170, Section 6.1, 7.2] and [22, Section 5.1] for implementation details for these algorithms). To approximately solve the subminimization problem (22a), NCG or LBFGS is applied with initial point x_k^C until $\min\{\|x_{k,i} - \Pi_C(x_{k,i} - \nabla m_k(x_{k,i}))\|_X, \|\nabla m_k(x_{k,i})\|_X\} <$ tol (where tol is the same stopping tolerance as in the outer iteration of Algorithm 3) or the constraint (22ab) is violated. In our experiment, the iteration to solve (22a) terminated because the constraint (22ab) was violated only in the first one of two outer iterations. Thus, in most outer iterations, the termination criteria $\min\{\|x_{k,i} - \Pi_C(x_{k,i} - \nabla m_k(x_{k,i}))\|_X, \|\nabla m_k(x_{k,i})\|_X\} <$ tol was reached. This may 'oversolve' the problem in early iterations, but since the models are computationally less expensive we use the model as long as it is trusted (quantified by the constraints (22ab)). For both NCG and BFGS algorithms, a maximum of 20 suboptimization iterations were allowed. The relative residual stopping tolerance for CG in is set as $\eta_{k,i} = \min\{\|\nabla m_k(x_{k,i})\|_X^2, 0.01\|\nabla m_k(x_{k,i})\|_X\}$. The number of BFGS updates is limited to $L = 20$. Since at most 20 sub-iterations were allowed in our experiments, the LBFGS is equal to BFGS.

In all examples we use ROMs, as outlined in Section 3, but we use Galerkin ROMs, i.e. $V = W \in \mathbb{R}^{n_y \times r}$. Details of the construction and updates of the ROM basis V are problem specific and will be discussed for each example.

We set the stopping tol from Algorithm 3 to tol $= 10^{-4}$ for the bypass problem in Section 4.2 and the airfoil design problem in Section 4.3; for the fin problem we use tol $= 10^{-6}$. Our implementation of the Backtracking Line-Search Algorithm 1 follows [14, Section 6.3.2] with parameters $\beta_1 = 0.1$, $\beta_2 = 0.5$ and initial stepsize $\alpha_k^0 = 1$. Problem parameters in Algorithm 3 were set to $a_1 = 0.7$ and $a_2 = 0.95$. The initial function value tolerance is $\tau_0^f = 10^{-2}$. The relative gradient tolerance is fixed at $\tau_k^g = 0.5$, $k \geq 0$. The parameters for the additional backtrack of the generalized Cauchy point backtracking are set to $\widetilde{\beta}_1 = 0.1$ and $\widetilde{\beta}_2 = 0.9$. To select $\alpha_k \in [\widetilde{\beta}_1\alpha_k^{\text{old}}, \widetilde{\beta}_2\alpha_k^{\text{old}}]$ we use a quadratic model of the error function.

We use the implementation of the Yue-Meerbergen algorithm from [20, Alg. 2.6.2]. We use the parameters above along with Cauchy backtracking parameter 0.5, $\epsilon_L = 0.1$ and ϵ_L reduction parameter 0.9. Computation of the generalized Cauchy point uses our implementation of the Backtracking Line-Search Algorithm 1 following [14, Section 6.3.2] with parameters $\beta_1 = 0.1$, $\beta_2 = 0.5$, initial stepsize $\alpha_k^0 = 1$, and the additional truncation criteria that $\alpha_k = \alpha_k^{(\ell)}$ also satisfies $e_k(x_k + \alpha_k^{(\ell)} s_k) \le \epsilon_L |m_k(x_k + \alpha_k^{(\ell)} s_k)|$. To approximately solve (24a) we apply line-search NCG / LBFGS iterations starting at the generalized Cauchy point x_k^C with the same stopping criteria as the subproblem from Algorithm 3.

In all examples of this section $X = \mathbb{R}^n$ with the standard Euclidean inner product. If the Line-Search Algorithm 3 with Inexact Function Information and NCG or LBFGS is used with exact functions, $m_k = f$ for all k, then the algorithm is identical to a standard line-search NCG or line-search LBFGS algorithm. For the following examples, we will compare the Line-Search Algorithm 3 with Inexact Function Information with the corresponding exact versions.

4.1. Thermal fin example

The first example problem is a heat conduction optimization problem. This problem is described and applied in Section 5.1.1 of [39] and is referenced and applied in Section 5.2 of [27]. The goal is to determine heat conductivity denoted by $\kappa_i \in [0.1, 10]$, $i = 0, \dots, 4$, for different regions of a thermal find, as well as the Biot number $\mathrm{Bi} \in [0.01, 1]$, a nondimensional heat transfer coefficient, to that the average temperature at the bottom of the fin is close to a desired temperature $\widehat{T}$.

Discretizing the problem using piecewise linear finite elements leads to the optimization problem

$$\min_{\mu \in C} f(\mu) \overset{\text{def}}{=} \frac{1}{2} |\mathbf{b}^T \mathbf{y}(\mu) - \widehat{T}|^2 + \frac{\omega}{2} \|\mu - \widehat{\mu}\|_2^2, \tag{43}$$

where $\mu = [\kappa_0, \dots \kappa_4, \mathrm{Bi}] \in [0.1, 10]^4 \times [0.01, 1] =: C$, $X = \mathbb{R}^5$, and $\mathbf{y}(\mu) \in \mathbb{R}^{n_y}$ solves the FOM $\mathbf{A}(\mu)\mathbf{y}(\mu) = \mathbf{b}$. For $\mu \in C$ the matrix $\mathbf{A}(\mu) = \sum_{i=0}^{5} \mu_i \mathbf{A}_i$ is symmetric positive definite. The desired temperature distribution is computed as $\widehat{T} = \mathbf{b}^T \mathbf{y}(\widehat{\mu})$ using the optimal parameter $\widehat{\mu} = [1, 1, 1, 1, 1, 0.1]$. In our computations, using the finite element mesh provided by [39], the FOM dimension is $n_y = 17,899$.

We use a Galerkin ROM, and construction of the ROM matrix $\mathbf{V}$ is discussed below. Output error estimates for linear ROMs for this specific problem are given in [20,27,33,39]. We use these bounds to construct objective function error bounds (4), (5) and projected gradient error estimates (9), (10). These bounds are detailed in [20, Section 5.3.2], and are based on [39].

The reduced basis at the outer iterate k of the algorithm is computed from FOM states and adjoints computed at previous outer iterates and the current outer iterate. Actually, in this example, the FOM state equation and adjoint equation only differ by a scaling of the right hand side and therefore the states $\mathbf{y}(\mu)$ and adjoints $\mathbf{p}(\mu)$ differ only by a scalar. Thus using state information is sufficient in this example. Specifically, at the beginning of outer iteration k, states $\mathbf{y}(\mu_k), \dots, \mathbf{y}(\mu_0)$ have been computed and we compute the ROM using

$$\mathbf{V}_{k,0} = \mathrm{orth}\left(\left[\mathbf{y}(\mu_k), \dots, \mathbf{y}(\mu_0)\right]\right).$$

Table 1. Comparison of exact line-search algorithm, our implementation of the Yue-Meerbergen algorithm ('ROM YM') and our Line-Search Algorithm 3 ('ROM New') applied to the fin problem.

Algorithm	Iters	FOM	ROM	Rec.	C.V.
LBFGS FOM	9	11	–	–	–
LBFGS ROM New	4	5	30	0	2
LBFGS ROM YM	6	7	97	0	5

In particular, the ROM state and adjoint equations in the kth iteration are of size $r = k + 1$. In the experiments presented in this section, the kth model m_k never needed to be refined. If a refinement was needed, a state snapshot at the proposed Cauchy point would have been added, i.e. $\mathbf{V}_{k,i} = \text{orth}([\mathbf{V}_{k,i-1}\ \mathbf{y}(\mu^C_{k,i-1})])$.

Table 1 and Figure 2 summarize the performance of our Algorithm 3 compared to the algorithm using only exact function evaluations and our implementation of the Yue-Meerbergen algorithm. Algorithm 3 uses only 4 outer iterations and requires less than half the FOM solves required by the FOM only algorithm (see columns 'Iters' and 'FOM'). Since each outer iteration of Algorithm 3 requires an approximate solution of an optimization subproblem, the number of ROM evaluations is larger than the corresponding FOM solves (see column 'ROM'). However, the ROM state systems have at most dimension $r = 5$, while the FOM system is of dimension $n_y = 17,899$. In this case no recomputations of the model (line 8 in Algorithm 3) were required (column 'Rec.'). In the first two outer iterations, the iteration to approximately solve (22a) was terminated because the constraint (22ab) was violated ('C.V.'). In later iterations, the constraint (22ab) was never active, and accurate approximations to the solution of the minimization sub-problem (22a) could be computed using unconstrained optimization approaches. In Figure 2 the beginning of an outer iteration is indicated by a large circle, this is when a model update is used, i.e. FOM is solved. Jumps of ROM objective function values at the beginning of an outer iteration result from model recomputation. The Yue-Meerbergen algorithm requires slightly more FOM evaluations, slightly more outer iterations and slightly more total sub-optimization iterations (see Figure 2). Algorithm 3 required fewer ROM evaluations for two primary reasons. The first is that the right hand side of the constraint in the Yue-Meerbergen subproblem (24a) requires evaluation of the model m_k, i.e, a ROM evaluation (in addition to the evaluation of the error e_k). The second is that the constraint condition (24b) is more restrictive because in this example m_k becomes small (the true minimizer satisfies $f(x_*) = 0$), leading to more backtracking during the computation of the Cauchy point and the sub-optimization. The use of error estimators instead of exact error did not adversely impact the convergence of the algorithm.

We note that although the same example is used in [27] and in [39] with algorithms derived from [51], a direct comparisons of results is not possible. The paper [27] reports only timing information and uses a larger discretization size. The data reported in [39] is primarily concerned with the quality of the error estimates, and only aggregate data is reported for number of FOM evaluations. The mean number of FOM evaluations reported in [39] is about the same as the number of FOM evaluations reported in Table 1.

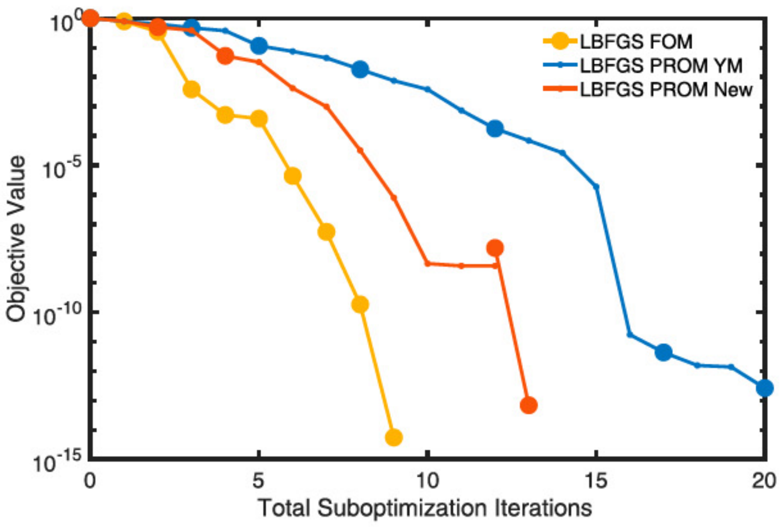

Figure 2. Iteration history for Yue-Meerbergen (YM) algorithm (blue) and new Algorithm 3 (red) applied to the thermal fin example (43). Small dots indicate subiterations. Large dots indicate beginning of outer iteration after model computation. Jumps in function values at beginning of an outer iteration are due to change in model.

4.2. *Optimal shape design of bypass*

This example is a shape optimization problem involving an aorta-coronary bypass. The goal is to determine the shape of domain whose boundary is represented by a Bezier curve with control points $\mu \in \mathbb{R}^8$ so that vorticity in a region of the domain is minimized. The formulation of this problem as well as the code for continuous finite element discretization and ROM generation was provided by Dr. Zahr and his student Tianshu Wen and is described in detail in [45]. We do not incorporate hyperreduction from [45].

4.2.1. *Problem formulation*

We follow [45] and denote the optimization variable by μ instead of x. The optimization problem is

$$\min_{\mu \in [-0.4,0.4]^8} f_0(\mathbf{v}(\mu), \mu) + \frac{\omega}{2}\|\mu\|_2^2, \tag{44}$$

where for given $\mu \in \mathbb{R}^8$ the discretized velocity $\mathbf{v}(\mu)$ together with the discretized pressure are computed as the solution $\mathbf{y}(\mu)$ of the discretized state equations $\mathbf{R}(\mathbf{y}(\mu), \mu) = \mathbf{0}$. The first subblock in $\mathbf{y}(\mu) \in \mathbb{R}^{n_y}$ is the discretized velocity $\mathbf{v}(\mu)$. The FOM state dimension is $n_y = 7,522$.

4.2.2. *Model construction and refinement*

The construction of the reduced basis $\mathbf{V}_{k,i}$ at step k of the minimization problem is nearly identical to the routine given Section 3.2. The only major difference is the possible inclusion of sensitivities of the state $\mathbf{y}(\mu)$ with respect to the parameters μ, which are given by

$$\mathbf{s}(\mu) := -\mathbf{R}_{\mathbf{y}}(\mathbf{y}(\mu), \mu)^{-1}\mathbf{R}_{\mu}(\mathbf{y}(\mu), \mu)^T. \tag{45}$$

Table 2. Performance of Algorithm 3 ('ROM New') with various ROM construction strategies ('Type 1-4', see Section 4.2.2) and corresponding exact line-search algorithm ('FOM') applied to the bypass problem with Reynolds number Re = 50.

Algorithm	Iters	FOM	ROM	Rec.	C.V.	ROM size
LBFGS FOM	11	13	–	–	–	–
LBFGS ROM New, Type 1	8	12	79	3	4	2/24
LBFGS ROM New, Type 2	9	10	78	0	3	10/28
LBFGS ROM New, Type 3	5	6	52	0	2	10/20
LBFGS ROM New, Type 4	4	5	53	0	1	10/50
LBFGS ROM YM, Type 2	8	9	159	1	3	10/26
LBFGS ROM YM, Type 3	5	6	119	1	3	10/20
LBFGS ROM YM, Type 4	4	5	100	0	3	10/50

Note that $\mathbf{R}_\mathbf{y}$ and $\mathbf{R}_\mu$ are both computed already when solving for the adjoint (31) and gradient (30), respectively. In this section, we consider four different approaches for constructing the ROM models, we will denote them Types 1-4. Below, $\mathbf{y}(\mu) \in \mathbb{R}^{n_y}$ is the solution of the discretized state equations $\mathbf{R}(\mathbf{y}(\mu), \mu) = \mathbf{0}$, $\mathbf{s}(\mu) \in \mathbb{R}^{n_y \times 8}$ are the corresponding sensitivities (45), and $\mathbf{p}(\mu) \in \mathbb{R}^{n_y}$ is the solution of the discrete adjoint equation. The initial reduced basis is thus given by

$$\begin{aligned} \mathbf{V}_0 &= \text{orth}\left(\mathbf{y}(\mu_0), \mathbf{p}(\mu_0)\right) \quad \text{Type 1,} \\ \mathbf{V}_0 &= \text{orth}\left(\mathbf{y}(\mu_0), \mathbf{p}(\mu_0), \mathbf{s}(\mu_0)\right) \quad \text{Type 2 through Type 4.} \end{aligned}$$

Note that Type 1 does not include the sensitivities in the initial ROM basis while Types 2-4 do. The paper [45] uses Type 1 and Type 2.

Model updates and refinement proceed similarly as described in Section 3.2 with the possible inclusion of updated sensitivity information, the initial model at each iteration k introduces the FOM state and adjoint

$$\begin{aligned} \mathbf{V}_{k,0} &= \text{orth}\left(\mathbf{V}_{k-1}, \mathbf{y}(\mu_k), \mathbf{p}(\mu_k)\right) \quad \text{Type 1 and Type 2,} \\ \mathbf{V}_{k,0} &= \text{orth}\left(\mathbf{y}(\mu_0), \mathbf{p}(\mu_0), \dots, \mathbf{y}(\mu_k), \mathbf{p}(\mu_k), \mathbf{s}(\mu_k)\right) \quad \text{Type 3,} \\ \mathbf{V}_{k,0} &= \text{orth}\left(\mathbf{V}_{k-1}, \mathbf{y}(\mu_k), \mathbf{p}(\mu_k), \mathbf{s}(\mu_k)\right) \quad \text{Type 4.} \end{aligned}$$

Note that Type 1 still includes no sensitivity information and Type 2 only retains the sensitivity information from the initial iterate μ_0. Type 3 replaces the sensitivity information from the previous iteration with $\mathbf{s}(\mu_k)$, i.e. includes the sensitivities at iteration k only, while Type 4 includes all sensitivities $\mathbf{s}(\mu_j)$ for $j = 0, \dots, k$. Model refinement proceeds the same for every method implemented, refinement includes the FOM state and adjoint from the previous candidate Cauchy point

$$\mathbf{V}_{k,i} = \text{orth}\left(\mathbf{V}_{k,i-1}, \mathbf{y}(\mu^C_{k,i-1}), \mathbf{p}(\mu^C_{k,i-1})\right).$$

4.2.3. *Results*

Table 2 summarizes the performance of our Algorithm 3 using Reynolds number Re = 50 and various ROM construction strategies compared to the algorithm using only exact function evaluations.

Table 3. Performance of Algorithm 3 ('ROM New') with various ROM construction strategies ('Type 1-4', see Section 4.2.2) and corresponding exact line-search algorithm ('FOM') applied to the bypass problem with Reynolds number Re = 500.

Algorithm	Iters	FOM	ROM	Rec.	C.V.	ROM size
LBFGS FOM	15	18	–	–	–	
LBFGS ROM New, Type 2	17	23	217	5	13	10/54
LBFGS ROM New, Type 3	8	11	109	2	4	10/30
LBFGS ROM New, Type 4	10	18	293	7	6	10/124
LBFGS ROM YM, Type 2	34	36	1254	18	5	10/54
LBFGS ROM YM, Type 3	7	9	276	0	3	10/26
LBFGS ROM YM, Type 4	28*	29	1137	21	2	10/-

Note: Starred iterations indicate failed convergence.

In all cases, Algorithm 3 converged to the desired gradient tolerance. In this case, some of the ROM construction strategies used similar numbers of FOM evaluations and iterations to the algorithm with exact function evaluations. Thus, in case where the model m_k is only a good approximation near the current point x_k, our Algorithm 3 still converges but offers little to no performance gains over the more straightforward approach that only uses FOMs. Table 2 highlights the importance of choosing a good ROM when implementing Algorithm 3. As the quality of the ROM increases (recall that Types 2-4 include sensitivities (45) while Types 3 and 4 continually update sensitivities), the number of iterations and the number of FOM evaluations decrease. With a ROM of Type 3 or 4 computational savings of more that 50% (in terms of outer iterations and FOM evaluations) were achieved. The last column ('ROM size') in Table 2 shows the ROM size in the initial and in the final iteration. The ROMs are substantially smaller than the FOM, which is of size $n_y = 7,522$. Note that ROM sizes increase linearly throughout the algorithm, refinements increase the size by 2 while new iterations increase the size by 2 (Types 1-3) or by 10 (Type 4), see Section 4.2.2. Comparison between our Algorithm 3 and the Yue-Meerbergen algorithm shows that the number of iterations and FOM evaluations are very similar while ROM evaluations for Yue and Meerbergen are about double the evaluations used in our algorithm.

Table 3 repeats the computations using Reynolds number Re $=$ 500. For Reynolds number Re $=$ 500, the situation is a bit more complex.

In all cases our Algorithm 3 converges, and the better the ROM, the more computationally efficient Algorithm 3 becomes. In the best reported case nearly 50% computational savings (in terms of outer iterations and FOM evaluations) were achieved. ROMs of Type 2 performed more poorly, as they did in the Re $=$ 50 case. ROMs of Type 3 led to substantial improvements. In this case, the Type 3 ROM lead to better performance than the Type 4 ROM. The reason seems to be the Newton-type iterative solver for $\mathbf{R}(\mathbf{y}(\mu), \mu) = \mathbf{0}$ and the corresponding ROM solver. For Re $=$ 500 these equations become more difficult to solve. In several instances the Newton-type iteration to solve the ROM at trial point $\mu_{k,i} + \alpha_{k,i}s_{k,i}$ did not converge to the required residual tolerance. In this case we set $m_{k,i}(\mu_{k,i} + \alpha_{k,i}s_{k,i}) = \infty$ and reduce the step size $\alpha_{k,i}$. This leads to smaller steps, smaller right hand sides in the constraint (22b) of the subproblem to compute μ_{k+1}, smaller tolerances for the new model (Step 24 in Algorithm 3), and overall slower progress. This is why the iteration for approximately solving (22a) terminated more often due to constraint violation ('C.V.'). This also led to more Cauchy step re-computations ('Rec.') and

FOM evaluations. This could be addressed by an improved solution strategy for solving the ROM. The most successful version of Algorithm 3 with ROMs of Type 3 uses nearly half the FOM solves and iterations required by the FOM only algorithm. Again size of the ROM state systems are reported in the rightmost column ('ROM size'); the first value is the initial ROM size and the second value is the final size. For all but ROMs of Type 4, the ROM sizes are less than 1% of the size of the FOM, which has state dimension $n_y = 7,522$. For both the Re = 50 and Re = 500 cases, our iteration to solve the minimization sub-problem (22a) was terminated in the first few outer iterations because the constraint (22ab) was violated ('C.V.'). In later iterations, the constraint (22ab) was never active, and, as in the Termal Fin Example 4.1, accurate approximations to the solution of the minimization sub-problem (22a) could be computed using unconstrained optimization approaches.

Both algorithms sometimes require tight error tolerances (see Algorithm 3, Line 24 and [20, Assumptions 2.6.5]), especially when the current iterate μ_k, is close to a minimizer. To construct ROMs with sufficiently low model error tolerances at μ_k, the underlying nonlinear simulations $\mathbf{R}(\mathbf{y}(\mu), \mu) = \mathbf{0}$ to compute the FOM state (needed for ROM model updates/refinements) and $\mathbf{V}^T\mathbf{R}(\mathbf{V}\widehat{\mathbf{y}}(\mu), \mu) = \mathbf{0}$ to compute the ROM state (needed for ROM model evaluations) need to be carried out with sufficiently high precision. This can increase computational demand and in some cases the Newton-type algorithm used to solve the nonlinear systems failed to converge. In particular, for the Re = 500 example, these strict state solve requirements were not always satisfiable and led to early termination of the iteration using certain model refinement strategies. Lower gradient tolerances to stop the algorithms would avoid this issue.

In the Re = 500 case, for Type 3, the YM algorithm performs slightly better in terms of iterations and FOM evaluations but requires about 3 times the ROM computations. For Types 2 and 4, the YM algorithm requires significantly more iterations, FOM, and ROM evaluations; in addition, the Type 4 algorithm failed to converge to the given tolerance because the ROM size became too large and the ROM state computations stopped converging. As before, the worse performance of Type 4 comes from the failure of the Newton-type iterative solver to solve the ROM state equation.

4.3. Inverse design of airfoil

In this example, we seek to recover the RAE2822 airfoil shape from its flow field. The formulation of this problem as well as the code for the Discontinuous Galerkin (DG) discretization and ROM was also provided by Dr. Zahr and his student Tianshu Wen and is described in more detail in [45]. We do not incorporate the hyperreduction [45].

Again, we follow [45] and denote the optimization variable by μ instead of x. The optimization problem is

$$\min_{\mu\in\mathbb{R}^{18}} \frac{1}{2}(\mathbf{y}(\mu) - \mathbf{y}^*)^T\mathbf{M}(\mathbf{y}(\mu) - \mathbf{y}^*), \qquad (46)$$

where $\mathbf{M}$ is a DG mass matrix and, for given $\mu \in \mathbb{R}^{18}$, the discretized state $\mathbf{y}(\mu) \in \mathbb{R}^{n_y}$ is computed as the solution of the discretized state equations $\mathbf{R}(\mathbf{y}(\mu), \mu) = \mathbf{0}$. In this example the FOM size is $n_y = 17,568$.

Model construction and refinement proceeds exactly as described in Section 4.2.2.

Table 4. Performance of Algorithm 3 ('ROM New') with various ROM construction strategies ('Type 2-4', see Section 4.2.2) and corresponding exact line-search algorithm ('FOM') applied to the airfoil shape optimization problem.

Algorithm	Iters	FOM	ROM	Rec.	C.V.	ROM size
LBFGS FOM	29	142	–	–	–	–
LBFGS ROM New, Type 2	8	9	384	1	4	20/36
LBFGS ROM New, Type 3	4	5	268	0	0	20/28
LBFGS ROM New, Type 4	4	5	268	0	0	20/100
LBFGS ROM YM, Type 2	21*	21	1286	0	23	20/60
LBFGS ROM YM, Type 3	9	10	605	0	7	20/38
LBFGS ROM YM, Type 4	11	12	814	0	9	20/240

Note: Starred iterations indicate failed convergence.

Table 4 summarizes the performance of our Algorithm 3 using various ROM construction approaches compared to the algorithm using only exact function evaluations and compared to the Yue-Meerbergen algorithm.

All ROM construction approaches led to much fewer FOM evaluations and overall iterations than the FOM only algorithm. We have excluded Type 1 construction given its relatively poor performance compared to the other ROM construction approaches for the bypass example (see Section 4.2.3). The Type 2 approach used about 3 times fewer iterations and about 16 times fewer FOM evaluations than the FOM only algorithm. The other two approach traced the same optimization iteration trajectories, so they had identical iteration and evaluation counts. They required about 7 times fewer iterations and almost 30 times fewer FOM evaluations compared to the FOM only algorithm. Only the Type 2 approach required any refinement and only when the Type 2 approach is used, did our iteration to solve the minimization sub-problem (22a) terminate because the constraint (22ab) was violated ('C.V.'). This happened in the early iterations. As in the previous two example, accurate approximations to the solution of the minimization sub-problem (22a) could be computed using unconstrained optimization approaches.

The number of iterations and FOM evaluations are significantly higher when using Type 2 for the Yue and Meerbergen algorithm algorithm compared to the new algorithm. For Types 3 and 4, the Yue-Meerbergen algorithm required roughly double the number of iterations and FOM evaluations and two to four times the number of ROM evaluations. These increases appear to be due to two different factors. The first is that for Type 2, the algorithm was not able to converge because the iteration to solve the nonlinear FOM equation to compute the FOM state solve did not converge, leading to the early termination of the optimization algorithm. Second, although iterations were never rejected, the constraint (24b) was violated frequently during the iteration which led to minimal progress on many iterations. This frequent constraint violation appears to stem from combination of factors. The first is the poor scaling of the initial Hessian approximation leading to large initial step sizes outside of the constraint region (see below for a more complete discussion). The second is the relative strictness of the of the constraint (24b); since the optimal value is at $f(x^*) = 0$, the constraint becomes very restrictive as the optimal value is approached. The slightly worse performance of Type 4 compared to Type 3 for YM is related to the same phenomena found in the Bypass Re $= 500$ example (see

Table 3). For some larger ROM sizes the iterative method used to solve the nonlinear system to compute the ROM state fails to converge leading to more backtracking and slower performance.

The objective function in this example does not contain a parameter penalty $\omega\|\mu\|_2^2$ with $\omega > 0$. Therefore we use $H_{k,0} = I$ as the initial BFGS matrix. In earlier (inner) iterations this leads to less well scaled descent directions $s_{k,i}$ and more backtracks in Backtracking Line-Search Algorithm 1. Therefore, we observe a larger number of FOM evaluations compared to the total number of iterations in the FOM only algorithm, and a larger number of ROM evaluations in Algorithm 3. However, note that in the FOM only algorithm the less good $H_{k,0} = I$ is absorbed by computationally expensive FOMs resulting from additional backtracks in the line seaerch. In our Algorithm 3, the less good $H_{k,0} = I$ is absorbed by computationally inexpensive ROMs. Algorithm 3 with all ROM construction approaches led to huge computational savings measured in outer iterations and FOM computations.

5. Conclusions

We have introduced a line-search algorithm that uses objective function models with tunable accuracy for the solution of unconstrained optimization problems. The algorithm was motivated by an algorithm developed in [51]. Our algorithm leverages efficient approximate models with error bounds and tunable accuracy to reduce the number of expensive objective evaluations. We presented general convergence results for inexact line-search algorithms and provided the details of an implementable algorithm. Specifically, we proved that our algorithm has the same first-order global convergence properties as standard line-search methods, provided the objective function models meet the required accuracy requirements. Our algorithm only uses these models and corresponding error functions but never directly accesses the original objective function. The accuracy requirement of the model are adjusted to the progress of the optimization algorithm. In addition, we proved that for a large class of commonly used model functions based on ROMs, the constructed models meet the accuracy requirements specified by our algorithm. Our algorithm is arguably simpler than the one proposed in [51], and when ROMs are used, we have a complete convergence proof.

Our algorithm was applied to numerical examples with implicit PDE constraints, and converged in all cases and with a range of ROMs used. We demonstrated that with appropriately chosen models the algorithm demonstrates significant reductions both in the number of iterations and in the number of expensive exact objective solves required to reach an optimal solution, when compared to optimization strategies using solely exact objective information. In numerical examples our algorithm also performed better than the previously proposed algorithm from [51].

There are a number of possible extensions that are under investigation. Currently, bounds for the error between true and model objective function and for the error between their gradients are needed. For implementations, an extension that allows the use of asymptotic bounds is desirable. Further improvements to the numerical results may be found by using more refined techniques for solving the constrained minimization subproblems (22a) if more information, e.g. derivatives, for the error functions are available. However, the benefits of a more accurate solution of the optimization sub-problem has to be balanced with the increased cost. In our numerical examples of this paper, the constraint (22ab) was only active in the first few iterations. In the final iterations, (22ab) was never active, which indicates that an essentially unconstrained optimization iteration, as the one described at the beginning of Section 4 may be sufficiently efficient overall.

Finally, it seems possible to extend our approach to problems with nonlinear constraints. For example, the paper [24] already discusses a line-search ℓ_1 penalty SQP algorithm that allows inexact objective function and objective gradient evaluations but exact constraint function and constraint Jacobian. In [24], the objective function and gradient error are reduced at a specified constant rate. Extending the approach in this paper to the constrained case would adjust the objective function and gradient error dynamically, and provide a mechanism to explore the current objective function model over a larger region in the optimization variable space.

Acknowledgments

Thanks to Dr. Qian and her collaborators [39], and to Dr. Keil and his collaborators [27] for providing their FEM mesh and geometry for the problem in Section 4.1, and to Dr. Zahr and T. Wen for providing their FOM and ROM code from [45] for the problems in Sections 4.2 and 4.3.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This research was supported in part by AFOSR Grant FA9550-22-1-0004 and a 2022 National Defense Science and Engineering Graduate (NDSEG) Fellowship.

References

[1] N. Alexandrov, J.E. Dennis Jr., R.M. Lewis, and V. Torczon, *A trust region framework for managing the use of approximation models in optimization*, Struct. Optim. 15 (1998), pp. 16–23. https://dx.doi.org/10.1007/BF01197433

[2] D. Amsallem, M.J. Zahr, Y. Choi, and C. Farhat, *Design optimization using hyper-reduced-order models*, Struct. Multidiscip. Optim. 51 (2015), pp. 919–940. https://dx.doi.org/10.1007/s00158-014-1183-y

[3] P. Benner, A. Cohen, M. Ohlberger, and K. Willcox, eds., *Model Reduction and Approximation: Theory and Algorithms*, Computational Science and EngineeringSIAM, Philadelphia, 2017. https://dx.doi.org/10.1137/1.9781611974829

[4] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, and L.M. Silveira, eds., *Model Order Reduction. Volume 2: Snapshot-Based Methods and Algorithms*, Walter de Gruyter & Co., Berlin, 2021.https://dx.doi.org/10.1515/9783110671490

[5] A.S. Berahas, L. Cao, and K. Scheinberg, *Global convergence rate analysis of a generic line search algorithm with noise*, SIAM J. Optim. 31 (2021), pp. 1489–1518. https://dx.doi.org/10.1137/19M1291832

[6] D.P. Bertsekas, *Projected Newton methods for optimization problems with simple constraints*, SIAM J. Control Optim. 20 (1982), pp. 221–246. https://dx.doi.org/10.1137/0320018

[7] D.P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, Massachusetts, 1995.

[8] R. Bollapragada, R.H. Byrd, and J. Nocedal, *Adaptive sampling strategies for stochastic optimization*, SIAM J. Optim. 28 (2018), pp. 3312–3343. https://dx.doi.org/10.1137/17M1154679

[9] R.H. Byrd, G.M. Chin, W. Neveitt, and J. Nocedal, *On the use of stochastic Hessian information in optimization methods for machine learning*, SIAM J. Optim. 21 (2011), pp. 977–995. https://dx.doi.org/10.1137/10079923X

[10] K.T. Carlberg, M. Barone, and H. Antil, *Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction*, J. Comput. Phys. 330 (2017), pp. 693–734. https://dx.doi.org/10.1016/j.jcp.2016.10.033

[11] R.G. Carter, *On the global convergence of trust region algorithms using inexact gradient information*, SIAM J. Numer. Anal. 28 (1991), pp. 251–265. https://dx.doi.org/10.1137/0728014

[12] R.G. Carter, *Numerical experience with a class of algorithms for nonlinear optimization using inexact function and gradient information*, SIAM J. Sci. Comput. 14 (1993), pp. 368–388. https://dx.doi.org/10.1137/0914023

[13] A.R. Conn, N.I.M. Gould, and P.L. Toint, *Trust-Region Methods*, MPS/SIAM Series on Optimization, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000. https://dx.doi.org/10.1137/1.9780898719857Mathematical Programming Society (MPS), Philadelphia, PA

[14] J.E. Dennis Jr. and R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Classics in Applied Mathematics Vol. 16, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996. https://dx.doi.org/10.1137/1.9781611971200 Corrected reprint of the 1983 original.

[15] M. Fahl and E. Sachs, *Reduced order modelling approaches to PDE–constrained optimization based on proper orthogonal decomposition*, in *Large-Scale PDE-Constrained Optimization*, L.T. Biegler, O. Ghattas, M. Heinkenschloss and B. van Bloemen Waanders, eds., Lecture Notes in Computational Science and Engineering, Vol. 30, Springer-Verlag, Heidelberg, 2003, pp. 268–280. https://dx.doi.org/10.1007/978-3-642-55508-4_16.

[16] E.M. Gafni and D.P. Bertsekas, *Two-metric projection methods for constrained optimization*, SIAM J. Control Optim. 22 (1984), pp. 936–964. https://dx.doi.org/10.1137/0322061

[17] C. Gogu, *Improving the efficiency of large scale topology optimization through on-the-fly reduced order model construction*, Internat. J. Numer. Methods Engrg. 101 (2015), pp. 281–304. https://dx.doi.org/10.1002/nme.4797

[18] S. Gratton, P.L. Toint, and A. Tröltzsch, *How much gradient noise does a gradient-based linesearch method tolerate?*, Tech. Rep. NAXYS-04-2012, Namur Center for Complex Systems, University of Namur, 61, rue de Bruxelles, B5000 Namur (Belgium), 2012.

[19] M.A. Grepl and A.T. Patera, *A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations*, M2AN Math. Model. Numer. Anal. 39 (2005), pp. 157–181.

[20] D.S. Grundvig, *Line-search based optimization using function approximations with tunable accuracy*, Master's thesis, Department of Computational Applied Mathematics and Operations Research, Rice University, Houston, TX, 2023. available at https://hdl.handle.net/1911/115072.

[21] B. Haasdonk and M. Ohlberger, *Efficient reduced models and a posteriori error estimation for parametrized dynamical systems by offline/online decomposition*, Math. Comput. Model. Dyn. Syst. 17 (2011), pp. 145–161. http://www.informaworld.com/10.1080/13873954.2010.514703.

[22] M. Heinkenschloss and L.N. Vicente, *An interface between optimization and application for the numerical solution of optimal control problems*, ACM Trans. Math. Softw. 25 (1999), pp. 157–190. https://dx.doi.org/10.1145/317275.317278

[23] M. Heinkenschloss and L.N. Vicente, *Analysis of inexact trust-region SQP algorithms*, SIAM J. Optim.12 (2002), pp. 283–302. https://dx.doi.org/10.1137/S1052623499361543

[24] W. Hess and S. Ulbrich, *An inexact ℓ_1 penalty SQP algorithm for PDE-constrained optimization with an application to shape optimization in linear elasticity*, Optim. Methods Softw. 28 (2013), pp. 943–968. https://dx.doi.org/10.1080/10556788.2011.651082

[25] J.S. Hesthaven, G. Rozza, and B. Stamm, *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*, Springer Briefs in Mathematics, Springer, New York, 2015https://dx.doi.org/10.1007/978-3-319-22470-1

[26] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, *Optimization with PDE Constraints*, Mathematical Modelling, Theory and Applications Vol. 23, Springer Verlag, Heidelberg, New York, Berlin, 2009. https://dx.doi.org/10.1007/978-1-4020-8839-1

[27] T. Keil, L. Mechelli, M. Ohlberger, F. Schindler, and S. Volkwein, *A non-conforming dual approach for adaptive trust-region reduced basis approximation of PDE-constrained parameter optimization*, ESAIM Math. Model. Numer. Anal. 55 (2021), pp. 1239–1269. https://dx.doi.org/10.1051/m2an/2021019

[28] C.T. Kelley, *Iterative Methods for Optimization*, Frontiers in Applied Mathematics Vol. 18, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999. https://dx.doi.org/10.1137/1.9781611970920

[29] D.P. Kouri and D. Ridzal, *Inexact trust-region methods for PDE-constrained optimization*, in *Frontiers in PDE-Constrained Optimization*, H. Antil, D.P. Kouri, M.D. Lacasse, and D. Ridzal, eds., IMA Vol. Math. Appl. Vol. 163, Springer, New York, 2018, pp. 83–121. https://dx.doi.org/10.1007/978-1-4939-8636-1_3

[30] D.P. Kouri, M. Heinkenschloss, D. Ridzal, and B.G. van Bloemen Waanders, *A trust-region algorithm with adaptive stochastic collocation for PDE optimization under uncertainty*, SIAM J. Sci. Comput.35 (2013), pp. A1847–A1879. https://dx.doi.org/10.1137/120892362

[31] D.P. Kouri, M. Heinkenschloss, D. Ridzal, and B.G. van Bloemen Waanders, *Inexact objective function evaluations in a trust-region algorithm for PDE-constrained optimization under uncertainty*, SIAM J. Sci. Comput. 36 (2014), pp. A3011–A3029. https://dx.doi.org/10.1137/140955665

[32] Q. Li, O. Sigmund, J.S. Jensen, and N. Aage, *Reduced-order methods for dynamic problems in topology optimization: a comparative study*, Comput. Methods Appl. Mech. Engrg. 387 (2021), pp. 114149, 34. https://dx.doi.org/10.1016/j.cma.2021.114149

[33] L. Machiels, Y. Maday, A.T. Patera, and D.V. Rovas, *A blackbox reduced-basis output bound method for shape optimization*, in *Proceedings of the 12th International Conference on Domain Decomposition Methods in Chiba, Japan*, T. Chan, T. Kako and H.K.O. Pironneau, eds., DDM.org, 2001, pp. 429–436. https://www.ddm.org.

[34] A. Manzoni, A. Quarteroni, and G. Rozza, *Shape optimization for viscous flows by reduced basis methods and free- form deformation*, Int. J. Numer. Methods. Fluids. 70 (2012), pp. 646–670. Available at https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.2712

[35] C. Nguyen, X. Zhuang, L. Chamoin, X. Zhao, H. Nguyen-Xuan, and T. Rabczuk, *Three-dimensional topology optimization of auxetic metamaterial using isogeometric analysis and model order reduction*, Comput. Methods Appl. Mech. Engrg. 371 (2020), pp. 113306, 17. https://dx.doi.org/10.1016/j.cma.2020.113306

[36] J. Nocedal and S.J. Wright, *Numerical Optimization*, 2nd ed., Springer Verlag, Berlin, Heidelberg, New York, 2006. https://dx.doi.org/10.1007/978-0-387-40065-5

[37] C. Paquette and K. Scheinberg, *A stochastic line search method with expected complexity analysis*, SIAM J. Optim. 30 (2020), pp. 349–376. https://dx.doi.org/10.1137/18M1216250

[38] C. Prud'homme, D.V. Rovas, K. Veroy, L. Machiels, Y. Maday, A.T. Patera, and G. Turinici, *Reliable real-time solution of parametrized partial differential equations: reduced-basis output bound methods*, J. Fluids. Eng. 124 (2002), pp. 70–80. Available at http://link.aip.org/link/?JFG/124/70/1

[39] E. Qian, M.A. Grepl, K. Veroy, and K. Wilcox, *A certified trust region reduced basis approach to PDE-constrained optimization*, SIAM J. Sci. Comput. 39 (2017), pp. S434–S460. https://dx.doi.org/10.1137/16M1081981

[40] A. Quarteroni, A. Manzoni, and F. Negri, *Reduced Basis Methods for Partial Differential Equations. An Introduction*, Unitext Vol. 92, Springer, Cham, 2016. https://dx.doi.org/10.1007/978-3-319-15431-2

[41] G. Rozza, T. Lassila, and A. Manzoni, *Reduced basis approximation for shape optimization in thermal flows with a parametrized polynomial geometric map*, in *Spectral and High Order Methods for Partial Differential Equations*, J.S. Hesthaven and E.M. Rønquist, eds., Lect. Notes Comput. Sci. Eng. Vol. 76, Springer, Heidelberg, 2011, pp. 307–315. https://dx.doi.org/10.1007/978-3-642-15337-2_28

[42] A. Schmidt, D. Wittwar, and B. Haasdonk, *Rigorous and effective a-posteriori error bounds for nonlinear problems–application to RB methods*, Adv. Comput. Math. 46 (2020), pp. 32, 30. https://dx.doi.org/10.1007/s10444-020-09741-x

[43] M. Schmidt, N. Le Roux, and F. Bach, *Convergence rates of inexact proximal-gradient methods for convex optimization*, in *NIPS'11: Proceedings of the 24th International Conference on Neural Information Processing Systems, December 2011*, 2011, pp. 1458–1466. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2011/file/8f7d807e1f53eff5f9efbe5cb81090fb-Paper.pdf.

[44] S. Ulbrich and J.C. Ziems, *Adaptive multilevel trust-region methods for time-dependent PDE-constrained optimization*, Port. Math. 74 (2017), pp. 37–67. https://dx.doi.org/10.4171/PM/1992

[45] T. Wen and M.J. Zahr, *A globally convergent method to accelerate large-scale optimization using on-the-fly model hyperreduction: application to shape optimization*, J. Comput. Phys. 484 (2023), pp. 112082, 33. https://dx.doi.org/10.1016/j.jcp.2023.112082

[46] D. Wirtz, D.C. Sorensen, and B. Haasdonk, *A posteriori error estimation for DEIM reduced nonlinear dynamical systems*, SIAM J. Sci. Comput. 36 (2014), pp. A311–A338. https://dx.doi.org/10.1137/120899042

[47] M. Xiao, D. Lu, P. Breitkopf, B. Raghavan, S. Dutta, and W. Zhang, *On-the-fly model reduction for large-scale structural topology optimization using principal components analysis*, Struct. Multidiscip. Optim. 62 (2020), pp. 209–230. https://dx.doi.org/10.1007/s00158-019-02485-3

[48] M. Yano, T. Huang, and M.J. Zahr, *A globally convergent method to accelerate topology optimization using on-the-fly model reduction*, Comput. Methods Appl. Mech. Engrg. 375 (2021), pp. 113635, 38. https://dx.doi.org/10.1016/j.cma.2020.113635

[49] G.H. Yoon, *Structural topology optimization for frequency response problem using model reduction schemes*, Comput. Methods Appl. Mech. Engrg. 199 (2010), pp. 1744–1763. https://dx.doi.org/10.1016/j.cma.2010.02.002

[50] Y.X. Yuchen, R. Bollapragada, R.H. Byrd, and J. Nocedal, *Constrained and composite optimization via adaptive sampling methods*, IMA J Numer Anal. 44 (2024), pp. 680–709. https://doi.org/10.1093/imanum/drad020

[51] Y. Yue and K. Meerbergen, *Accelerating optimization of parametric linear systems by model order reduction*, SIAM J. Optim. 23 (2013), pp. 1344–1370. https://dx.doi.org/10.1137/120869171

[52] M.J. Zahr and C. Farhat, *Progressive construction of a parametric reduced-order model for PDE-constrained optimization*, Int. J. Numer. Methods. Eng. 102 (2015), pp. 1111–1135. https://dx.doi.org/10.1002/nme.4770

[53] M.J. Zahr, K.T. Carlberg, and D.P. Kouri, *An efficient, globally convergent method for optimization under uncertainty using adaptive model reduction and sparse grids*, SIAM/ASA J. Uncertain. Quantif.7 (2019), pp. 877–912. https://dx.doi.org/10.1137/18M1220996

[54] J.C. Ziems and S. Ulbrich, *Adaptive multilevel inexact SQP methods for PDE-constrained optimization*, SIAM J. Optim. 21 (2011), pp. 1–40. https://dx.doi.org/10.1137/080743160