Optimization by Space Transformation and Decomposition

S. Gratton * L. N. Vicente† Z. Zhang ‡

December 30, 2019

Abstract

We introduce a *space transformation framework* for unconstrained optimization of a function f that is smooth but not necessarily convex. At each iteration of the framework, the gradient of f is mapped by a forward transformation to an auxiliary (possibly lifted) space, in which a model is established based on the transformed gradient and a step is obtained by minimizing this model. A backward transformation then maps the step back to the original space, where the iterate is updated accordingly. Using a trust-region globalization scheme, and by inspection of the consistency between the forward and backward transformations, the framework is guaranteed to converge globally to first-order criticality with an $\mathcal{O}(\epsilon^{-2})$ worst-case iteration complexity to reach a gradient with norm smaller than $\epsilon > 0$. The complexity is improved to $\mathcal{O}(\epsilon^{-1})$ and $\mathcal{O}(\log(\epsilon^{-1}))$ when f is convex and strongly convex respectively.

The space transformation framework can be directly specialized to a parallel space decomposition framework for nonlinear optimization, which can be regarded as an extension of the parallel Schwarz domain decomposition method for PDEs and linear systems. Such a decomposition framework is motivated by problems that cannot be directly solved (or even saved) in the full space and hence divide-and-conquer is needed. As in domain decomposition, introducing *overlaps* between subspaces can improve the performance of space decomposition methods provided that overlaps are handled properly. Our space decomposition framework covers a wide range of overlap-handling strategies, including Restricted Additive Schwarz, Weighted Restricted Additive Schwarz, and Additive Schwarz with Harmonic Extension, all of which have achieved remarkable success in domain decomposition. Similar to the coarse grid techniques in domain decomposition, we incorporate a *coarse space correction* into our framework. An unsophisticated implementation of the framework works quite well in our numerical experiments. With Restricted Additive Schwarz and coarse space correction, the algorithm scales nicely when the number of subspaces increases.

At the same time, the space transformation framework can be specialized to analyze trust-region methods when the gradient of f is evaluated *inaccurately*. Applying the theory of the space transformation framework, we provide *sharp bounds* on the gradient inaccuracy that

*ENSEEIHT, INPT, 2 rue Charles Camichel, 31071, Toulouse Cedex 7, France (serge.gratton@enseeiht.fr).
Support from the ANR-3IA Artificial and Natural Intelligence Toulouse Institute is gratefully acknowledged.

†Department of Industrial and Systems Engineering, Lehigh University, 200 West Packer Avenue, Bethlehem, PA 18015-1582, USA and Centre for Mathematics of the University of Coimbra (CMUC) (lnv@lehigh.edu).
Support for this author was partially provided by FCT/Portugal under grants UID/MAT/00324/2019 and P2020 SAICTPAC/0011/2015.

‡Department of Applied Mathematics, The Hong Kong Polytechnic University, Hong Kong, China (zaikun.zhang@polyu.edu.hk). Research of this author was supported by FCT under grant PTDC/MAT/116736/2010 (Portugal), by Fondation RTRA STAE under the project FILAOS and IRT Saint Exupéry (France), and currently by RGC Hong Kong under grants PolyU 253012/17P (ECS) and F-PolyU502/16 (PROCORE).

guarantee global convergence of trust-region methods, and prove new global convergence results while recovering some existing ones. More importantly, we establish a complete theory of worst-case complexities for trust-region methods based on inaccurate gradients. If the gradient inaccuracy respects the aforementioned bounds, these complexities are of the same order as when the gradients are accurate, and the gradient inaccuracy only enlarges the multiplicative constants in them by a mild magnitude that is entirely decided by trust-region algorithmic parameters and *independent* of the optimization problem. On the other hand, once the gradient inaccuracy exceeds such bounds, the behavior of trust-region methods *deteriorates dramatically*, which is demonstrated by our numerical experiment. Our theory provides theoretical insights into the tight connection between algorithmic parameters and the amount of relative gradient error that trust-region methods can tolerate. We show that such a connection is clearly observable in computation. This enables us to propose parameters that are well adapted to a given accuracy level of the gradient.

Keywords: Space transformation, space decomposition, coarse space, parallel, inaccurate gradient, trust-region methods, worst-case complexity

Contents

1 Introduction

Consider minimizing a smooth but possibly nonconvex function over a finite-dimensional space

$$\min_{x\in\mathbb{R}^n} f(x). \tag{1.1}$$

We propose a space transformation framework (Algorithm 2.1) for solving this problem. The framework is iterative. At iteration k, instead of seeking an update to the iterate in $\mathbb{R}^n$, the framework transfers the information about f to an *auxiliary space* $\mathbb{R}^{N_k}$, pursues a step in it, and then maps the step back to $\mathbb{R}^n$, where the iterate is updated accordingly. In this process, information is communicated by transformations between $\mathbb{R}^n$ and the auxiliary spaces as illustrated in Figure 1, where a forward transformation (abbreviated as for. trans.) refers to a transformation from $\mathbb{R}^n$ to an auxiliary space while a backward transformation (abbreviated as back. trans.) is a transformation in the other direction.

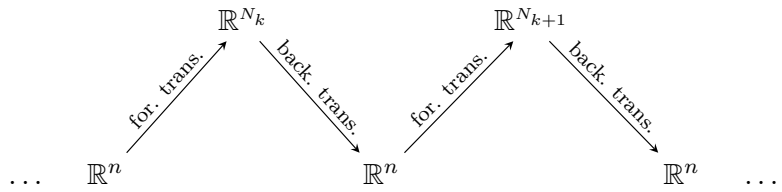


Figure 1: Information flow of the space transformation framework.

This framework is motivated by the optimization problems with increasingly high dimensionality and complexity that are emerging progressively from various areas including statistics, data science, and artificial intelligence. As a reaction to such problems, three trends, besides many others, are observable in the development of optimization algorithms and their theory: first, divide-and-conquer strategy for problems that are impossible to be solved — or even saved — on a single computer; second, exploration of inaccurate information when accurate information is unavailable or prohibitively expensive; third, investigation of worst-case complexity for understanding the global performance of algorithms and analyzing how the performance is affected by problem properties and algorithmic parameters. Our framework provides a unified

perspective connecting the first two points, and its theory complies with the third one, as will be elaborated in the following subsections.

1.1 Optimization by space decomposition

The first motivation of the space transformation framework comes from optimization methods based on space decomposition. When the direct minimization of f is out of reach, a natural idea is to split it into subproblems involving subsets of variables, to solve them separately in parallel,[1] and to combine all the subproblem solutions into a solution for the original problem. Obviously, unless the problem enjoys a form of separability, such a process cannot solve (1.1) in one step and hence has to be iterated.

The idea of decomposing the variable space is not new and can be traced back to Jacobi's method for linear systems (1845) [83]. Even though this paper focuses on deterministic methods, we shall first mention that the highly successful randomized Block Coordinate Descent (BCD) type optimization methods (see, for example, [104, 149, 122, 112]) can be regarded as space decomposition methods, although, thanks to the randomization, the subspaces in each iteration do not need to constitute a decomposition of the full $\mathbb{R}^n$ space. More related to this paper, many deterministic methods for solving (1.1) by decomposing $\mathbb{R}^n$ have been proposed in the optimization community, examples including Parallel Variable Distribution [49, 131], Parallel Gradient Distribution [93], Parallel Variable Transformation [52], and Parallel Line Search Subspace Correction [42]. All these (deterministic) methods integrate two stages at each iteration: first, decompose $\mathbb{R}^n$ into possibly overlapping subspaces, in each of which a subspace step is obtained by minimizing the objective function or a model; second, map the subspace steps back to $\mathbb{R}^n$ and update the iterate by a full-space step that is computed based on the subspace steps.[2] In these methods, the full-space step generally lies in the linear space spanned by the subspace steps and possibly the current iterate (see [52, equation (3)]). For instance, one may seek such a step by minimizing the objective function over the affine hull of the subspace steps [49], or, in convex cases, taking a convex combination of the subspace steps [93, 134, 135], or simply taking the subspace step that renders the smallest objective function value. After the full-space step is obtained, one may further improve it by a line search along its direction [42].

Meanwhile, as a major inspiration for our work, the methodology of space decomposition has been long explored in numerical methods for partial differential equations (PDEs) and linear systems, where it is more often known as *domain decomposition*.[3] Located in the center of an active research area of scientific computing, the paradigm of domain decomposition is vast and contains, for instance, the Schwarz, Neuman-Neuman, FETI, and optimized Schwarz methods. The literature is enormous, classical references including [151, 24, 130, 121, 141], and a more recent one being [41]. Our framework is most related to the parallel Schwarz domain decomposition method, which was devised by Lions [91] as a parallel variant of the alternating method by Schwarz [128] (see Gander [53] for historical remarks). The basic idea of parallel Schwarz method is to divide the global domain of the PDE into subdomains, solve the PDE in the subdomains

[1] There do exist sequential approaches but this paper will focus on parallel ones.

[2] We will focus on synchronous approaches. Asynchronous ones (*e.g.*, [1]) are discussed in Section 7.

[3] In numerical methods for PDEs, decision variables correspond to nodes in the domain where the PDE is defined and discretized. Decomposing the domain leads to grouping the variables, and hence decomposing the variable space. We will stick to the term *space decomposition* in an optimization context, as we do *not* assume that the optimization problem discretizes any infinite-dimensional problem over a domain. The word "space decomposition" is also common in the literature of numerical PDEs, [151] being an example.

simultaneously, and construct an approximate global solution by the local ones. The procedure is iterated after exchanging information between subdomains. When the subdomains exhibit *overlaps*, one has to adopt a strategy to assign values to the variables repeated on several subdomains. A naive approach known as the *Additive Schwarz* is to update the overlapping variables by summing up the updates suggested by the subdomains that contain these variables (see [53, equation (3.20)]). Additive Schwarz is mostly used in the design of preconditioners and does not converge in general as an iterative solver. One can easily observe a serious defect of Additive Schwarz by considering a decomposition consisting of two overlapping subdomains and a PDE that is perfectly decoupled with respect to such a decomposition. In this scenario, the update on the overlap is twice what it should be. A better way is to update the overlapping variables by a convex combination of the updates suggested by related subdomains, leading to the well-known (*Weighted*) *Restricted Additive Schwarz* approach by Cai and Sarkis [14], which performs much better than Additive Schwarz. There exist other approaches like (*Weighted*) *Additive Schwarz with Harmonic Extension* [14], which also outperforms Additive Schwarz in practice.

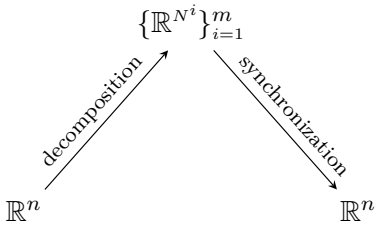
Despite the remarkable success of domain decomposition, its essential techniques are little explored in optimization algorithms based on space decomposition, unless the optimization problem is the discretization of a certain infinite-dimensional problem. It is for instance well known that the good performance of domain decomposition methods crucially relies on overlapping strategies such as Restricted Additive Schwarz and Additive Schwarz with Harmonic Extension. It is also commonly noted that *coarse grid* techniques are indispensable for the optimal convergence of domain decomposition in very large problems, without which the performance of domain decomposition methods will deteriorate when the number of subdomains increases and hence loose *scalability*. The question we raise in this paper is how to incorporate these techniques into a framework for nonlinear optimization, in such a way that *global convergence*[4] is guaranteed and standard *worst-case complexities* of nonlinear optimization methods are preserved.

To answer this question, we propose in Algorithm 5.1 a space decomposition framework for problem (1.1). Each iteration of this framework decomposes $\mathbb{R}^n$ into several subspaces $\{\mathcal{X}^i\}_{i=1}^m$, calculates subspace steps by minimizing subspace models of f, and then updates the iterate in $\mathbb{R}^n$ according to a full-space step formed by linear transformations of the subspace ones. In this framework, the subspaces $\{\mathcal{X}^i\}_{i=1}^m$ are not shown explicitly but represented by spaces $\{\mathbb{R}^{N^i}\}_{i=1}^m$. Such a framework, as we will elaborate in Section 5, extends the parallel Schwarz domain decomposition method to optimization, accommodating naturally (Weighted) Restricted Additive Schwarz, (Weighted) Additive Schwarz with Harmonic Extension, and coarse space correction.

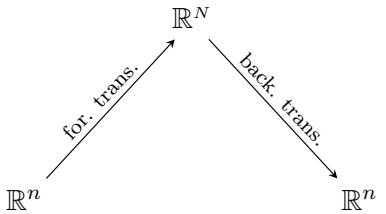
It turns out that our space decomposition framework can be interpreted as a special instance of the space transformation framework. The key idea is to introduce a *lifted space* $\mathbb{R}^N \simeq \bigoplus_{i=1}^m \mathbb{R}^{N^i}$ with $N = \sum_{i=1}^m N^i$, congregate the subspace models into a model of f in $\mathbb{R}^N$, and regard the subspace steps as the components of a step in $\mathbb{R}^N$, as rigorously formulated in Subsection 5.4. From this perspective, the decomposition framework indeed models the objective function in the (lifted) auxiliary space $\mathbb{R}^N$, seeks a step in this space, and then updates the iterate in $\mathbb{R}^n$ by a linear transformation of this step, which fits into the space transformation framework. Figure 2 illustrates a single iteration of the space decomposition framework from both the decomposition view and the transformation one. Note that the dimension N of the auxiliary space $\mathbb{R}^N$ can exceed that of $\mathbb{R}^n$ due to overlapping variables, and N can change along

[4] The word *global* has two meanings in this paper. Global convergence for an optimization algorithm means convergence regardless of the starting point. Global in the context of a PDE refers to the whole of its domain.

with iterations if it is desirable to vary the decomposition from one iteration to another.



(a) Decomposition view.



(b) Transformation view. ($N = \sum_{i=1}^m N^i$)

Figure 2: Information flow of the space decomposition framework (*a single iteration*).

1.2 Optimization based on inaccurate gradients

The second motivation for our space transformation framework comes from the situations where (1.1) is solved based on inaccurate gradients. A trivial source of gradient inaccuracy is finite precision arithmetic [59, 78], which happens to virtually *all* optimization problems, yet it is mostly neglected with or without a theoretical basis. In the traditional realm dominated by double-precision or at least single-precision computation, it might have been safe to neglect such inaccuracy, but now we are forced to include it into the analysis of optimization algorithms due to the increasing importance of low-precision computation in applications, particularly in machine learning [73, 85]. Gradient inaccuracy also occurs naturally in methods that approximate gradients by finite differences [20, 6]. In addition to these trivial cases, there are three more scenarios that motivate us to investigate optimization methods based on inaccurate gradients.

In the first scenario, gradient calculation involves solving a lower-level problem whose solution can only be approximated. This occurs, for instance, when gradients are obtained by solving certain *adjoint equations*. Such a gradient-evaluation approach is applied in PDE-constrained optimization [8, 79, 9, 38], multidisciplinary design optimization (MDO) [95], and more recently in deep learning [25], the adjoint equations being PDEs, linear systems, and ODEs, respectively. Despite well-developed solvers for these equations, solving them accurately is impossible in practice except for rather special cases, and hence gradient inaccuracy is inevitable.

It may be intuitive that gradient inaccuracy is undesirable and should be avoided if possible. To the contrast of such an intuition, in the second scenario, gradients are evaluated inaccurately in a deliberate way to *improve* the overall performance of optimization algorithms, reducing their expenses in terms of time or energy without affecting the accuracy of the final results. For example, in the adjoint approach mentioned above, lower accuracy of gradient evaluation may imply a *much lower* computational cost per gradient, and an overall speedup is possible if the global convergence and convergence rate of the optimization algorithm is not damaged by gradient inaccuracy. Indeed, such strategical exploration of inaccurate computation is becoming a prominent methodology of algorithm design due to the predictable saturation of Moore's law and the dramatically increasing dimensionality of modern application problems. We refer to [2, 110, 87, 89] for discussions of this methodology in scientific computing, to [68] for a recent investigation in optimization, and to [73] for an application in machine learning.

The third scenario, more counter-intuitive than the second one, is to intentionally introduce inaccuracy to gradients even if they are readily available with high accuracy. This happens

in the quantized gradient method [129, 60], where gradients are *quantized* to very few bits (hence inaccurate) in order to be transmitted with low communication latency in distributed computation. Another example is to impose perturbations to gradients in gradient descent methods, which can provably help to avoid convergence to saddle points [58, 84].

With these interesting scenarios where gradients are inaccurate, there exists plenty of research on the behavior of optimization algorithms under gradient inaccuracy, examples including [132, 37, 39], to name but a few. The intriguing analysis of [3, 67, 23, 145] on optimization methods based on probabilistic models can also be included in this category. In these works, gradient inaccuracy is regarded as noise, error, or perturbation. We, however, take an entirely different perspective: inaccurate gradient is a *transformation* of the accurate one.

For example, consider an iterative algorithm for (1.1). Suppose that, at iteration k, the algorithm evaluates an inaccurate gradient $\hat{g}_k$, then generates a step d_k based on the inaccurate information, and finally updates x_k according to d_k. Defining [5]

$$R_k \;=\; \frac{\hat{g}_k[\nabla f(x_k)]^{\mathsf{T}}}{\|\nabla f(x_k)\|^2} \;\in\; \mathbb{R}^{n\times n},$$

we can re-interpret the algorithm as follows: it first applies R_k to transform the gradient information from $\nabla f(x_k) \in \mathbb{R}^n$ to $\hat{g}_k \in \mathbb{R}^n$, then obtains a step d_k based on the *transformed information*, and finally updates x_k according to d_k. This fits into our space transformation framework, with the forward transformation being R_k, the auxiliary space being $\mathbb{R}^n$, and the backward transformation being the identity matrix.

Therefore, optimization based on inaccurate gradients can be regarded as a special instance of optimization based on space transformation, falling in the same category as space decomposition methods. Covering two apparently distinct subjects within one framework, the space transformation perspective economizes research effort. Moreover, for two seemingly unrelated topics, our framework enables us to see through superficial distinctions and reveal hidden similarities.

1.3 Organization of the paper and main results

The remaining part of this paper consists of three blocks. The first block (Sections 2–4) presents our space transformation framework, establishes its global convergence and worst-case complexities based on a trust-region globalization strategy, and discusses how to incorporate coarse spaces into the framework. The second block (Section 5) introduces the space decomposition framework and establishes the theory of its trust-region version by casting the results of the space transformation framework to this case. The third block (Section 6) specializes the theory of the space transformation framework to optimization based on inaccurate gradients, obtaining global convergence and worst-case complexity theory of a trust-region framework that uses inaccurate gradients. The second and third blocks are independent of each other.

Section 2 formulates the space transformation framework in its prototypical form as Algorithm 2.1. For such a framework to be practically useful when minimizing possibly nonconvex functions, Section 3 adopts a *trust-region* globalization technique, resulting in a trust-region version of the space transformation framework presented in Algorithm 3.1 of Subsection 3.1. The key step to analyze this framework is to prove that the reduction radio surpasses a certain threshold when the trust-region radius becomes small. To this end, Subsection 3.2 establishes a

[5] Our analysis focuses on the iterations before x_k becomes stationary (if this ever happens), and hence there is no concern about zero denominator in the definition of R_k.

lower bound for the reduction ratio based on the geometrical interplay between the trust-region strategy and the transformations (Lemma 3.1), which motivates the assumptions on trust-region steps (Assumption 3.4) and on the transformations (Assumptions 3.5–3.7), in addition to those inherited from classical trust-region analysis (Assumptions 3.1–3.3). With these assumptions, Subsection 3.3 establishes two crucial facts (Lemmas 3.2–3.3) that lead to the convergence properties of the framework. Using such facts, Subsection 3.4 proves the global convergence of the framework in terms of first-order stationarity (Theorem 3.1), and Subsection 3.5 establishes worst-case iterations complexities of orders $\mathcal{O}(\epsilon^{-2})$, $\mathcal{O}(\epsilon^{-1})$, and $\mathcal{O}(\log(\epsilon^{-1}))$ respectively in the nonconvex, convex, and strongly convex cases (Theorems 3.2–3.4). Subsection 3.6 is oriented to connoisseurs and practitioners of trust-region methods, and can be safely skipped in a first reading. It discusses practical updating schemes for trust-region radius, shows that the theory in Subsections 3.4–3.5 remains true under very broad updating rules (Theorems 3.5–3.6), and provides general strategies that guarantee $\|\nabla f(x_k)\| \to 0$ without assuming convexity or enforcing sufficient decease (Theorems 3.7–3.8), while traditional results ensure only $\liminf \|g_k\| = 0$.

Motivated by coarse grid techniques in domain decomposition methods, Section 4 discusses how to incorporate coarse space correction into the space transformation framework. The idea is to complement the auxiliary space and its model with information that reflects the overall behavior of the problem, which is particularly important in the case of space decomposition methods. Subsection 4.1 presents a space transformation framework with coarse space in Algorithm 4.1 along with its trust-region version in Algorithm 4.2. Subsection 4.2 establishes the global convergence and worst-case complexities of the trust-region framework (Theorem 4.1), obtaining similar results to what is established in Section 3. The key is again to establish a lower bound for the reduction ratio, which is done in Lemma 4.1.

Section 5 investigates space decomposition methods as a special instance of the space transformation framework. After an illustrative introduction to the parallel Schwarz domain decomposition method for linear systems in Subsection 5.2, Subsection 5.3 presents in Algorithm 5.1 our space decomposition framework for solving (1.1), extending the parallel Schwarz method to nonlinear optimization in a natural way. Subsection 5.4 shows that such a framework can be regarded as a special instance of the space transformation framework introduced in Section 2. Subsection 5.5 elaborates how to implement the Additive Schwarz type decomposition/synchronization strategies within our framework, covering the (Weighted) Restricted Additive Schwarz and Additive Schwarz with Harmonic Extension techniques, which have been proved essential for the success of the parallel Schwarz method. Subsection 5.6 presents a trust-region version of the space decomposition framework (Algorithm 5.2), and obtains its global convergence and worst-case complexities (Theorem 5.1) by directly applying the theory that has been obtained for the space transformation framework in Section 3. Capitalizing on the work of Section 4, Subsection 5.7 introduces a coarse space correction technique into our space decomposition framework (Algorithms 5.3 and 5.4) and establishes its convergence theory (Theorem 5.2). Subsection 5.8 demonstrates numerically the effectiveness and potential of our space decomposition framework on nonlinear and possibly nonconvex optimization problems, showing encouraging — sometimes surprisingly good — behavior of Restricted Additive Schwarz and coarse space correction on problems that do *not* necessarily have any PDE background.

Section 6 studies optimization based on inaccurate gradient information with a focus on trust-region methods. As already known, trust-region methods are extremely robust with respect to gradient inaccuracy [97, 138, 15, 17, 75, 81]. Section 6 further characterizes such robustness by providing provably tight bounds for the tolerable inaccuracy, establishing worst-case complex-

ities, and quantifying the impact of gradient inaccuracy upon these complexities. Within the bounds that we provide, gradient inaccuracy does not affect the order of the worst-case complexities of trust-region methods but only enlarges the multiplicative constants in them by a quite benign magnitude that is completely *independent* of the optimization problem. Subsection 6.2 presents in Algorithm 6.1 a trust-region framework for solving (1.1) based on inaccurate gradients and interprets it as a particular instance of the trust-region space transformation framework formulated in Algorithm 3.1. Based on such an interpretation, by a direct application of the theory developed in Section 3, Subsections 6.3–6.4 establish the global convergence and worst-case complexities for Algorithm 6.1 when the relative gradient error is below certain bounds (Theorems 6.1–6.2). Subsection 6.5 unifies the results of Subsections 6.3–6.4 into Theorem 6.3, which determines the largest admissible region for inaccurate gradients which can ensure the convergence and worst-case complexities of Algorithm 6.1. Subsection 6.6 contains a brief discussion on gradient descent with Armijo line search, showing that Theorems 6.1–6.3 are applicable to such an algorithm by recognizing it as a trust-region method in disguise. Subsection 6.7 observes the numerical behavior of Algorithm 6.1, demonstrating the sharpness of our theory.

Having finished all the three blocks, we conclude by Section 7 with some discussions and directions for future investigation. There are several appendices after Section 7. Most propositions in Sections 5–6 are proved there, as they can be justified by direct applications of the theory in Section 3. Appendix G discusses some alternatives to the assumptions imposed by Section 3 on the trust-region step and transformations, sketches how they can render almost the same theory as in Section 3 (Theorem G.1), and, as an application, establishes a theory for trust-region methods based on inaccurate gradients with dynamically adapted accuracy (Theorem G.2).

Partially due to the robustness and the ability to achieve second-order optimality (see, *e.g.*, [30, Sections 6.6, 8.1, and 9.3] and [32, 22, 64]), trust-region methods are enjoying an increasing attention in recent applications [127, 150, 74, 152, 153]. Being neither our main objective *nor our contribution*, one can specialize the analysis in Section 3 by setting $R_k = T_k = I$ to obtain the global convergence and worst-case complexities for the classical trust-region methods in the unconstrained and smooth case. Almost all such results are known [116, 118, 30, 156, 120, 139, 63, 159, 33]. We hope our work could provide a summary of these results as a supplement to the documentation in [30, 159]. The success of trust-region strategy on our space decomposition framework serves as another demonstration that, despite its long history, the trust-region methodology will continuously provide powerful algorithms so long as we continue to explore.

Notation

Due to the abundance of contents, this paper involves a large set of notation. Table 1 lists the major symbols that we use in this paper. It omits the self-explaining f and n, which are first used in (1.1), and excludes the notation locally used in a particular context.

Unless otherwise specified, subscripts will signify iteration counters, while indices for subspaces and associated objects will be denoted by superscripts. Power of numbers will be written as $[a]^p$ when the context does not suffice to avoid confusion (for example, Δ_k^i means a trust-region radius corresponding to iteration k and subspace i, while $[\Delta_k]^p$ is the pth power of a trust-region radius related to iteration k). We adopt the MATLAB-style notation $[a\,;b\,;\cdots;c\,]$ to denote a *vertical* array with a, b,..., c being the entries, while $[a\,,b\,,\cdots,c\,]$ means a horizontal one. They are particularly useful in Section 5 when we discuss the space decomposition framework.

Symbol	Explanation	First appearance
$\|\cdot\|$	2-norm of vectors or matrices	Algorithm 3.1
$\|\cdot\|$	absolute value of numbers *or* cardinality of sets	Lemma 3.3
k	iteration counter	Algorithm 2.1
ϵ	convergence threshold	Subsection 3.5
k_ϵ^g	$\min\{k \in \mathbb{N} : \|g_k\| \leq \epsilon\}$	Subsection 3.5.1
k_ϵ^f	$\min\{k \in \mathbb{N} : f(x_k) - \inf_{x\in\mathbb{R}^n} f(x) \leq \epsilon\}$ (convex case)	Subsection 3.5.2
k_ϵ^x	$\min\{k \in \mathbb{N} : \|x_k - \operatorname{argmin} f\| \leq \epsilon\}$ (strongly convex case)	Subsection 3.5.3
N_k	dimension of auxiliary space	Algorithm 2.1
R_k, T_k	forward transformation and backward transformation	Algorithm 2.1
h_k	model of f in auxiliary space $\mathbb{R}^{N_k}$	Algorithm 2.1
d_k	step in auxiliary space $\mathbb{R}^{N_k}$	Algorithm 2.1
s_k	step in original space $\mathbb{R}^n$	Algorithm 2.1
Δ_k	trust-region radius in trust-region algorithms	Algorithm 3.1
ρ_k	reduction ratio in trust-region algorithms	Algorithm 3.1
η_0, η_1, η_2	thresholds for ρ_k in trust-region algorithms	Algorithm 3.1
γ_0, γ_1, γ_2	expansion or contraction factors for trust-region radius	Algorithm 3.1
g_k	gradient of f at iterate x_k	Subsection 3.2
$\tilde{g}_k$	the vector with minimal norm among g_0, g_1, ..., g_k	Proposition 3.3
ϕ_k	angle between $-d_k$ and $R_k g_k$	Subsection 3.2.2
ψ_k	angle between $T_k^\mathsf{T} g_k$ and $R_k g_k$	Subsection 3.2.3
φ_k	angle between $-d_k$ and $T_k^\mathsf{T} g_k$	Subsection 3.2.3
L_f	Lipschitz constant for ∇f	Assumption 3.1
L_h	common Lipschitz constant for $\{\nabla h_k\}$	Assumption 3.2
α	constant in decrease condition for trust-region step	Assumption 3.3
β, p	constants in angle condition for trust-region step	Assumption 3.4
τ	upper bound for $\{\|T_k\|\}$	Assumption 3.5
κ	non-degeneracy constant for $\{R_k\}$	Assumption 3.6
λ	consistency constant for $\{R_k\}$ and $\{T_k\}$	Assumption 3.7
μ	lower bound for $\{\Delta_k/\|\tilde{g}_k\|\}$	Proposition 3.3
ς	strong convexity constant for f (if f is strongly convex)	Subsection 3.5.3
C_k	forward transformation for coarse space	Algorithm 4.1
H_k	model of f in coarse space	Algorithm 4.1
L_H	common Lipschitz constant for $\{\nabla H_k\}$	Assumption 4.1
δ_k	coarse space step	Algorithm 4.1
ω_k	convex combination coefficient for d_k and δ_k	Algorithm 4.1
ω	lower bound for $\{\omega_k\}$	Assumption 4.1
ν	upper bound for $\{\|C_k\|\}$	Assumption 4.1
m	number of subspaces in space decomposition	Algorithm 5.1
N^i	dimension of subspace i	Algorithm 5.1
R^i, T^i	decomposition/synchronization matrices for subspace i	Algorithm 5.1
h_k^i	model of f in subspace i	Algorithm 5.1
d_k^i	subspace step in subspace i	Algorithm 5.1
θ	multiplicity of space decomposition	Definition 5.3
U^i, $\tilde{U}^i$, W^i	Additive Schwarz type matrices	Subsection 5.5
$\hat{g}_k$	inaccurate gradient of f at x_k	Subsection 6.1
ζ	magnitude of relative error in inaccurate gradients	Theorem 6.1

Table 1: Symbols used in this paper.

2 A framework for optimization by space transformation

We propose Algorithm 2.1 as a framework for solving (1.1). Instead of tackling the problem in $\mathbb{R}^n$, at iteration k, we transfer the local information of f to an auxiliary space $\mathbb{R}^{N_k}$, where a step is obtained by minimizing a model h_k based on the transferred in information. Then the step is sent back to $\mathbb{R}^n$ in order to update the iterate. Inevitably, this framework relies on transformations between $\mathbb{R}^n$ and the auxiliary spaces, and therefore we call it "optimization by space transformation".

Algorithm 2.1 Optimization by Space Transformation (prototype)

Input: $x_0 \in \mathbb{R}^n$.

For $k = 0, 1, 2, \ldots$, iterate the following steps.

1. **Forward Transformation.** Define a function $h_k : \mathbb{R}^{N_k} \to \mathbb{R}$ satisfying
$$\nabla h_k(0) \;=\; R_k \nabla f(x_k)$$
with a certain positive integer N_k and a certain matrix $R_k \in \mathbb{R}^{N_k \times n}$.
2. **Minimization.** Calculate $d_k \approx \operatorname{argmin}\left\{h_k(d) : d \in \mathbb{R}^{N_k}\right\}$.
3. **Backward Transformation.** Define
$$s_k \;=\; T_k d_k$$
with a certain matrix $T_k \in \mathbb{R}^{n \times N_k}$.
4. **Update.** Set x_{k+1} to either $x_k + s_k$ or x_k *depending on the quality of* s_k.

Each iteration of the framework consists of four phases. The **Forward Transformation** phase defines a model $h_k : \mathbb{R}^{N_k} \to \mathbb{R}^n$ linked with f by $\nabla h_k(0) = R_k \nabla f(x_k)$, where $R_k \in \mathbb{R}^{N_k \times n}$ is a linear transformation mapping the first-order information of f from $\mathbb{R}^n$ to $\mathbb{R}^{N_k}$. The **Minimization** phase seeks a step d_k in the auxiliary space $\mathbb{R}^{N_k}$ by approximately minimizing h_k. The **Backward Transformation** phase transmits d_k back to $\mathbb{R}^n$ by a linear transformation $T_k \in \mathbb{R}^{n \times N_k}$, obtaining a step $s_k = T_k d_k \in \mathbb{R}^n$. Finally, the **Update** phase updates the iterate x_k in $\mathbb{R}^n$ according to the quality of s_k, the exact sense of which is left vague on purpose.

In the **Minimization** phase, we use $d_k \approx \operatorname{argmin}\{h_k(d) : d \in \mathbb{R}^{N_k}\}$ to signify that d_k minimizes h_k to some extent, as will be elaborated later. Since $\nabla h_k(0) = R_k \nabla f(x_k)$, we can interpret $h_k(d)$ as a first-order model for $f(x_k + R_k^{\mathsf{T}} d)$, and regard d_k as an approximate minimizer of $f(x_k + R_k^{\mathsf{T}} d)$. Therefore, when $x_{k+1} = x_k + s_k$, we have

$$x_{k+1} \;\approx\; x_k + T_k \left[\operatorname*{argmin}_{d \in \mathbb{R}^{N_k}} f(x_k + R_k^{\mathsf{T}} d)\right].$$

Hence it is tempting to set

$$T_k \;=\; R_k^{\mathsf{T}}. \tag{2.1}$$

However, it will turn out that equality (2.1) does not hold in the situations that interest us in this paper except for trivial cases. Indeed, in space decomposition methods, we violate equality (2.1) voluntarily for efficiency considerations (see RAS and ASH in Subsection 5.5), while

the equality fails intrinsically when we discuss optimization based on inaccurate gradient information (see (6.1) and (6.2)). In both cases, it is exactly the possibility of allowing discrepancy between T_k and R_k^T that makes the framework presented in Algorithm 2.1 nontrivial and useful.

One can extend Algorithm 2.1 by allowing R_k and T_k to be nonlinear transformations, as will be explained in Section 7. Concentrating on linear transformations between finite-dimensional spaces with fixed basis, we do not distinguish linear transformations and their matrix representations, and we will sometimes refer to them as "operators" as well. Since R_k plays the role of transferring gradients, it is indeed an operator between the *dual spaces* of $\mathbb{R}^n$ and $\mathbb{R}^{N_k}$, which will be important to note if the framework is to be generalized to optimization in Banach spaces. Since we will focus on convergence to first-order stationarity, Algorithm 2.1 only specifies the transmission of the first-order information explicitly. Higher-order versions of this framework will have to include transformations for higher-order information.

Remark 2.1. *For simplicity, the* ***Minimization*** *phase of Algorithm 2.1 denotes the variable of h_k^i by d* regardless of k. *Writing $h_k(d)$, we imply that $d \in \mathbb{R}^{N_k}$. Therefore, when we see the symbol d without any subscript or superscript, its dimension depends on whose variable it is standing for, which will always be clear in context. See also Remark 5.1 in Section 5.*

3 A trust-region version of the space transformation framework

3.1 Globalizing the space transformation framework by a trust region

Algorithm 2.1 obvious needs more specifications to be implementable and convergent. In particular, its **Minimization** and **Update** phases are intentionally vague, leaving many possibilities to develop practical versions of the framework. In this work, we will focus on an implementation of these two phases based on the trust-region methodology [30]. This leads us to Algorithm 3.1, a trust-region version of the space transformation framework.

In Algorithm 3.1, the **Minimization** phase sets d_k to be an inexact solution to the trust-region subproblem $\min_{\|d\| \le \Delta_k} h_k(d)$. No matter how inexact d_k is, we suppose that the constraint $\|d_k\| \le \Delta_k$ is respected. Following the terminology of trust-region methods, we will refer to h_k as the trust-region model, Δ_k as the trust-region radius, d_k as the trust-region step, s_k as the trial step,[6] $h_k(0) - h_k(d_k)$ as the predicted reduction, $f(x_k) - f(x_k + s_k)$ as the actual reduction, and ρ_k as the reduction radio. If the reduction ratio is big enough, the trial step is accepted (see (3.2)) and the trust-region radius is possibly increased (see (3.3)). Algorithm 3.1 uses two separate parameters η_0 and η_1 for the acceptance of trial steps and update of trust-region radius, a fashion similar to [118], [138, Algorithm 2], and [108, Algorithm 4.1]. When $\eta_0 = 0$, Algorithm 3.1 accepts trial steps with simple decreases (*i.e.*, $f(x_k + s_k) < f(x_k)$).

The essential distinction between Algorithm 3.1 and classical trust-region methods (see, for instance, [30, Algorithm 6.1.1]) lies in the transformations R_k and T_k. If both R_k and T_k are always the identity matrix, then Algorithm 3.1 reduces to a classical trust-region framework.

When $N_k \equiv n$, one may expect that Algorithm 3.1 converges provided that the transforma-

[6] In classical trust-region methods, the trust-region step and the trial step are identical. In Algorithm 3.1, however, the latter is a linear transformation of the former.

Algorithm 3.1 Optimization by Space Transformation (trust-region version)

Input: $x_0 \in \mathbb{R}^n$, $\Delta_0 > 0$; $\eta_0 \geq 0$, $\eta_1 > 0$ with $\eta_0 \leq \eta_1$; γ_0, γ_1, γ_2 with $0 < \gamma_0 \leq \gamma_1 < 1 \leq \gamma_2$.

For $k = 0, 1, 2, \ldots$, iterate the following steps.

1. **Forward Transformation.** Identical to Algorithm 2.1.
2. **Minimization.** Calculate
$$d_k \;\approx\; \operatorname{argmin}\big\{h_k(d) : d \in \mathbb{R}^{N_k},\, \|d\| \leq \Delta_k\big\}.$$
3. **Backward Transformation.** Identical to Algorithm 2.1.
4. **Update.** Calculate
$$\rho_k \;=\; \frac{f(x_k) - f(x_k + s_k)}{h_k(0) - h_k(d_k)}. \tag{3.1}$$
Then define
$$x_{k+1} \;=\; \begin{cases} x_k & \text{if } \rho_k \leq \eta_0, \\ x_k + s_k & \text{if } \rho_k > \eta_0, \end{cases} \tag{3.2}$$
and set Δ_{k+1} in such a way that
$$\Delta_{k+1} \in \begin{cases} [\gamma_0\Delta_k,\, \gamma_1\Delta_k] & \text{if } \rho_k \leq \eta_1, \\ [\Delta_k,\, \gamma_2\Delta_k] & \text{if } \rho_k > \eta_1. \end{cases} \tag{3.3}$$

tions R_k and T_k distort $\mathbb{R}^n$ to a controllable extent in the sense that, for all $v \in \mathbb{R}^n$,

$$\begin{cases} v^{\mathsf{T}} R_k v \;\geq\; a\|v\|\|R_k v\| \quad \text{and} \quad b\|v\| \;\leq\; \|R_k v\| \;\leq\; c\|v\|, & (3.4) \\ v^{\mathsf{T}} T_k v \;\geq\; a\|v\|\|T_k v\| \quad \text{and} \quad b\|v\| \;\leq\; \|T_k v\| \;\leq\; c\|v\|, & (3.5) \end{cases}$$

where a, b, and c are positive constants independent of k, meaning that R_k and T_k never change the direction or the length of any vector dramatically. However, such an expectation is incorrect. For the reason, see Example 6.1 and the comments afterwards.

Indeed, as we will see in our analysis, the transformations R_k and T_k have to be compatible with the value of η_1 in (3.3), the exact sense being elaborated in Assumption 3.7. If both R_k and T_k are always the identity matrix, the compatibility requirement will reduce to $\eta_1 < 1$, which is entirely conventional in trust-region methods. Among all the algorithmic parameters in Algorithm 3.1, it turns out the role played by η_1 is particularly important, which will become clear along our analysis. See Subsection 3.6.4, Theorem 5.1, and Subsection 6.5 for instances.

3.2 Assumptions and their consequences

To analyze Algorithm 3.1, we have to postulate a set of assumptions on the objective function f, the models $\{h_k\}$, the trust-region steps $\{d_k\}$, and the matrices $\{R_k\}$ and $\{T_k\}$. This subsection presents these assumptions and their consequences that will facilitate our analysis.

We denote henceforth
$$g_k \;=\; \nabla f(x_k).$$
Consequently, $\nabla h_k(0) = R_k g_k$. Nevertheless, we will write $\nabla f(x_k)$ and $\nabla h_k(0)$ instead of g_k and $R_k g_k$ when it is necessary to highlight that they are the gradients of f and h_k respectively.

3.2.1 Assumptions on the objective function f and the models $\{h_k\}$

The assumptions on the objective function and the models are conventional.

Assumption 3.1 (Lower boundedness and smoothness of objective function)**.** *f is bounded from below and differentiable in $\mathbb{R}^n$, and ∇f is L_f-Lipschitz continuous in $\mathbb{R}^n$ with a constant $L_f > 0$.*

According to this assumption, $\inf_{x\in\mathbb{R}^n} f(x)$ is a finite number and we will denote it by $f_{\inf}$. For simplicity, we assume the *global* Lipschitz continuity of ∇f. It is analogous to [30, Assumption AF.3], which requires the boundedness of $\nabla^2 f$ over $\mathbb{R}^n$. As in [30, Chapter 6], our analysis is still applicable after minor modifications if ∇f is only Lipschitz continuous in a neighborhood of the level set $\{x \in \mathbb{R}^n : f(x) \le f(x_0)\}$ (see [108, Theorem 4.5] for a similar case).

Assumption 3.2 (Smoothness of models)**.** *For each $k \ge 0$, h_k is continuously differentiable in $\mathbb{R}^{N_k}$, and ∇h_k is L_h-Lipschitz continuous in $\mathbb{R}^{N_k}$ with a constant $L_h > 0$ independent of k.*

Note that we assume that all the model gradients $\{\nabla h_k\}$ share a common Lipschitz constant. This assumption is not unusual in the analysis of trust-region methods (for example, see [108, Theorem 4.5]). Using a technique by Powell [118, 120], one can indeed establish the global convergence of Algorithm 3.1 even if the Lipschitz constants of $\{\nabla h_k\}$ are unbounded, as long as they do not grow too fast. It is also adequate to assume that $\nabla h_k(d)$ is Lipschitz continuous for $\|d\| \le \Delta_k$ (instead of $d \in \mathbb{R}^{N_k}$) with a common Lipschitz constant L_h for all $k \ge 0$.

3.2.2 Assumptions on the trust-region steps $\{d_k\}$ (I)

The following assumption is typical in the analysis of trust-region methods (*e.g.*, [116, 118, 30]).

Assumption 3.3 (Sufficient decrease of trust-region steps)**.** *There exists a constant $\alpha \in (0, 1]$ such that*

$$h_k(0) - h_k(d_k) \;\ge\; \frac{\alpha}{2}\|\nabla h_k(0)\| \min\left\{\Delta_k,\; \frac{\|\nabla h_k(0)\|}{L_h}\right\} \quad \textit{for each} \quad k \ge 0.$$

Later we will introduce an assumption that ensures $\nabla h_k(0) \ne 0$ as long as $g_k \ne 0$ (see Assumption 3.6 in Subsection 3.2.5). Such an assumption and Assumption 3.3 guarantee that

$$h_k(0) - h_k(d_k) \;>\; 0 \quad \text{and} \quad \|d_k\| > 0$$

unless $g_k = 0$. In our analysis, we will suppose by convention that $g_k = 0$ never happens within finite iterations so that Algorithm 3.1 iterates infinitely with ρ_k well defined for each $k \ge 0$. In computation, if $h_k(0) - h_k(d_k) = 0$ occurs then one can terminate the algorithm and conclude $g_k = 0$ according to Assumptions 3.3 and 3.6.

By Taylor expansion, we have

$$h_k(d) \;\le\; h_k(0) + d^{\mathsf{T}}\nabla h_k(0) + \frac{L_h}{2}\|d\|^2. \tag{3.6}$$

Consider the point that minimizes the right-hand side of (3.6) on the half line $\{-t\nabla h_k(0) : t \ge 0\}$ under the constraint $\|d\| \le \Delta_k$. Let us denote this point by $\bar{d}_k$. By straightforward calculations,

$$\bar{d}_k \;=\; -\min\left\{\frac{\Delta_k}{\|\nabla h_k(0)\|},\; \frac{1}{L_h}\right\}\nabla h_k(0). \tag{3.7}$$

According to Taylor expansion and the fact that $-\bar{d}_k^{\mathsf{T}}\nabla h_k(0) = \|\nabla h_k(0)\|\|\bar{d}_k\| \geq L_h\|\bar{d}_k\|^2$,

$$\begin{aligned} h_k(0) - h_k(\bar{d}_k) \;&\geq\; -\bar{d}_k^{\mathsf{T}}\nabla h_k(0) - \frac{L_h}{2}\|\bar{d}_k\|^2 \\ &\geq \frac{1}{2}\|\nabla h_k(0)\|\|\bar{d}_k\| \\ &= \frac{1}{2}\|\nabla h_k(0)\| \min\left\{\Delta_k,\;\; \frac{\|\nabla h_k(0)\|}{L_h}\right\}. \end{aligned} \tag{3.8}$$

Therefore, Assumption 3.3 is fulfilled as long as the reduction in h_k achieved by d_k is at least a constant fraction of the amount achieved by $\bar{d}_k$.

Recall that the classical Cauchy step d_k^{C} of the trust-region subproblem $\min_{\|d\|\leq\Delta_k} h_k(d)$ is defined as a minimizer of h_k on the half line $\{-t\nabla h_k(0) : t \geq 0\}$ subject to $\|d\| \leq \Delta_k$, which is easily computable when h_k is quadratic (see [30, Section 6.3] and [108, Section 4.1]). Obviously,

$$h_k(d_k^{\mathrm{C}}) \;\leq\; h_k(\bar{d}_k).$$

Hence $\bar{d}_k$ defined in (3.7) is slightly less demanding than the classical Cauchy step as it requires no more reduction in h_k. If we set d_k to d_k^{C} or any step that reduces h_k even more than d_k^{C}, then

$$h_k(d_k) \;\leq\; h_k(\bar{d}_k)$$

and consequently guarantee Assumption 3.3. When h_k is quadratic, such a step can be Powell's dog-leg step [115] if $\nabla^2 h_k$ is positive definite, Toint-Steihaug truncated conjugate gradient step [133, 137] (see also [155]), the two-dimensional subspace minimization step in [12], and obviously, the exact step [98]. We refer to [30, Section 6.3.3] for a backtracking technique that yields a step d_k satisfying Assumption 3.3 when h_k is a nonlinear model.

Let ϕ_k be the angle between d_k and $-\nabla h_k(0)$. Then Assumption 3.3 ensures

$$0 \;\leq\; \phi_k \;\leq\; \frac{\pi}{2} \quad \text{when} \quad \Delta_k \;\leq\; \frac{\alpha\|\nabla h_k(0)\|}{L_h}, \tag{3.9}$$

In fact, by Assumption 3.3 and Taylor expansion, when $\Delta_k \leq \alpha\|\nabla h_k(0)\|/L_h$,

$$\frac{\alpha}{2}\|\nabla h_k(0)\|\Delta_k \;\leq\; h_k(0) - h_k(d_k) \;\leq\; -d_k^{\mathsf{T}}\nabla h_k(0) + \frac{L_h}{2}[\Delta_k]^2,$$

which implies that

$$-d_k^{\mathsf{T}}\nabla h_k(0) \;\geq\; \frac{\alpha}{2}\|\nabla h_k(0)\|\Delta_k - \frac{L_h}{2}[\Delta_k]^2 \;=\; \frac{L_h}{2}\Delta_k\left(\frac{\alpha\|\nabla h_k(0)\|}{L_h} - \Delta_k\right) \;\geq\; 0. \tag{3.10}$$

As in classical trust-region analysis, the key to establishing the convergence of Algorithm 3.1 is to prove that ρ_k will become sufficiently large when Δ_k is sufficiently small, as will be detailed in Proposition 3.2. To this end, we need an instrumental estimation for ρ_k. Assumptions 3.1–3.3 can already render such an estimation. We present it in Lemma 3.1 before postulating more assumptions. This lemma will clarify how the quality of d_k and the geometrical properties of R_k and T_k influence the value of ρ_k, and it will play a central role in our analysis.

3.2.3 The critical lemma: A lower bound for the reduction ratio ρ_k

Our estimation for ρ_k is inspired by geometrical observations, the following angles being crucial:

$$\begin{cases} \phi_k &= \text{the angle between } -d_k \text{ and } R_k g_k, & (3.11) \\ \psi_k &= \text{the angle between } T_k^{\mathsf{T}} g_k \text{ and } R_k g_k. & (3.12) \end{cases}$$

Since $\nabla h_k(0) = R_k g_k$, ϕ_k is indeed the angle between d_k and $-\nabla h_k(0)$ as mentioned before. Analytically, $\phi_k = \arccos[-d_k^{\mathsf{T}}(R_k g_k)/(\|d_k\|\|R_k g_k\|)]$, and ψ_k can be defined in a similar way. By convention, the angle between a zero vector and any other vector is defined to be $\pi/2$ so that we do not need to be concerned about zero denominators. These angles will turn out to be key factors that affect the value of ρ_k.

To see how ϕ_k and ψ_k come into play, let us make a quick estimation for ρ_k by geometry without analytical details. Being adequate for our later usage, we only consider the case where

$$\phi_k + \psi_k \;<\; \frac{\pi}{2}. \tag{3.13}$$

Taking the first-order approximations of the numerator and denominator in

$$\rho_k \;=\; \frac{f(x_k) - f(x_k + s_k)}{h_k(0) - h_k(d_k)}$$

while noting that $s_k = T_k d_k$ and $\nabla h_k(0) = R_k g_k$, we have

$$\rho_k \;\approx\; \frac{-(T_k d_k)^{\mathsf{T}} g_k}{-d_k^{\mathsf{T}}(R_k g_k)} \;=\; \frac{d_k^{\mathsf{T}}(T_k^{\mathsf{T}} g_k)}{d_k^{\mathsf{T}}(R_k g_k)}. \tag{3.14}$$

We will estimate the right-hand side of (3.14) in a geometrical fashion. For convenience, denote

$$\mathbf{r}_k \;=\; R_k g_k, \quad \mathbf{t}_k \;=\; T_k^{\mathsf{T}} g_k. \tag{3.15}$$

Then

$$\frac{d_k^{\mathsf{T}}(T_k^{\mathsf{T}} g_k)}{d_k^{\mathsf{T}}(R_k g_k)} \;=\; \frac{d_k^{\mathsf{T}}\mathbf{t}_k}{d_k^{\mathsf{T}}\mathbf{r}_k} \;=\; \frac{\mathbf{t}_k^{\mathsf{T}}\mathbf{r}_k}{\|\mathbf{r}_k\|^2}\frac{-d_k^{\mathsf{T}}\mathbf{t}_k}{\|d_k\|\|\mathbf{t}_k\|}\frac{\|d_k\|\|\mathbf{r}_k\|}{-d_k^{\mathsf{T}}\mathbf{r}_k}\frac{\|\mathbf{t}_k\|\|\mathbf{r}_k\|}{\mathbf{t}_k^{\mathsf{T}}\mathbf{r}_k} \;=\; \frac{\mathbf{t}_k^{\mathsf{T}}\mathbf{r}_k}{\|\mathbf{r}_k\|^2}\frac{\cos\varphi_k}{\cos\phi_k\cos\psi_k}, \tag{3.16}$$

where φ_k is the angle between $-d_k$ and $\mathbf{t}_k$. Now note a simple geometrical fact that

$$\varphi_k \;\le\; \phi_k + \psi_k, \tag{3.17}$$

which is illustrated in Figure 3. Since we assume $\phi_k + \psi_k < \pi/2$, the above inequality implies

$$\cos\varphi_k \;\ge\; \cos(\phi_k + \psi_k) \;=\; \cos\phi_k\cos\psi_k - \sin\phi_k\sin\psi_k. \tag{3.18}$$

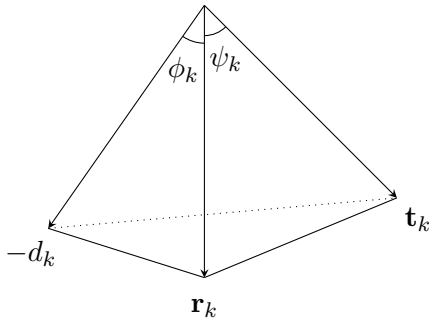
As $\cos\phi_k$ and $\cos\psi_k$ are both positive by (3.13), we obtain from (3.15), (3.16), and (3.18) that

$$\frac{d_k^{\mathsf{T}}(T_k^{\mathsf{T}} g_k)}{d_k^{\mathsf{T}}(R_k g_k)} \;\ge\; \frac{g_k^{\mathsf{T}} T_k R_k g_k}{\|R_k g_k\|^2}\left(1 - \tan\phi_k\tan\psi_k\right). \tag{3.19}$$

Estimations (3.14) and (3.19) lead us to

$$\rho_k \;\gtrapprox\; \frac{g_k^{\mathsf{T}} T_k R_k g_k}{\|R_k g_k\|^2}\left(1 - \tan\phi_k\tan\psi_k\right), \tag{3.20}$$

where we use "$\gtrapprox$" to signify the inexactness introduced by the first-order approximation (3.14). Lemma 3.1 will take this inexactness into account and formulates (3.20) precisely.



The angle between $-d_k$ and $\mathbf{t}_k$ cannot exceed $\phi_k + \psi_k$.

Figure 3: Illustration of (3.17).

Lemma 3.1 (Lower bound for reduction ratio)**.** *Under Assumptions 3.1–3.3, if $\phi_k + \psi_k < \pi/2$ and $\Delta_k \le \alpha\|R_k g_k\|/L_h$, then it holds for Algorithm 3.1 that*

$$\rho_k \;\ge\; \frac{g_k^{\mathsf{T}} T_k R_k g_k}{\|R_k g_k\|^2}\left(1 - \tan\phi_k \tan\psi_k - \frac{L_h \Delta_k}{\alpha\|R_k g_k\|}\right) - \frac{L_f \|T_k\|^2 \Delta_k}{\alpha\|R_k g_k\|}, \tag{3.21}$$

where ϕ_k and ψ_k are the angles defined in (3.11) and (3.12).

Proof. To estimate ρ_k rigorously, we only need to prove (3.18) and bound the inexactness in the first-order approximation (3.14). We will leave the former to Appendix A and focus on the latter. For convenience, denote

$$\bar{\rho}_k \;=\; \frac{d_k^{\mathsf{T}}(T_k^{\mathsf{T}} g_k)}{d_k^{\mathsf{T}}(R_k g_k)}.$$

Since $\phi_k + \psi_k < \pi/2$, we have $g_k^{\mathsf{T}} T_k R_k g_k > 0$ and $\tan\phi_k \tan\psi_k < \tan\phi_k \tan(\pi/2 - \phi_k) = 1$. Hence (3.19) guarantees that $\bar{\rho}_k > 0$. Recalling that $\nabla h_k(0) = R_k g_k$ and $s_k = T_k d_k$, we obtain by Taylor expansion that

$$\begin{aligned}
&\bar{\rho}_k[h_k(0) - h_k(d_k)] - [f(x_k) - f(x_k + s_k)] \\
&\qquad \le \bar{\rho}_k\left[-d_k^{\mathsf{T}}(R_k g_k) + \frac{L_h}{2}\|d_k\|^2\right] - \left[-(T_k d_k)^{\mathsf{T}} g_k - \frac{L_f}{2}\|T_k d_k\|^2\right] \\
&\qquad \le \frac{1}{2}(\bar{\rho}_k L_h + L_f\|T_k\|^2)\|d_k\|^2.
\end{aligned}$$

Recalling Assumption 3.3 while noting that $\nabla h_k(0) = R_k g_k$ and $\Delta_k \le \|R_k g_k\|/L_h$, we have

$$h_k(0) - h_k(d_k) \;\ge\; \frac{\alpha}{2}\|R_k g_k\|\Delta_k$$

Therefore,

$$\begin{aligned}
\bar{\rho}_k - \rho_k \;&= \frac{\bar{\rho}_k[h_k(0) - h_k(d_k)] - [f(x_k) - f(x_k + s_k)]}{h_k(0) - h_k(d_k)} \\
&\le \frac{(\bar{\rho}_k L_h + L_f\|T_k\|^2)\|d_k\|^2}{\alpha\|R_k g_k\|\Delta_k} \\
&\le \frac{\bar{\rho}_k L_h \Delta_k}{\alpha\|R_k g_k\|} + \frac{L_f\|T_k\|^2\Delta_k}{\alpha\|R_k g_k\|},
\end{aligned}$$

Hence

$$\rho_k \;\geq\; \bar{\rho}_k \left(1 - \frac{L_h \Delta_k}{\alpha \|R_k g_k\|}\right) - \frac{L_f \|T_k\|^2 \Delta_k}{\alpha \|R_k g_k\|}.$$

It remains to bound the right-hand side of the above inequality from below by invoking (3.19), whose left-hand side is $\bar{\rho}_k$. Since $1 - (L_h \Delta_k)/(\alpha \|R_k g_k\|) \geq 0$, we obtain from (3.19) that

$$\begin{aligned}
\bar{\rho}_k \left(1 - \frac{L_h \Delta_k}{\alpha \|R_k g_k\|}\right) &\geq \frac{g_k^T T_k R_k g_k}{\|R_k g_k\|^2} (1 - \tan\phi_k \tan\psi_k) \left(1 - \frac{L_h \Delta_k}{\alpha \|R_k g_k\|}\right) \\
&= \frac{g_k^T T_k R_k g_k}{\|R_k g_k\|^2} \left(1 - \tan\phi_k \tan\psi_k - \frac{L_h \Delta_k}{\alpha \|R_k g_k\|} + \tan\phi_k \tan\psi_k \frac{L_h \Delta_k}{\alpha \|R_k g_k\|}\right) \\
&\geq \frac{g_k^T T_k R_k g_k}{\|R_k g_k\|^2} \left(1 - \tan\phi_k \tan\psi_k - \frac{L_h \Delta_k}{\alpha \|R_k g_k\|}\right),
\end{aligned}$$

where the last line removes a non-negative term $(\tan\phi_k \tan\psi_k)(L_h \Delta_k)/(\alpha \|R_k g_k\|)$ from the one above it, leading to a lower bound. This completes the proof. □

It is worth noting that (3.19) is tight. To see this, consider a problem with dimension $n = 2$. Then (3.19) will become an equality when $R_k g_k$ lies between $-d_k$ and $T_k^T g_k$ (see Figure 3, where $\mathbf{r}_k = R_k g_k$ and $\mathbf{t}_k = T_k^T g_k$), which cannot be excluded without enforcing particular assumptions on matrices R_k, T_k and trust-region step d_k. Due to the tightness of (3.19), inequality (3.21) is *asymptotically tight* when $\Delta_k / \|R_k g_k\|$ (that is, $\Delta_k / \|\nabla h_k(0)\|$) becomes small.

3.2.4 Assumptions on the trust-region steps $\{d_k\}$ (II)

In addition to Assumption 3.3, we assume that d_k is approximately in the direction of $-\nabla h_k(0)$ when Δ_k becomes small compared with $\|\nabla h_k(0)\|$, as is specified in Assumption 3.4.

Assumption 3.4 (Angle assumption on trust-region steps)**.** *There exist constants $\beta \geq 0$ and $p > 0$ such that*

$$\sin\phi_k \;\leq\; \beta \left[\frac{\Delta_k}{\|\nabla h_k(0)\|}\right]^p \qquad \textit{for each} \quad k \geq 0, \tag{3.22}$$

where ϕ_k is the angle between d_k and $-\nabla h_k(0)$ as defined in (3.11).

The motivation to impose Assumption 3.4 is the $\tan\phi_k \tan\psi_k$ term in Lemma 3.1. Under this assumption, such a term is negligible when $\Delta_k / \|\nabla h_k(0)\|$ is sufficiently small, as long as ψ_k (the angle between $R_k g_k$ and $T_k^T g_k$) is uniformly bounded away from $\pi/2$.

We note that a similar assumption is postulated in [15, Theorem 3.2] for establishing the global convergence of trust-region methods using inaccurate gradient information, where it is assumed that the trust-region step will be approximately in the direction of the negative model gradient when the trust-region radius becomes small. The reason for [15, Theorem 3.2] to enforce such an assumption is the gradient inaccuracy. Similarly, the reason for us to postulate Assumption 3.4 is to accommodate the possible mismatch between $R_k g_k$ and $T_k^T g_k$. If $R_k g_k$ and $T_k^T g_k$ are always in the same direction, then $\tan\phi_k \tan\psi_k \equiv 0$ and Assumption 3.4 *can be waived*, which is the case when the transformations R_k and T_k are the transposes of each other. See Additive Schwarz in Subsection 5.5.2 for such an example. In the general case of Algorithm 3.1, R_k and T_k are not necessarily the transposes of each other and hence Assumption 3.4 is indispensable. As we will show in Section 6, our analysis can cover trust-region methods using

inaccurate gradient information, where the inaccurate gradient is regarded as a transformation of the accurate one. This gives an example for $T_k \neq R_k^T$ (T_k is identity while R_k is not; see Section 6 for details) and explains the similarity between Assumption 3.4 and the one in [15, Theorem 3.2]. See Appendix G for more discussions on how the behavior of R_k and T_k influences the necessity of Assumption 3.4.

In fact, to align better with [15, Theorem 3.2], we should change (3.22) to

$$\phi_k \to 0 \quad \text{when} \quad \Delta_k \to 0.$$

The motivation for us to assume the rate $\sin\phi_k = \mathcal{O}([\Delta_k/\|\nabla h_k(0)\|]^p)$ is the following observation, whose proof will be given in Appendix B. It tells us that Assumption 3.4 will hold automatically if d_k reduces h_k as much as $\bar{d}_k$ does. As remarked on page 15, it suffices to set d_k to the classical Cauchy step or any step that reduces h_k even further.

Proposition 3.1. *Under Assumption 3.2, if*

$$h_k(d_k) \;\leq\; h_k(\bar{d}_k)$$

with the $\bar{d}_k$ defined in (3.7), then Assumption 3.4 holds with $\beta = \sqrt{2L_h}$ and $p = 1/2$.

3.2.5 Assumptions on the transformations $\{R_k\}$ and $\{T_k\}$

It is not stringent to require that the transformations $\{T_k\}$ in Algorithm 3.1 remain bounded.

Assumption 3.5 (Boundedness of $\{T_k\}$)**.** *There exists a constant $\tau > 0$ such that $\|T_k\| \leq \tau$ for each $k \geq 0$.*

We require R_k to be uniformly non-degenerate in the direction of g_k in the following sense.

Assumption 3.6 (Non-degeneracy of $\{R_k\}$)**.** *There exists a constant $\kappa > 0$ such that*

$$\|R_k g_k\| \;\geq\; \kappa\|g_k\| \quad \textit{for each} \quad k \geq 0.$$

Since $\nabla h_k(0) = R_k g_k$, Assumption 3.6 can be equivalently stated as

$$\|\nabla h_k(0)\| \;\geq\; \kappa\|g_k\| \quad \text{for each} \quad k \geq 0,$$

meaning that the norm of model gradient at 0 should be at least a multiple of $\|\nabla f(x_k)\|$.

Although Algorithm 3.1 does not require R_k and T_k to be the transposes of each other, we do need them to have certain "consistency" along the direction of g_k. We can see this from Lemma 3.1 (note the $(g_k^T T_k R_k g_k)/\|R_k g_k\|^2$ term). Recall that the key to establishing the convergence of Algorithm 3.1 is to ensure that the trial step s_k can achieve $\rho_k > \eta_1$ when Δ_k is small enough. Since the estimation for ρ_k given in Lemma 3.1 is essentially tight, it is sensible to impose the following assumption.

Assumption 3.7 (Consistency between $\{R_k\}$ and $\{T_k\}$)**.** *There exists a constant $\lambda > 0$ such that*

$$\frac{g_k^T T_k R_k g_k}{\|R_k g_k\|^2} \;\geq\; \lambda \;>\; \eta_1 \quad \textit{for each} \quad k \geq 0.$$

Assumption 3.7 is indeed not new. It reduces to the conventional requirement that $\eta_1 < 1$ when R_k and T_k are always the transposes of each other like in classical trust-region methods. In [15], which investigates the global convergence of trust-region methods using inaccurate gradient information, it is assumed that the relative error in the inaccurate gradient is bounded by a constant $\zeta < 1 - \eta_1$ (see [15, inequality (7)]). Assumption 3.7 can be interpreted as an generalization of this assumption. Actually, if we regard the inaccurate gradient as a linear transformation of the accurate one, which we will do in Section 6, then the aforesaid bound on the relative error will guarantee Assumption 3.7 with $\lambda = 1 - \zeta > \eta_1$ (see Theorem 6.2).

As remarked after the proof of Lemma 3.1, the $(g_k^T T_k R_k g_k)/\|R_k g_k\|^2$ term is tight in the bound (3.21).Thus Assumption 3.7 is generally indispensable for the convergence of Algorithm 3.1, as will be demonstrated by Example 6.1 (see the comments succeeding the example).

Although Assumptions 3.5–3.7 involve g_k, they may be verified without knowing g_k. This is the case for the instances of Algorithm 3.1 in Sections 5 and 6. To fulfill Assumption 3.7, we have to obtain the value of λ for given matrices $\{R_k\}$ and $\{T_k\}$, or to choose $\{R_k\}$ and $\{T_k\}$ for a given λ. For the aforementioned instances, this is not a problem either.

It is worth noting that, under Assumptions 3.5–3.7, the angle ψ_k between $R_k g_k$ and $T_k^T g_k$ is always acute and is uniformly bounded away from $\pi/2$, because

$$\cos\psi_k \;=\; \frac{g_k^T T_k R_k g_k}{\|T_k^T g_k\|\|R_k g_k\|} \;=\; \frac{g_k^T T_k R_k g_k}{\|R_k g_k\|^2}\,\frac{\|R_k g_k\|}{\|T_k^T g_k\|} \;\geq\; \frac{\kappa\lambda}{\tau}. \tag{3.23}$$

Inequality (3.23) tells us that the constants τ, κ, and λ satisfy

$$\kappa\lambda \;\leq\; \tau. \tag{3.24}$$

Assumptions 3.5–3.7 also imply $\|T_k^T g_k\| \geq \kappa\lambda\|g_k\|$ and $\|R_k g_k\| \leq \lambda^{-1}\tau\|g_k\|$ for each $k \geq 0$. Hence, along the direction of g_k, T_k^T is uniformly non-degenerate and R_k is uniformly bounded.

3.2.6 A lower bound for the trust-region radius Δ_k

We have stated all the assumptions needed for the analysis of Algorithm 3.1, namely Assumptions 3.1–3.7. Such assumptions ensure $\rho_k > \eta_1$ when Δ_k is small enough compared with $\|\nabla h_k(0)\|$ (see Proposition 3.2). Consequently, the updating rule (3.3) for Δ_k renders a lower bound for Δ_k in terms of the length of model gradients (see Proposition 3.3).

Proposition 3.2. *Under Assumptions 3.1–3.7, we have $\rho_k > \eta_1$ in Algorithm 3.1 when*

$$\frac{\Delta_k}{\|\nabla h_k(0)\|} \;\leq\; \min\left\{\left[\frac{\kappa(\lambda-\eta_1)}{\sqrt{5}\tau\beta}\right]^{\frac{1}{p}},\; \frac{\alpha(\lambda-\eta_1)}{2(\lambda L_h + \tau^2 L_f)}\right\}. \tag{3.25}$$

Since $\nabla h_k(0) = R_k g_k$, the left-hand side of (3.25) can also be written as $\Delta_k/\|R_k g_k\|$. We display it as $\Delta_k/\|\nabla h_k(0)\|$ to stress the fact that such a value is the ratio between the trust-region radius and the length of the model gradient.

Proof. We claim that (3.25) guarantees

$$\tan\phi_k \tan\psi_k \;<\; \frac{\lambda-\eta_1}{2\lambda}. \tag{3.26}$$

Before proving this claim, let us show how (3.25) and (3.26) lead us to $\rho_k > \eta_1$.

The second term in (3.25) implies that $\Delta_k \le \alpha\|\nabla h_k(0)\|/L_h$, ensuring $0 \le \phi_k \le \pi/2$ according to (3.9). At the same time, (3.23) guarantees $0 \le \psi_k \le \pi/2$. In addition, inequality (3.26) entails trivially that $\tan\phi_k \tan\psi_k < 1$. Therefore,

$$\phi_k + \psi_k \;<\; \frac{\pi}{2}.$$

Hence, recalling that $\Delta_k \le \alpha\|\nabla h_k(0)\|/L_h$, we can invoke the bound in Lemma 3.1, namely

$$\rho_k \;\ge\; \frac{g_k^{\mathsf{T}} T_k R_k g_k}{\|R_k g_k\|^2}\left(1 - \tan\phi_k\tan\psi_k - \frac{L_h\Delta_k}{\alpha\|R_k g_k\|}\right) - \frac{L_f\|T_k\|^2\Delta_k}{\alpha\|R_k g_k\|}. \tag{3.27}$$

Meanwhile, by (3.26) and the second term in (3.25), and noting that $\nabla h_k(0) = R_k g_k$, we have

$$1 - \tan\phi_k\tan\psi_k - \frac{L_h\Delta_k}{\alpha\|R_k g_k\|} \;>\; 1 - \frac{\lambda-\eta_1}{2\lambda} - \frac{L_h}{\alpha}\cdot\frac{\alpha(\lambda-\eta_1)}{2(\lambda L_h + \tau^2 L_f)} \;>\; 1 - \frac{1}{2} - \frac{1}{2} \;=\; 0,$$

which allows us to push the estimation in (3.27) forward by replacing $g_k^{\mathsf{T}} T_k R_k g_k/\|R_k g_k\|^2$ with its lower bound λ in Assumption 3.7, and, trivially, replacing $\|T_k\|$ with its upper bound τ in Assumption 3.5, leading to

$$\rho_k \;\ge\; \lambda\left(1 - \tan\phi_k\tan\psi_k - \frac{L_h\Delta_k}{\alpha\|R_k g_k\|}\right) - \frac{\tau^2 L_f\Delta_k}{\alpha\|R_k g_k\|} \;=\; \lambda - \lambda\tan\phi_k\tan\psi_k - \frac{(\lambda L_h + \tau^2 L_f)\Delta_k}{\alpha\|R_k g_k\|}.$$

Invoking again (3.26) and the second term in (3.25), we arrive at

$$\rho_k \;>\; \lambda - \lambda\cdot\frac{\lambda-\eta_1}{2\lambda} - \frac{\lambda L_h + \tau^2 L_f}{\alpha}\cdot\frac{\alpha(\lambda-\eta_1)}{2(\lambda L_h + \tau^2 L_f)} \;=\; \eta_1.$$

It remains to verify (3.26). By Assumption 3.4 and (3.25), we have

$$\sin\phi_k \;\le\; \beta\left[\frac{\Delta_k}{\|\nabla h_k(0)\|}\right]^p \;\le\; \beta\cdot\frac{\kappa(\lambda-\eta_1)}{\sqrt{5}\tau\beta} \;=\; \frac{\kappa(\lambda-\eta_1)}{\sqrt{5}\tau}. \tag{3.28}$$

Since $\kappa\lambda \le \tau$ (see (3.24)), inequality (3.28) implies $\sin\phi_k < 1/\sqrt{5}$. Recalling that $0 \le \phi_k \le \pi/2$, we have $\cos\phi_k > 2/\sqrt{5}$, and consequently

$$0 \;\le\; \tan\phi_k \;<\; \frac{\kappa(\lambda-\eta_1)}{\sqrt{5}\tau}\cdot\frac{\sqrt{5}}{2} \;=\; \frac{\kappa(\lambda-\eta_1)}{2\tau}.$$

Meanwhile, (3.23) implies that $\tan\psi_k \le \tau/(\kappa\lambda)$. Thus (3.26) holds, completing the proof. □

To establish the desired lower bound for Δ_k, we let $\tilde{g}_k$ be a vector in $\{g_0, g_1, \ldots, g_k\}$ with

$$\|\tilde{g}_k\| \;=\; \min_{0\le\ell\le k}\|g_\ell\|.$$

Proposition 3.3. *Under Assumptions 3.1–3.7, it holds for Algorithm 3.1 that*

$$\Delta_k \;\ge\; \mu\|\tilde{g}_k\| \quad \textit{for each} \quad k \ge 0, \tag{3.29}$$

where

$$\mu \;=\; \min\left\{\frac{\Delta_0}{\|g_0\|},\; \gamma_0\kappa\left[\frac{\kappa(\lambda-\eta_1)}{\sqrt{5}\tau\beta}\right]^{\frac{1}{p}},\; \frac{\alpha\gamma_0\kappa(\lambda-\eta_1)}{2(\lambda L_h + \tau^2 L_f)}\right\}. \tag{3.30}$$

Proof. We prove by contradiction. Assume that the conclusion does not hold. Let $\bar{k}$ be the first integer such that $\Delta_{\bar{k}} < \mu\|\tilde{g}_{\bar{k}}\|$. Then

$$\Delta_{\bar{k}} \;<\; \frac{\Delta_0}{\|g_0\|}\|\tilde{g}_{\bar{k}}\| \;\leq\; \frac{\Delta_0}{\|g_0\|}\|g_0\| \;=\; \Delta_0,$$

implying that $\bar{k} \geq 1$. By the updating rule (3.3), we have

$$\Delta_{\bar{k}-1} \;\leq\; \gamma_0^{-1}\Delta_{\bar{k}} \;<\; \gamma_0^{-1}\mu\|\tilde{g}_{\bar{k}}\| \;\leq\; \gamma_0^{-1}\mu\|g_{\bar{k}-1}\|, \tag{3.31}$$

where the last inequality holds because $\|\tilde{g}_{\bar{k}}\| \leq \|g_{\bar{k}-1}\|$. Invoking Assumption 3.6, we obtain

$$\Delta_{\bar{k}-1} \;<\; \gamma_0^{-1}\kappa^{-1}\mu\|R_{\bar{k}-1}g_{\bar{k}-1}\| \;=\; \gamma_0^{-1}\kappa^{-1}\mu\|\nabla h_{\bar{k}-1}(0)\|.$$

According to the definition (3.30) of μ, we can check that the scenario of Proposition 3.2 happens at iteration $\bar{k}-1$. Thus $\rho_{\bar{k}-1} > \eta_1$. Hence (3.3) and the definition of $\bar{k}$ gives us

$$\Delta_{\bar{k}} \;\geq\; \Delta_{\bar{k}-1} \;\geq\; \mu\|\tilde{g}_{\bar{k}-1}\| \;\geq\; \mu\|\tilde{g}_{\bar{k}}\|,$$

contradicting the definition of $\bar{k}$. □

3.3 Key facts that lead to the convergence properties

In this subsection, we present two key lemmas that will lead us to the global convergence and worst-case complexity theory in Subsections 3.4 and 3.5.

The first lemma states that the predicted reduction of iteration k is at least a constant multiple of $\|\tilde{g}_k\|^2$.

Lemma 3.2 (Sufficient predicted reduction)**.** *Under Assumptions 3.1–3.7, Algorithm 3.1 fulfills*

$$h_k(0) - h_k(d_k) \;\geq\; \frac{1}{2}\alpha\kappa\mu\|\tilde{g}_k\|^2 \quad \textit{for each} \quad k \geq 0.$$

Proof. According to Assumption 3.6 Proposition 3.3,

$$\|\nabla h_k(0)\|\Delta_k \;=\; \|R_k g_k\|\Delta_k \;\geq\; \kappa\mu\|\tilde{g}_k\|^2.$$

Meanwhile, due to the third term in the definition (3.30) of μ, we have $\mu < \kappa/L_h$, and hence

$$\frac{\|\nabla h_k(0)\|^2}{L_h} \;=\; \frac{\|R_k g_k\|^2}{L_h} \;\geq\; \frac{\kappa^2}{L_h}\|g_k\|^2 \;\geq\; \kappa\mu\|\tilde{g}_k\|^2.$$

Thus Assumption 3.3 implies the desired reduction in h_k. □

The second lemma presented below reveals that, among the first K iterations ($K \geq 1$), a significant proportion of them can achieve $\rho_k > \eta_1$. Similar results have been used in the study of adaptive regularization methods (see [19, Theorem 2.1] and [7, Lemma 2.4] for instances).

Lemma 3.3 (Significant proportion of iterations with $\rho_k > \eta_1$)**.** *Let K be a positive integer and*

$$\mathcal{S}_K \;=\; \{k \in \mathbb{N} : 0 \leq k \leq K-1 \textit{ and } \rho_k > \eta_1\}, \tag{3.32}$$

ρ_k being the reduction ratio (3.1). Under Assumptions 3.1–3.7, it holds for Algorithm 3.1 that

$$K \;\leq\; \frac{\log(\gamma_2^{-1}\gamma_1)}{\log\gamma_1}|\mathcal{S}_K| + \frac{\log(\gamma_1\Delta_0^{-1}\mu\|\tilde{g}_{K-1}\|)}{\log\gamma_1}, \tag{3.33}$$

where μ is the positive constant defined in (3.30) and $|\mathcal{S}_K|$ denotes the cardinality of $\mathcal{S}_K$.

Before the proof, we observe that $\mathcal{S}_K$ only contains integers up to $K-1$, and the right-hand side of (3.33) involves $\|\tilde{g}_{K-1}\|$ rather than $\|\tilde{g}_K\|$ in its second term. Such facts will be instrumental when we invoke Lemma 3.3 (see the proof of Theorem 3.2 for instance). In addition, it will be important to recall that $0 < \gamma_1 < 1$ and hence $\log \gamma_1 < 0$ when proving and applying this lemma.

Proof. Noting that $\mu\|\tilde{g}_{K-1}\| \le \mu\|g_0\| \le \|\Delta_0\|$ by (3.30), and that $\log \gamma_1 < 0$, we have

$$\frac{\log(\gamma_1 \Delta_0^{-1} \mu \|\tilde{g}_{K-1}\|)}{\log \gamma_1} \ge \frac{\log(\gamma_1 \Delta_0^{-1} \Delta_0)}{\log \gamma_1} = 1. \tag{3.34}$$

Thus we can assume $K \ge 2$ when proving (3.33). Setting $K' = K-1$, it suffices to prove that [7]

$$K' \le \frac{\log(\gamma_2^{-1}\gamma_1)}{\log \gamma_1} |\mathcal{S}_{K'}| + \frac{\log(\Delta_0^{-1}\mu\|\tilde{g}_{K'}\|)}{\log \gamma_1}, \tag{3.35}$$

which will lead to (3.33) by adding 1 on both sides and noting that $|\mathcal{S}_{K'}| \le |\mathcal{S}_K|$.

According to (3.3), it holds that

$$\Delta_{k+1} \le \begin{cases} \gamma_1 \Delta_k & \text{if } k \notin \mathcal{S}_{K'}, \\ \gamma_2 \Delta_k & \text{if } k \in \mathcal{S}_{K'}. \end{cases} \tag{3.36}$$

Hence, with S denoting $|\mathcal{S}_{K'}|$, we have

$$\Delta_{K'} \le \gamma_1^{K'-S} \gamma_2^S \Delta_0 = \gamma_1^{K'} (\gamma_1^{-1}\gamma_2)^S \Delta_0.$$

Dividing both ends of the above inequality by Δ_0 and then taking logarithm, we have

$$\log(\Delta_0^{-1}\Delta_{K'}) \le K' \log \gamma_1 + S \log(\gamma_1^{-1}\gamma_2).$$

Dividing this inequality by $\log \gamma_1 < 0$ and then rearranging the terms, we arrive at

$$K' \le \frac{\log(\gamma_2^{-1}\gamma_1)}{\log \gamma_1} S + \frac{\log(\Delta_0^{-1}\Delta_{K'})}{\log \gamma_1}. \tag{3.37}$$

Recalling that $\Delta_{K'} \ge \mu\|\tilde{g}_{K'}\|$ according to Proposition 3.3, and that $\log \gamma_1 < 0$, we have

$$\frac{\log(\Delta_0^{-1}\Delta_{K'})}{\log \gamma_1} \le \frac{\log(\Delta_0^{-1}\mu\|\tilde{g}_{K'}\|)}{\log \gamma_1}.$$

Hence inequality (3.35) is true, which completes the proof. □

Note that Lemma 3.3 holds true even if $\mathcal{S}_K$ is empty, as we can see from the proof.

Apart from the lower bound (3.29), the proof of Lemma 3.2 relies only on Assumptions 3.3 and 3.6, while that of Lemma 3.3 only on the updating rule (3.3). The proofs do not refer to Assumptions 3.1, 3.2, 3.4, 3.5, or 3.7, yet Lemmas 3.2 and 3.3 inherit dependence on these assumptions from Proposition 3.3. This map of logic reliance will guide us to derive analogues of Lemmas 3.2 and 3.3 under other settings and hence establish global convergence and worst-case complexity theory when context changes. See Theorems 3.5, 3.6, 3.8, 4.1, and G.1 for examples.

[7] When $K = 1$ and $K' = 0$, the set $\mathcal{S}_{K'}$ is undefined, which is why we need to assume $K \ge 2$ for (3.35).

3.4 Global convergence

Based on Lemmas 3.2–3.3, we establish the following global convergence results for Algorithm 3.1. For the lim inf-type convergence (3.38) and the lim-type convergence (3.39) with a positive η_0, the proofs will only be briefly sketched because they are essentially the same as the known arguments for classical trust-region methods. In addition, the proof techniques for (3.39) will be fully illustrated in Subsection 3.6.3, where we prove a lim-type convergence result without requiring $\eta_0 > 0$ (Theorem 3.7).

Theorem 3.1. *Under Assumptions 3.1–3.7, it holds for Algorithm 3.1 that*

$$\liminf_{k\to\infty} \|g_k\| \;=\; 0. \tag{3.38}$$

Moreover, the lim*-type convergence can be guaranteed in each of the following two cases.*
1. If $\eta_0 > 0$, then

$$\lim_{k\to\infty} \|g_k\| \;=\; 0. \tag{3.39}$$

2. If f is convex and level-bounded,[8] *then*

$$\lim_{k\to\infty} f(x_k) \;=\; f_{\inf} \quad \textit{and} \quad \lim_{k\to\infty} \|g_k\| \;=\; 0. \tag{3.40}$$

Proof. Similar to the proof of [30, Theorem 6.4.5], we can prove (3.38) by examining the reduction of f over the iterations with $\rho_k > \eta_1$. Assume by contradiction that $\liminf_{k\to\infty} \|g_k\| > 0$. Then Lemma 3.2 ensures that each iteration with $\rho_k > \eta_1$ reduces f by an amount uniformly bounded from zero, and Lemma 3.3 implies that there exist infinitely many such iterations. Thus we have a contraction because f is bounded from below.

When $\eta_0 > 0$, one can follow the proof of [30, Theorem 6.4.6] to demonstrate (3.39) by the Thomas argument [136], a minor difference being that $\|x_k - x_{k+1}\|$ is bounded by $\tau\Delta_k$ instead of Δ_k due to the transformation T_k. See the proof of Theorem 3.7 for details of the argument.

We now prove (3.40) when f is convex and level-bounded. Consider the level set

$$\mathcal{L}(x_0) \;=\; \{x : f(x) \le f(x_0)\},$$

which is compact. According to the continuity of f, there exists a point $x_* \in \mathcal{L}(x_0)$ such that $f(x_*) = f_{\inf}$. The monotonicity of $\{f(x_k)\}$ guarantees that $\{x_k\} \subset \mathcal{L}(x_0)$. Therefore, by the convexity of f, it holds for each $k \ge 0$ that

$$f(x_k) - f_{\inf} \;=\; f(x_k) - f(x_*) \;\le\; g_k^{\mathsf{T}}(x_k - x_*) \;\le\; \|g_k\| D, \tag{3.41}$$

where D is the diameter of $\mathcal{L}(x_0)$. Hence (3.38) ensures $\liminf_{k\to\infty} f(x_k) = f_{\inf}$, which implies $f(x_k) \to f_{\inf}$ due to the monotonicity of $\{f(x_k)\}$. We then obtain $\|g_k\| \to 0$ from the fact that

$$f(x_k) - f_{\inf} \;\ge\; \frac{1}{2L_f}\|g_k\|^2 \quad \text{for each} \quad k \ge 0, \tag{3.42}$$

which is a consequence of Assumption 3.1 (independent of the convexity of f).[9] □

[8] An extended real-valued function f is level-bounded if for each $y \in \mathbb{R}$ the level set $\mathrm{lev}_y \equiv \{x : f(x) \le y\}$ is bounded (possibly empty) [125, Definition 1.8]. If f is convex, then it is level-bounded if and only if lev_α is bounded for a certain y larger than the infimum of f [124, Corollary 8.7.1].

[9] $f(x_k) - f_{\inf} \ge f(x_k) - f(x_k - g_k/L_f) \ge -(-g_k/L_f)^{\mathsf{T}} g_k - L_f\|-g_k/L_f\|^2/2 \;=\; \|g_k\|^2/(2L_f)$.

As we can see from the proof, given the convexity and lower-boundedness of f as well as Assumption 3.1, (3.40) is a consequence of $\liminf_{k\to\infty}\|g_k\| = 0$ and the monotonicity of $\{f(x_k)\}$ without further dependence on Algorithm 3.1.

According to the example by Yuan [154], the lim-type convergence (3.39) can fail for Algorithm 3.1 if the objective function is nonconvex and $\eta_0 = 0$, meaning that the algorithm accepts trial steps with simple decreases. Note that lim-type convergence can be achieved in the nonconvex and simple-decrease case for other variants of trust-region methods, an example being [32, Theorem 5.9], where the trust-region radius is updated in accordance with a station arty measure. We will establish such kind of results for Algorithm 3.1 in Subsection 3.6.3.

3.5 Worst-case complexity

3.5.1 Nonconvex case

For any positive constant $\epsilon < \|g_0\|$, define

$$k_\epsilon^g \;=\; \min\{k \in \mathbb{N} : \|g_k\| \le \epsilon\}.$$

Then k_ϵ^g is a positive integer, and

$$\|g_k\| \;>\; \epsilon \quad \text{for each} \quad k \in \{0, 1, \ldots, k_\epsilon^g - 1\}. \tag{3.43}$$

Using Lemma 3.33, we can derive the following bound for k_ϵ^g.

Theorem 3.2. *Under Assumptions 3.1–3.7, it holds for Algorithm 3.1 that*

$$k_\epsilon^g \;\le\; \frac{2\log(\gamma_2^{-1}\gamma_1)}{\alpha\kappa\mu\eta_1\log\gamma_1}(f_0 - f_{\inf})\epsilon^{-2} + \frac{\log(\gamma_1\Delta_0^{-1}\mu\epsilon)}{\log\gamma_1}, \tag{3.44}$$

where $f_0 = f(x_0)$, $f_{\inf} = \inf_{x\in\mathbb{R}^n} f(x)$, and μ is the positive constant defined in (3.30).

Proof. Denote $K = k_\epsilon^g$ and define $\mathcal{S}_K$ by (3.32) as in Lemma 3.3. According to (3.32) and the iterate updating scheme (3.2), for any $k \in \mathcal{S}_K$,

$$f(x_k) - f(x_{k+1}) \;=\; f(x_k) - f(x_k + s_k) \;\ge\; \eta_1[h_k(0) - h_k(d_k)] \;\ge\; \frac{1}{2}\alpha\kappa\mu\eta_1\epsilon^2,$$

where the last inequality is because of Lemma 3.2. Since $\{f(x_k)\}$ is non-increasing, we have [10]

$$\sum_{k\in\mathcal{S}_K}[f(x_k) - f(x_{k+1})] \;\le\; \sum_{k=0}^{\infty}[f(x_k) - f(x_{k+1})] \;\le\; f_0 - f_{\inf}.$$

These two inequalities imply that

$$|\mathcal{S}_K| \;\le\; \frac{2}{\alpha\kappa\mu\eta_1}(f_0 - f_{\inf})\epsilon^{-2}. \tag{3.45}$$

Combining (3.45) with Lemma 3.3, we obtain

$$K \;\le\; \frac{\log(\gamma_2^{-1}\gamma_1)}{\log\gamma_1}\cdot\frac{2}{\alpha\kappa\mu\eta_1}(f_0 - f_{\inf})\epsilon^{-2} + \frac{\log(\gamma_1\Delta_0^{-1}\mu\|\tilde{g}_{K-1}\|)}{\log\gamma_1}.$$

[10] If $\mathcal{S}_K = \emptyset$, we follow the convention that a sum over an empty set equals zero.

Therefore, we arrive at (3.44) by noting that

$$\frac{\log(\gamma_1\Delta_0^{-1}\mu\|\tilde{g}_{K-1}\|)}{\log\gamma_1} \;\leq\; \frac{\log(\gamma_1\Delta_0^{-1}\mu\epsilon)}{\log\gamma_1}, \tag{3.46}$$

which is true because $\log\gamma_1 < 0$ and $\|\tilde{g}_{K-1}\| > \epsilon$ according to (3.43). □

3.5.2 Convex case

When f is convex, it is of interest to find an upper bound for the positive integer

$$k_\epsilon^f \;=\; \min\{k \in \mathbb{N} : f(x_k) - f_{\inf} \leq \epsilon\}, \tag{3.47}$$

where $\epsilon < f(x_0) - f_{\inf}$ is a positive constant. For k_ϵ^g and k_ϵ^f, we have the following results.

Theorem 3.3. *If f is convex and level-bounded, then under Assumptions 3.1–3.7, it holds for Algorithm 3.1 that*

$$k_\epsilon^g \;\leq\; \frac{4D\log(\gamma_2^{-1}\gamma_1)}{\alpha\kappa\mu\eta_1\log\gamma_1}\epsilon^{-1} + \frac{\log(\gamma_1\Delta_0^{-1}\mu\epsilon)}{\log\gamma_1}, \tag{3.48}$$

$$k_\epsilon^f \;\leq\; \frac{4D^2\log(\gamma_2^{-1}\gamma_1)}{\alpha\kappa\mu\eta_1\log\gamma_1}\epsilon^{-1} + \frac{\log(\gamma_1 D^{-1}\Delta_0^{-1}\mu\epsilon)}{\log\gamma_1}, \tag{3.49}$$

where D is the diameter of the initial level set $\mathcal{L}(x_0) = \{x : f(x) \leq f(x_0)\}$, and μ is the positive constant defined in (3.30).

Before proving the theorem, we note that inequality (3.41) can be strengthened to

$$f(x_k) - f_{\inf} \;\leq\; D\|\tilde{g}_k\|. \tag{3.50}$$

Indeed, setting $\tilde{k}$ to be an integer in $\{0, 1, 2, \ldots, k\}$ such that $\|g_{\tilde{k}}\| = \min_{1\leq i\leq k}\|g_k\|$, we have

$$f(x_k) - f_{\inf} \;\leq\; f(x_{\tilde{k}}) - f_{\inf} \;\leq\; g_{\tilde{k}}^{\mathsf{T}}(x_{\tilde{k}} - x_*) \;\leq\; D\|g_{\tilde{k}}\| \;=\; D\|\tilde{g}_k\|.$$

Proof. We first derive (3.48) for k_ϵ^g. Denote $K = k_\epsilon^g$, and define $\mathcal{S}_K$ by (3.32) as in Lemma 3.3. We claim that

$$|\mathcal{S}_K| \;\leq\; \frac{4D}{\alpha\kappa\mu\eta_1\epsilon}. \tag{3.51}$$

Once (3.51) is justified, we can obtain (3.48) by plugging (3.51) into Lemma 3.3 and then noting (3.46), the arguments being similar to the proof of Theorem 3.2,

To prove (3.51), we assume that $\mathcal{S}_K \neq \emptyset$. Let $S = |\mathcal{S}_K|$, enumerate $\mathcal{S}_K$ in ascending order as

$$\mathcal{S}_K \;=\; \{k_1, k_2, \ldots, k_S\},$$

and define $k_{S+1} = k_S + 1$. For each $i \in \{1, 2, \ldots, S\}$, by (3.50) and the monotonicity of $\{f(x_k)\}$,

$$\|\tilde{g}_{k_i}\| \;\geq\; D^{-1}[f(x_{k_i}) - f_{\inf}] \;\geq\; D^{-1}\sum_{j=i}^{S}[f(x_{k_j}) - f(x_{k_j+1})] \;\geq\; D^{-1}\sum_{j=i}^{S}\eta_1[h_{k_j}(0) - h_{k_j}(d_{k_j})].$$

According to Lemma 3.2, the above inequality implies that

$$\|\tilde{g}_{k_i}\| \ \geq \ \frac{\alpha\kappa\mu\eta_1}{2D}\sum_{j=i}^{S}\|\tilde{g}_{k_j}\|^2.$$

Therefore, by (3.43) and Lemma D.1,

$$\epsilon \ < \ \min_{1\leq i\leq S}\|\tilde{g}_{k_i}\| \ \leq \ \frac{4D}{\alpha\kappa\mu\eta_1 S},$$

which implies (3.51) since $S = |\mathcal{S}_K|$. Thus we have proved (3.48).

Inequality (3.48) implies (3.49). Indeed, according to (3.41), $f(x_k) - f_{\inf} \leq \epsilon$ is fulfilled as long as $\|g_k\| \leq D^{-1}\epsilon$ is achieved, which implies that $k_\epsilon^f \ \leq \ k^g_{D^{-1}\epsilon}$. □

3.5.3 Strongly convex case

Now we consider the case that f is ς-strongly convex for a positive constant ς, namely

$$f(y) \ \geq \ f(x) + (y-x)^{\mathsf{T}}\nabla f(x) + \frac{\varsigma}{2}\|y-x\|^2$$

for all x and y in $\mathbb{R}^n$. In this case, we know that there exists a unique point $x_* \in \mathbb{R}^n$ where f attains its infimum $f_{\inf}$, and that

$$f(x) - f_{\inf} \ \leq \frac{1}{2\varsigma}\|\nabla f(x)\|^2, \tag{3.52}$$

$$f(x) - f_{\inf} \ \geq \frac{\varsigma}{2}\|x - x_*\|^2 \tag{3.53}$$

for all $x \in \mathbb{R}^n$. Obviously, it holds that $\varsigma \leq L_f$.

In this part, besides k_ϵ^f and k_ϵ^g, we will also present the worst-case complexity in terms of

$$k_\epsilon^x \ = \ \min\{k \in \mathbb{N} : \|x_k - x_*\| \leq \epsilon\},$$

where $\epsilon < \|x_0 - x_*\|$ is a positive constant. We have the following bounds.

Theorem 3.4. *If f is ς-strongly convex, then under Assumptions 3.1–3.7, it holds for Algorithm 3.1 that*

$$k_\epsilon^f \ \leq \frac{\log(\gamma_2^{-1}\gamma_1)}{\alpha\kappa\mu\eta_1\varsigma\log\gamma_1}\log[(f_0 - f_{\inf})\epsilon^{-1}] + \frac{\log(\gamma_2^{-1}\gamma_1^2\Delta_0^{-1}\mu\sqrt{2\varsigma\epsilon})}{\log\gamma_1}, \tag{3.54}$$

$$k_\epsilon^g \ \leq \frac{2\log(\gamma_2^{-1}\gamma_1)}{\alpha\kappa\mu\eta_1\varsigma\log\gamma_1}\log\left[\sqrt{2L_f(f_0 - f_{\inf})}\epsilon^{-1}\right] + \frac{\log(\gamma_2^{-1}\gamma_1^2\Delta_0^{-1}\mu\sqrt{\varsigma/L_f}\,\epsilon)}{\log\gamma_1}, \tag{3.55}$$

$$k_\epsilon^x \ \leq \frac{2\log(\gamma_2^{-1}\gamma_1)}{\alpha\kappa\mu\eta_1\varsigma\log\gamma_1}\log\left[\sqrt{2\varsigma^{-1}(f_0 - f_{\inf})}\epsilon^{-1}\right] + \frac{\log(\gamma_2^{-1}\gamma_1^2\Delta_0^{-1}\mu\varsigma\epsilon)}{\log\gamma_1}, \tag{3.56}$$

where $f_0 = f(x_0)$, $f_{\inf} = \inf_{x\in\mathbb{R}^n} f(x)$, and μ is the positive constant defined in (3.30).

Proof. We first prove (3.54) for k_ϵ^f. Denote $K = k_\epsilon^f$, and define $\mathcal{S}_K$ by (3.32) as in Lemma 3.3. To invoke Lemma 3.3 later, we note that

$$\|\tilde{g}_{K-1}\| \;>\; \sqrt{2\varsigma\epsilon}, \tag{3.57}$$

which holds since $\|\tilde{g}_{K-1}\| = \min_{0\le k\le K-1}\|g_k\|$ and, by (3.52) and the definition (3.47) of $k_\epsilon^f \equiv K$,

$$\|g_k\| \;\ge\; \sqrt{2\varsigma[f(x_k)-f_{\inf}]} \;>\; \sqrt{2\varsigma\epsilon} \quad \text{for each} \quad k\in\{0,1,\ldots,K-1\}.$$

With (3.57), we can demonstrate (3.54) by showing that

$$\big|\mathcal{S}_K\big| \;\le\; \frac{\log[(f_0-f_{\inf})\epsilon^{-1}]}{\alpha\kappa\mu\eta_1\varsigma}+1. \tag{3.58}$$

Indeed, if (3.58) holds, plugging (3.57)–(3.58) into Lemma 3.3, we will obtain (3.54) as follows:

$$\begin{aligned} k_\epsilon^f \;=\; K \;&\le\; \frac{\log(\gamma_2^{-1}\gamma_1)}{\log\gamma_1}\left[\frac{\log[(f_0-f_{\inf})\epsilon^{-1}]}{\alpha\kappa\mu\eta_1\varsigma}+1\right]+\frac{\log(\gamma_1\Delta_0^{-1}\mu\sqrt{2\varsigma\epsilon})}{\log\gamma_1}\\ &=\; \frac{\log(\gamma_2^{-1}\gamma_1)}{\alpha\kappa\mu\eta_1\varsigma\log\gamma_1}\log[(f_0-f_{\inf})\epsilon^{-1}]+\frac{\log(\gamma_2^{-1}\gamma_1^2\Delta_0^{-1}\mu\sqrt{2\varsigma\epsilon})}{\log\gamma_1}. \end{aligned}$$

To prove (3.58), we assume that $\mathcal{S}_K\neq\emptyset$. Let $S=|\mathcal{S}_K|$, enumerate $\mathcal{S}_K$ in ascending order as

$$\mathcal{S}_K \;=\; \{k_1,\,k_2,\,\ldots,\,k_S\},$$

and define $k_{S+1}=k_S+1$. For each $i\in\{1,2,\ldots,S\}$, we will show that

$$f(x_{k_{i+1}})-f_{\inf} \;\le\; (1-\alpha\kappa\mu\eta_1\varsigma)[f(x_{k_i})-f_{\inf}]. \tag{3.59}$$

To this end, we note that $f(x_{k_{i+1}})\le f(x_{k_i+1})$ and exploit Lemma 3.2, obtaining

$$f(x_{k_i})-f(x_{k_{i+1}}) \;\ge\; f(x_{k_i})-f(x_{k_i+1}) \;\ge\; \eta_1[h_{k_i}(0)-h_{k_i}(d_{k_i})] \;\ge\; \frac{1}{2}\alpha\kappa\mu\eta_1\|\tilde{g}_{k_i}\|^2. \tag{3.60}$$

According to the definition of $\tilde{g}_{k_i}$, there exists a $\tilde{k}\in\{0,1,\ldots,k_i\}$ such that $\|g_{\tilde{k}}\| = \|\tilde{g}_{k_i}\|$. Inequality (3.52) and the monotonicity of $\{f(x_k)\}$ then lead us to

$$\|\tilde{g}_{k_i}\|^2 \;=\; \|g_{\tilde{k}}\|^2 \;\ge\; 2\varsigma[f(x_{\tilde{k}})-f_{\inf}] \;\ge\; 2\varsigma[f(x_{k_i})-f_{\inf}]. \tag{3.61}$$

Combining (3.60) and (3.61), we have

$$f(x_{k_i})-f(x_{k_{i+1}}) \;\ge\; \alpha\kappa\mu\eta_1\varsigma[f(x_{k_i})-f_{\inf}],$$

which implies (3.59) because $f(x_{k_{i+1}})-f_{\inf} = [f(x_{k_i})-f_{\inf}]-[f(x_{k_i})-f(x_{k_{i+1}})]$. With inequality (3.59) justified, we have [11]

$$\log\left[\frac{f(x_{k_{i+1}})-f_{\inf}}{f(x_{k_i})-f_{\inf}}\right] \;\le\; \log(1-\alpha\kappa\mu\eta_1\varsigma) \;\le\; -\alpha\kappa\mu\eta_1\varsigma \qquad \text{for each} \quad i\in\{1,2,\ldots,S\}. \tag{3.62}$$

[11] We note that inequality (3.59) itself implies that $\alpha\kappa\mu\eta_1\varsigma<1$, because both $f(x_{k_{i+1}})-f_{\inf}$ and $f(x_{k_i})-f_{\inf}$ are positive numbers (recall our convention that g_k can never become zero within finite iterations). Hence it is legitimate to use $\log(1-\alpha\kappa\mu\eta_1\varsigma)$ in (3.62). One can also verify $\alpha\kappa\mu\eta_1\varsigma<1$ directly. Indeed, substituting the definition (3.30) of μ into $\alpha\kappa\mu\eta_1\varsigma$, we have $\alpha\kappa\mu\eta_1\varsigma\le\alpha\kappa\eta_1\varsigma\cdot\gamma_0\alpha\kappa(\lambda-\eta_1)/[2(\lambda L_h+\tau^2L_f)]$. Since $\alpha\le1$, $\gamma_0<1$, $\eta_1<\lambda$, and $\varsigma\le L_f$, it holds that $\alpha\kappa\mu\eta_1\varsigma<\kappa^2\lambda^2/\tau^2\le1$, where the last inequality is due to (3.24).

Note that $\epsilon < f(x_{k_S}) - f_{\inf}$ by the fact that $k_S \leq K-1$ and definition (3.47) of $k_\epsilon^f \equiv K$. Thus

$$\log[(f_0 - f_{\inf})^{-1}\epsilon] \;<\; \log\left[\frac{f(x_{k_S}) - f_{\inf}}{f(x_{k_1}) - f_{\inf}}\right] \;=\; \sum_{i=1}^{S-1} \log\left[\frac{f(x_{k_{i+1}}) - f_{\inf}}{f(x_{k_i}) - f_{\inf}}\right] \;\leq\; -\alpha\kappa\mu\eta_1\varsigma\,(S-1).$$

Hence,

$$S \;\leq\; \frac{\log[(f_0 - f_{\inf})\epsilon^{-1}]}{\alpha\kappa\mu\eta_1\varsigma} + 1,$$

which is identical to (3.58) since $S = |\mathcal{S}_K|$. Thus we have proved (3.54).

By (3.42), we know that $\|g_k\| \leq \epsilon$ if $f(x_k) - f_{\inf} \leq \epsilon^2/(2L_f)$. Therefore, $k_\epsilon^g \leq k^f_{\epsilon^2/(2L_f)}$, and hence (3.54) implies (3.55). Similarly, (3.53) shows that $\|x - x_*\| \leq \epsilon$ when $f(x_k) - f_{\inf} \leq \varsigma\epsilon^2/2$. Thus $k_\epsilon^x \leq k^f_{(\varsigma\epsilon^2)/2}$, and hence (3.54) implies (3.56). □

3.6 Comments on the update of trust-region radius

This subsection can be skimmed over during a first reading. For experts in trust-region methods and those who are concerned with implementations, we discuss more details on the updating scheme (3.3) for trust-region radius. We introduce frequently used implementations of the scheme, and show that (3.3) can be further generalized to encompass more possibilities without jeopardizing the theory in Subsections 3.4–3.5. In addition, Subsection 3.6.3 provides trust-region radius updating strategies that ensure $\|g_k\| \to 0$ without demanding convexity of f or sufficient decrease in trial step acceptance, while traditional results ensure only $\liminf \|g_k\| = 0$.

3.6.1 Flexibility in the updating scheme and typical implementations

The updating scheme (3.3) leaves considerable flexibility in practice. We stress that it is not necessary to set always $\Delta_{k+1} = \gamma_2\Delta_k$ when $\rho_k > \eta_1$. Practical implementations of Algorithm 3.1 may introduce two more parameters $\eta_2 \in [\eta_1, \infty]$ and $\Delta_{\max} \in [\Delta_0, \infty]$ and then set

$$\Delta_{k+1} \;=\; \begin{cases} \gamma_1\Delta_k & \text{if } \rho_k \leq \eta_1, \\ \Delta_k & \text{if } \eta_1 < \rho_k \leq \eta_2, \\ \min\{\Delta_{\max}, \gamma_2\Delta_k\} & \text{if } \rho_k > \eta_2, \end{cases} \tag{3.63}$$

which is a special instance of (3.3). On top of $\rho_k > \eta_2$, we can enforce further conditions for invoking $\Delta_{k+1} = \min\{\Delta_{\max}, \gamma_2\Delta_k\}$, for instance $\|d_k\| = \Delta_k$ as in [108, Algorithm 4.1], meaning that we expand the trust-region radius only if d_k actually reaches the boundary of the trust region. When both R_k and T_k are always the identity matrix (*i.e.*, classical trust-region methods), $\eta_1 = 1/4$ and $\eta_2 = 3/4$ are implemented in `LANCELOT` (see [29, Sections 3.2.4 and 3.3.4]). Typical values of γ_1 include $1/4$ and $1/2$, and setting γ_2 to its reciprocal is not uncommon.

For classical trust-region methods, [30, Page 782] points out that "a number of practical implementations" manage the trust region by

$$\Delta_{k+1} \;=\; \begin{cases} \gamma_1\|d_k\| & \text{if } \rho_k \leq \eta_1, \\ \Delta_k & \text{if } \eta_1 < \rho_k \leq \eta_2, \\ \max\{\Delta_k, \gamma_2\|d_k\|\} & \text{if } \rho_k > \eta_2, \end{cases} \tag{3.64}$$

instead of (3.63). For this scheme, [30, Section 17.1] suggests $\eta_1 = 0.05$ and $\eta_2 = 0.9$, while [61, Section 4.4] recommends [12] $\eta_1 = 0.0001$ and $\eta_2 = 0.99$. Scheme (3.64) is not a special case of (3.3). To cover (3.64), we consider

$$\Delta_{k+1} \in \begin{cases} [\gamma_0 \|d_k\|, \gamma_1 \Delta_k] & \text{if } \rho_k \leq \eta_1, \\ [\Delta_k, \gamma_2 \Delta_k] & \text{if } \rho_k > \eta_1, \end{cases} \tag{3.65}$$

which is used by Yuan in [154, Algorithm 1.1] and [156, Algorithm 2.1]. Scheme (3.65) encompasses both (3.64) and (3.3) as well as the update rule in [108, Algorithm 4.1] as particular instances. We will show that, even if Algorithm 3.1 adopts the more general rule (3.65), the convergence and complexity theory in Subsections 3.4–3.5 still holds after a minor modification on the definition of the constant μ.

Theorem 3.5. *Let Algorithm 3.1 update the trust-region radius Δ_k according to (3.65) instead of (3.3). Then Theorem 3.1 remains true, while Theorems 3.2–3.4 are still valid as long as we shift the definition of μ from (3.30) to*

$$\mu \;=\; \min\left\{ \frac{\Delta_0}{\|g_0\|}, \;\; \frac{\alpha\gamma_0\kappa}{3}\left[\frac{\kappa(\lambda-\eta_1)}{\sqrt{5}\tau\beta}\right]^{\frac{1}{p}}, \;\; \frac{\alpha^2\gamma_0\kappa(\lambda-\eta_1)}{6(\lambda L_h + \tau^2 L_f)} \right\}. \tag{3.66}$$

Comparing (3.66) with (3.30), we see they differ only by a factor of $\alpha/3$ in the last two terms. This factor comes from the following proposition, which by itself is an interesting observation on trust-region methods. It reveals that a trust-region step achieving sufficient model decrease cannot be much shorter than $\min\{\Delta_k, \|\nabla h_k(0)\|/L_h\}$. We prove it in Appendix C.

Proposition 3.4. *Assumptions 3.2 and 3.3 ensure*

$$\|d_k\| \;\geq\; \frac{\alpha}{3}\min\left\{\Delta_k, \;\; \frac{\|\nabla h_k(0)\|}{L_h}\right\} \qquad \textit{for each} \qquad k \geq 0.$$

Now we can prove Theorem 3.5.

Proof of Theorem 3.5. We only need to justify Lemmas 3.2 and 3.3 with μ taking the new definition (3.66). This is because the proofs of Theorems 3.1–3.4 depend only on these two lemmas without explicitly referring to the updating rule for Δ_k, as we can check in Subsections 3.4–3.5.

Bearing in mind the new rule (3.65) for updating Δ_k and the new definition of μ, we will now follow the road map set up in Subsections 3.2 and 3.3 to verify the propositions and arguments that lead to Lemmas 3.2 and 3.3.

Lemma 3.1 and Proposition 3.2 still hold because they concern only a single step of Algorithm 3.1 and hence do not depend on the update of Δ_k.

Proposition 3.3 is where Proposition 3.4 and the new scheme (3.65) for Δ_k come into play, and also where the new μ come into being. We assume that the conclusion does not hold, and let $\bar{k}$ be the first integer such that $\Delta_{\bar{k}} < \mu\|\tilde{g}_{\bar{k}}\|$. Then we can replicate the original proof of Proposition 3.3 until (3.31), which should now be replaced with

$$\|d_{\bar{k}-1}\| \;\leq\; \gamma_0^{-1}\Delta_{\bar{k}} \;<\; \gamma_0^{-1}\mu\|\tilde{g}_{\bar{k}}\| \;\leq\; \gamma_0^{-1}\mu\|g_{\bar{k}-1}\|$$

[12] Note that both [30, Algorithm 6.1.1] and [61, Algorithm 1] use $\rho_k > \eta_1$ as the criterion for the acceptance of trial steps. This partially explains why such small values of η_1 are needed.

according to the new updating rule (3.65). Combining this with Proposition 3.4, we have

$$\min\left\{\Delta_{\bar{k}-1},\ \frac{\|\nabla h_{\bar{k}-1}(0)\|}{L_h}\right\} \ < \ \frac{3}{\alpha}\gamma_0^{-1}\mu\|g_{\bar{k}-1}\|.$$

In other words,

$$\text{either} \quad \Delta_{\bar{k}-1} \ < \ \frac{3}{\alpha}\gamma_0^{-1}\mu\|g_{\bar{k}-1}\| \quad \text{or} \quad \frac{\|\nabla h_{\bar{k}-1}(0)\|}{L_h} \ < \ \frac{3}{\alpha}\gamma_0^{-1}\mu\|g_{\bar{k}-1}\|. \tag{3.67}$$

However, the second inequality above is false, since the last term in (3.66) tells us that

$$\frac{3}{\alpha}\gamma_0^{-1}\mu\|g_{\bar{k}-1}\| \ \le \ \frac{\alpha(\lambda-\eta_1)}{2(\lambda L_h+\tau^2 L_f)}\kappa\|g_{\bar{k}-1}\| \ \le \ \frac{\alpha(\lambda-\eta_1)}{2(\lambda L_h+\tau^2 L_f)}\|\nabla h_{\bar{k}-1}(0)\| \ < \ \frac{\|\nabla h_{\bar{k}-1}(0)\|}{L_h}.$$

Thus we obtain from the first inequality in (3.67) and the definition (3.66) of μ that

$$\Delta_{\bar{k}-1} \ < \ \frac{3}{\alpha}\gamma_0^{-1}\mu\|g_{\bar{k}-1}\| \ \le \ \min\left\{\kappa\left[\frac{\kappa(\lambda-\eta_1)}{\sqrt{5}\tau\beta}\right]^{\frac{1}{p}},\ \frac{\alpha\kappa(\lambda-\eta_1)}{2(\lambda L_h+\tau^2 L_f)}\right\}\|g_{\bar{k}-1}\|.$$

By Assumption 3.6, $\|g_{\bar{k}-1}\| \le \kappa^{-1}\|R_{\bar{k}-1}g_{\bar{k}-1}\| = \|\nabla h_{\bar{k}-1}(0)\|$. Therefore,

$$\Delta_{\bar{k}-1} \ < \ \min\left\{\left[\frac{\kappa(\lambda-\eta_1)}{\sqrt{5}\tau\beta}\right]^{\frac{1}{p}},\ \frac{\alpha(\lambda-\eta_1)}{2(\lambda L_h+\tau^2 L_f)}\right\}\|\nabla h_{\bar{k}-1}(0)\|,$$

which ensures $\rho_{\bar{k}-1} > \eta_1$ by Proposition 3.2. Hence (3.65) and the definition of $\bar{k}$ gives us

$$\Delta_{\bar{k}} \ \ge \ \Delta_{\bar{k}-1} \ \ge \ \mu\|\tilde{g}_{\bar{k}-1}\| \ \ge \ \mu\|\tilde{g}_{\bar{k}}\|,$$

a contradiction to the definition of $\bar{k}$. Proposition 3.3 is hence justified.

Given Proposition 3.3, the proof of Lemma 3.2 is still valid. That of Lemma 3.3 is also applicable, as (3.36) remains true under the new rule (3.65). The proof is complete. □

3.6.2 Setting trust-region radius to a multiple of model gradient size

For trust-region methods without space transformation, Fan and Yuan [48] proposed to define

$$\Delta_k \ = \ \Upsilon_k\|\nabla h_k(0)\| \tag{3.68}$$

and update Δ_k indirectly by adjusting Υ_k according to ρ_k in a fashion similar to (3.3), that is

$$\Upsilon_{k+1} \in \begin{cases}[\gamma_0\Upsilon_k,\ \gamma_1\Upsilon_k] & \text{if } \rho_k \le \eta_1,\\ [\Upsilon_k,\ \gamma_2\Upsilon_k] & \text{if } \rho_k > \eta_1,\end{cases} \tag{3.69}$$

where the initial value Υ_0 is an algorithmic parameter selected beforehand. See [48, Section 2] for their motivations from Levenberg-Marquardt method and [159, equations (2.11)–(2.12)] for the motivation from local convergence rate analysis. This method is further studied in [47], and is included as a special case in the investigations on nonlinear step size control [139, 63, 64]. In [33], worst-case complexity is established for a trust-region algorithm based on an updating

scheme of this type. In [65], a similar rule is followed in a trust-region method with decoupled first/second-order steps technique that guarantees convergence towards second-order stationarity with improved complexity. Similar ideas are also reflected in the so-called criticality step (see [32, Algorithm 4.2] and [3, Algorithm 3.1]) proposed by [31] before [48].

Even if Algorithm 3.1 manages the trust region by (3.68)–(3.69) instead of (3.3), the convergence and complexity theory in Subsections 3.4–3.5 still holds *without any modification*.

Theorem 3.6. *Let Algorithm 3.1 define the trust-region radius Δ_k by (3.68) and update it according to (3.69). Then Theorems 3.1–3.4 remain true.*

Proof. The global convergence results in Theorem 3.1 can be proved by classical arguments that will be completely included in the proof of Theorem 3.8 (see footnote 16). To establish the complexity bounds in Theorems 3.2–3.4, similar to Theorem 3.5, it suffices to justify Lemmas 3.2 and 3.3 based on the new rules (3.68)–(3.69). We will follow again the road map set up in Subsections 3.2 and 3.3 and verify the propositions and arguments that lead to the two lemmas.

Lemma 3.1 and Proposition 3.2 still hold because they concern only a single step of Algorithm 3.1 and are independent of the update of Δ_k.

For Proposition 3.3, we first note that Proposition 3.2 renders a lower bound for $\{\Upsilon_k\}_{k\geq 1}$. Indeed, since $\Delta_k = \Upsilon_k \|\nabla h_k(0)\|$, Proposition 3.2 tells us that $\rho_k > \eta_1$ whenever

$$\Upsilon_k \;\leq\; \min\left\{\left[\frac{\kappa(\lambda-\eta_1)}{\sqrt{5}\tau\beta}\right]^{\frac{1}{p}},\;\; \frac{\alpha(\lambda-\eta_1)}{2(\lambda L_h+\tau^2 L_f)}\right\}.$$

A similar argument as in the proof of Proposition 3.3 will lead us to

$$\Upsilon_k \;\geq\; \min\left\{\gamma_0\left[\frac{\kappa(\lambda-\eta_1)}{\sqrt{5}\tau\beta}\right]^{\frac{1}{p}},\;\; \frac{\gamma_0\alpha(\lambda-\eta_1)}{2(\lambda L_h+\tau^2 L_f)}\right\} \quad \text{when} \quad k\geq 1.$$

This inequality above implies

$$\Upsilon_k \;\geq\; \kappa^{-1}\min\left\{\gamma_0\kappa\left[\frac{\kappa(\lambda-\eta_1)}{\sqrt{5}\tau\beta}\right]^{\frac{1}{p}},\;\; \frac{\gamma_0\kappa\alpha(\lambda-\eta_1)}{2(\lambda L_h+\tau^2 L_f)}\right\} \;\geq\; \kappa^{-1}\mu \quad \text{when} \quad k\geq 1, \tag{3.70}$$

the last inequality being due to the definition (3.30) of μ. Since $\Delta_k = \Upsilon_k\|\nabla h_k(0)\| = \Upsilon_k\|R_k g_k\|$, recalling $\|R_k g_k\| \geq \kappa\|g_k\|$ according to Assumption 3.6, we have

$$\Delta_k \;=\; \Upsilon_k\|R_k g_k\| \;\geq\; \kappa^{-1}\mu\cdot\kappa\|g_k\| \;=\; \mu\|g_k\| \quad \text{when} \quad k\geq 1. \tag{3.71}$$

This is adequate for justifying (3.29), as $\Delta_0 \geq \mu\|g_0\|$ trivially due to the definition (3.30) of μ. Indeed, we have proved that $\Delta_k \geq \mu\|g_k\|$ for each $k\geq 0$, which is stronger than (3.29).

Given Proposition 3.3, the proof of Lemma 3.2 is still valid.

Now let us check Lemma 3.3. Recall that Lemma 3.3 defines $\mathcal{S}_K$ as

$$\mathcal{S}_K \;=\; \{k\in\mathbb{N} : 0\leq k\leq K-1 \text{ and } \rho_k > \eta_1\}.$$

Similar to (3.34), inequality (3.33) can be justified easily when $K=1$ by noting that $\mu\|g_0\|\leq\Delta_0$. Focusing on $K\geq 2$ and denoting $K' = K-1$, it suffices again to prove (3.35). As a counterpart to (3.36), we obtain from (3.69) that

$$\Upsilon_{k+1} \;\leq\; \begin{cases}\gamma_1\Upsilon_k & \text{if } k\notin\mathcal{S}_{K'},\\ \gamma_2\Upsilon_k & \text{if } k\in\mathcal{S}_{K'}.\end{cases} \tag{3.72}$$

With S denoting $|\mathcal{S}_{K'}|$, inequalities (3.70) and (3.72) lead to

$$\kappa^{-1}\mu \;\le\; \Upsilon_{K'} \;\le\; \gamma_1^{K'-S}\gamma_2^{S}\Upsilon_0 \;=\; \gamma_1^{K'}(\gamma_1^{-1}\gamma_2)^{S}\Upsilon_0.$$

Similar to (3.37), we divide both ends of this inequality by Υ_0, take logarithm, and then divide the resultant inequality by $\log\gamma_1 < 0$, arriving at

$$K' \;\le\; \frac{\log(\gamma_2^{-1}\gamma_1)}{\log\gamma_1}S + \frac{\log(\Upsilon_0^{-1}\kappa^{-1}\mu)}{\log\gamma_1}. \tag{3.73}$$

Since $\log\gamma_1 < 0$, and

$$\Upsilon_0^{-1}\kappa^{-1}\mu \;=\; \frac{\|R_0 g_0\|}{\Delta_0}\kappa^{-1}\mu \;\ge\; \frac{\kappa\|g_0\|}{\Delta_0}\kappa^{-1}\mu \;=\; \Delta_0^{-1}\mu\|g_0\| \;\ge\; \Delta_0^{-1}\mu\|\tilde{g}_{K'}\|, \tag{3.74}$$

we know that (3.73) implies (3.35) and hence justifies Lemma 3.3. The proof is complete. □

As we can see from (3.74), under the settings of Theorem 3.6, inequality (3.33) can indeed be strengthened to

$$K \;\le\; \frac{\log(\gamma_2^{-1}\gamma_1)}{\log\gamma_1}|\mathcal{S}_K| + \frac{\log(\gamma_1\Delta_0^{-1}\mu\|g_0\|)}{\log\gamma_1},$$

where $\|\tilde{g}_{K-1}\|$ is replaced with $\|g_0\|$. Consequently, for each bound in Theorems 3.2–3.4, the second term can be improved to $\mathcal{O}(1)$. Yet this will not change the overall order of the bounds.

3.6.3 Trust-region radius and lim-type convergence

As we mentioned in Subsection 3.4, the example by Yuan [154] shows that Algorithm 3.1 does not ensure $\|g_k\| \to 0$ if the objective function is nonconvex and $\eta_0 = 0$. This subsection provides a simple remedy for the problem reflected by Yuan's example. In Theorem 3.7, we show that Algorithm 3.1 can indeed guarantee $\|g_k\| \to 0$ even if $\eta_0 = 0$ as long as the trust-region radius is not expanded (yet may stay unchanged) when it is already large compared to the model gradient, where the criterion of being large can be any prescribed threshold. Similarly, for the variant of Algorithm 3.1 that maintains the trust-region radius by (3.68)–(3.69) instead of (3.3), Theorem 3.8 shows that $\|g_k\| \to 0$ is secured without requiring $\eta_0 > 0$ provided that Υ_k is not expanded when it is already larger than a prescribed bound.

Theorem 3.7 is inspired by [32, Theorem 5.9], while Theorem 3.8, to the best our knowledge, is new even for classical trust-region methods, *i.e.*, when R_k and T_k are always identity.

Theorem 3.7. *Let Algorithm 3.1 update Δ_k according to (3.3), and additionally impose*

$$\Delta_{k+1} \le \Delta_k \quad \textit{whenever} \quad \Delta_k > \bar{\mu}\|\nabla h_k(0)\| \tag{3.75}$$

with a constant $\bar{\mu} \in [0,\infty)$. Then under Assumptions 3.1–3.7, it holds for Algorithm 3.1 that

$$\lim_{k\to\infty}\|g_k\| \;=\; 0.$$

In Yuan's example, $\Delta_{3k} \to 1$, $\Delta_{3k+1} \to 3$, and $\|\nabla h_{3k}(0)\| \to 0$. Hence, for all large k, $\Delta_{3k+1} > \Delta_{3k}$ even though $\Delta_{3k} \gg \|\nabla h_{3k}(0)\|$. This is why $\|g_k\| \not\to 0$ in view of Theorem 3.7.

Imposing (3.75) means to demand $\Delta_k \leq \bar{\mu}\|\nabla h_k(0)\|$ as a prerequisite for expanding the trust-region radius. It differs however from enforcing $\bar{\mu}\|\nabla h_k(0)\|$ as an upper bound for Δ_k. One can indeed require Algorithm 3.1 to update Δ_k according to (3.3) and (3.75) with $\bar{\mu} = 0$. In that case, $\Delta_k > \bar{\mu}\|\nabla h_k(0)\|$ trivially for each $k \geq 0$, which refrains the algorithm from expanding the trust-region radius at all. A practical scheme that obeys both (3.3) and (3.75) is

$$\Delta_{k+1} \;=\; \begin{cases} \gamma_1 \Delta_k & \text{if } \rho_k \leq \eta_1, \\ \gamma_2 \Delta_k & \text{if } \rho_k > \eta_2 \text{ and } \Delta_k \leq \bar{\mu}\|\nabla h_k(0)\|, \\ \Delta_k & \text{else,} \end{cases} \tag{3.76}$$

where η_2 is in $[\eta_1, \infty]$ as before. This resembles the criticality-step strategy in [31] (see also [32, Algorithm 4.2] and [3, Algorithm 3.1]) in the sense that the update of Δ_k takes the size of model gradient into consideration. Another instance is [86, Algorithm 3.3.2], where $\Delta_{k+1} > \Delta_k$ only if $\|d_k\| = \Delta_k \leq \text{constant} \cdot \|\nabla h_k(0)\|$. We also mention that $\|d_k\| \equiv \Delta_k$ in Yuan's example. Hence requiring $\Delta_{k+1} \leq \Delta_k$ whenever $\|d_k\| < \Delta_k$ instead of (3.75) does not guarantee $\|g_k\| \to 0$.

Now we tailor the techniques in [32] to prove Theorem 3.7. As a preparation, we study sets [13]

$$\mathcal{S} \;=\; \{k \in \mathbb{N} : \rho_k > \eta_1\} \quad \text{and} \quad \mathcal{V} \;=\; \{k \in \mathbb{N} : \rho_k > \eta_1 \text{ and } \Delta_k \leq \bar{\mu}\|\nabla h_k(0)\|\}\,.$$

For these sets, (3.3) and (3.75) imply that

$$\Delta_{k+1} \begin{cases} \leq \; \gamma_1 \Delta_k & \text{if } k \in \mathbb{N} \setminus \mathcal{S}, \\ = \; \Delta_k & \text{if } k \in \mathcal{S} \setminus \mathcal{V}, \\ \geq \; \Delta_k & \text{if } k \in \mathcal{V}. \end{cases} \tag{3.77}$$

In addition to (3.77), we have the following reduction estimations:[14]

$$f(x_k) - f(x_{k+1}) \;\geq\; \begin{cases} \dfrac{1}{2}\alpha\eta_1 \min\left\{\Delta_k,\, \dfrac{\kappa\|g_k\|}{L_h}\right\} \kappa\|g_k\| & \text{if } k \in \mathcal{S}, \qquad (3.78) \\ \dfrac{1}{2}\alpha\eta_1 \min\left\{\Delta_k,\, \dfrac{\Delta_k}{L_h\bar{\mu}}\right\} \dfrac{\Delta_k}{\bar{\mu}} & \text{if } k \in \mathcal{V}. \qquad (3.79) \end{cases}$$

Such estimations hold because the iterate update scheme (3.2) and Assumption 3.3 imply

$$f(x_k) - f(x_{k+1}) \geq \eta_1[h_k(0) - h_k(d_k)] \geq \frac{1}{2}\alpha\eta_1 \min\left\{\Delta_k,\, \frac{\|\nabla h_k(0)\|}{L_h}\right\} \|\nabla h_k(0)\| \quad \text{for each} \quad k \in \mathcal{S},$$

leading to (3.78) by Assumption 3.6 and to (3.79) by the definition of $\mathcal{V}$. Combining (3.77) and (3.79) with the fact that $f(x_k) - f(x_{k+1}) \to 0$ (consequence of the lower boundedness of f in Assumption 3.1), we have Lemma 3.4. The argument of its proof is classical.

Lemma 3.4. *Let Algorithm 3.1 update Δ_k according to (3.3) and (3.75). Then under Assumptions 3.1–3.3, it holds for Algorithm 3.1 that*

$$|\mathbb{N} \setminus \mathcal{S}| + |\mathcal{V}| \;=\; \infty \quad \Longrightarrow \quad \lim_{k\to\infty} \Delta_k \;=\; 0.$$

[13] Note that the definition of $\mathcal{V}$ involves η_1 but not η_2. Indeed, η_2 is not included in the general form of (3.3) and (3.75). Scheme (3.76) is merely a special instance but not the general case.

[14] In (3.79), we interpret $\Delta_k/\bar{\mu}$ as ∞ if $\bar{\mu} = 0$. Indeed, when $\bar{\mu} = 0$, (3.79) becomes a vacuous truth since $\mathcal{V} = \emptyset$.

Proof. If $|\mathbb{N} \setminus \mathcal{S}| = \infty$ and $|\mathcal{V}| < \infty$, then $\Delta_k \to 0$ by (3.77). Consider the case with $|\mathcal{V}| = \infty$. Then (3.79) and the fact that $f(x_k) - f(x_{k+1}) \to 0$ imply $\Delta_k \to 0$ when $k \in \mathcal{V}$ and $k \to \infty$. Let $\ell_k = \min\{\ell \in \mathcal{V} : \ell \geq k\}$. Then ℓ_k is well defined for each $k \geq 0$ as $\mathcal{V}$ is infinite, and $\Delta_{\ell_k} \to 0$ when $k \to \infty$. In addition, $\Delta_k \leq \Delta_{\ell_k}$ according to the first two lines of (3.77), because $\{\ell : k \leq \ell \leq \ell_k - 1\} \subset \mathbb{N} \setminus \mathcal{V}$ by the definition of ℓ_k. Hence $\Delta_k \to 0$. □

Now we are ready to prove Theorem 3.7. The key is to examine two cases, namely $\Delta_k \not\to 0$ and $\Delta_k \to 0$. Condition (3.75) is needed only in the first case, where the proof is simple thanks to Lemma 3.4. In the second one, the lim-type convergence is a feature of Algorithm 3.1 with *no dependence* on (3.75), and the proof technique is the Thomas argument [136].

Proof of Theorem 3.7. We consider the following two cases.

Case 1. $\Delta_k \not\to 0$ when $k \to \infty$. According to Lemma 3.4, both $\mathbb{N} \setminus \mathcal{S}$ and $\mathcal{V}$ are finite sets. Hence $\mathcal{S}$ contains all but finitely many positive integers, and Δ_k remains constant for sufficiently large k according to (3.77). Thus (3.78) implies $\|g_k\| \to 0$ since $f(x_k) - f(x_{k+1}) \to 0$.

Case 2. $\Delta_k \to 0$ when $k \to \infty$. Given a positive number ε, we will show that $\|g_k\| \leq \varepsilon$ when k is sufficiently large. Since $\Delta_k \to 0$, there exists an integer $K \geq 0$ such that

$$\frac{\Delta_k}{\kappa\varepsilon/2} \;\leq\; \min\left\{\left[\frac{\kappa(\lambda-\eta_1)}{\sqrt{5}\tau\beta}\right]^{\frac{1}{p}},\; \frac{\alpha(\lambda-\eta_1)}{2(\lambda L_h + \tau^2 L_f)}\right\} \quad \text{for each} \quad k \geq K. \tag{3.80}$$

With this K, we define

$$\mathcal{K} \;=\; \left\{k \in \mathbb{N} : k \geq K \text{ and } \|g_k\| \geq \frac{\varepsilon}{2}\right\}.$$

Then $\|\nabla h_k(0)\| \geq \kappa\varepsilon/2$ for each $k \in \mathcal{K}$ according to Assumption 3.6. Consequently, (3.80) and Proposition 3.2 imply $\mathcal{K} \subset \mathcal{S}$. Hence we can specialize (3.78) to $\mathcal{K}$ and obtain

$$f(x_k) - f(x_{k+1}) \;\geq\; \frac{1}{2}\alpha\eta_1 \min\left\{\Delta_k,\; \frac{\kappa\varepsilon}{2L_h}\right\}\frac{\kappa\varepsilon}{2} \;=\; \frac{1}{4}\alpha\eta_1\kappa\varepsilon\Delta_k \quad \text{for each} \quad k \in \mathcal{K},$$

where $\min\{\Delta_k, \kappa\varepsilon/(2L_h)\} = \Delta_k$ because (3.80) indeed implies $\Delta_k \leq \kappa\varepsilon/(4L_h)$. Therefore,

$$\sum_{k\in\mathcal{K}} \Delta_k \;\leq\; \frac{4}{\alpha\eta_1\kappa\varepsilon}\sum_{k\in\mathcal{K}}[f(x_k) - f(x_{k+1})] \;\leq\; \frac{4}{\alpha\eta_1\kappa\varepsilon}\sum_{k=0}^{\infty}[f(x_k) - f(x_{k+1})] \;\leq\; \frac{4[f(x_0) - f_{\inf}]}{\alpha\eta_1\kappa\varepsilon}.$$

Hence there exists an integer $K' \geq K$ such that [15]

$$\sum_{k\in\mathcal{K},\, k\geq K'} \Delta_k \;\leq\; \frac{\varepsilon}{2\tau L_f}, \tag{3.81}$$

where τ is the upper bound for $\{\|T_k\|\}$ in Assumption 3.5. It suffices to show that $\|g_k\| \leq \varepsilon$ for each integer $k \geq K'$. Let us consider such a k and set $\ell_k = \min\{\ell : \ell \geq k \text{ and } \|g_\ell\| < \varepsilon/2\}$, which is well defined because $\liminf_{k\to\infty} \|g_k\| = 0$ (Theorem 3.1). Then

$$\|g_{\ell_k}\| \;<\; \frac{\varepsilon}{2} \quad \text{and} \quad \{\ell \in \mathbb{N} : k \leq \ell \leq \ell_k - 1\} \;\subset\; \mathcal{K}.$$

[15] Inequality (3.81) is achievable even if $\mathcal{K}$ is finite (even empty).

We can conclude $\|g_k\| \leq \varepsilon$ from the L_f-Lipschitz continuity of ∇f and the fact that

$$\|x_k - x_{\ell_k}\| \;\leq\; \sum_{\ell=k}^{\ell_k-1} \|x_\ell - x_{\ell+1}\| \;\leq\; \sum_{\ell=k}^{\ell_k-1} \|s_\ell\| \;\leq\; \tau \sum_{\ell=k}^{\ell_k-1} \Delta_\ell \;\leq\; \tau \sum_{\ell\in\mathcal{K},\, \ell\geq K'} \Delta_\ell \;\leq\; \frac{\varepsilon}{2L_f},$$

which uses $\|s_\ell\| = \|T_\ell d_\ell\| \leq \tau\Delta_\ell$ in the third inequality and applies (3.81) in the last one. □

It is worth noting that Theorem 3.7 still holds if we replace (3.3) by the more general scheme (3.65), and the same proof is still applicable, because (3.77) is still guaranteed.

When Algorithm 3.1 defines and updates the trust-region radius according to (3.68)–(3.69) instead of (3.3), we have Theorem 3.8 as an analogue to Theorem 3.7.

Theorem 3.8. *Let Algorithm 3.1 manage Δ_k according to (3.68)–(3.69) instead of (3.3), and additionally impose*

$$\Upsilon_{k+1} \;\leq\; \Upsilon_k \quad \textit{whenever} \quad \Upsilon_k \;>\; \bar{\mu} \tag{3.82}$$

with a constant $\bar{\mu} \in [0, \infty)$. Then under Assumptions 3.1–3.7, it holds for Algorithm 3.1 that

$$\lim_{k\to\infty} \|g_k\| \;=\; 0.$$

To implement (3.68)–(3.69) with (3.82), we can for instance choose $\eta_2 \in [\eta_1, \infty]$ and set

$$\Upsilon_{k+1} \;=\; \begin{cases} \gamma_1\Upsilon_k & \text{if } \rho_k \leq \eta_1, \\ \gamma_2\Upsilon_k & \text{if } \rho_k > \eta_2 \text{ and } \Upsilon_k \leq \bar{\mu}, \\ \Upsilon_k & \text{else.} \end{cases}$$

In general, enforcing (3.82) does not ensure $\Upsilon_k \leq \bar{\mu}$. However, by a straightforward application of mathematical induction, one can show that (3.69) and (3.82) guarantee

$$\Upsilon_k \;\leq\; \bar{\Upsilon} \;\equiv\; \max\{\Upsilon_0,\, \gamma_2\bar{\mu}\} \quad \text{for each} \quad k \geq 0. \tag{3.83}$$

Now we give the proof of Theorem 3.8. It uses the bound (3.83) in its final step.

Proof of Theorem 3.8. Define

$$\mathcal{T} = \{k \in \mathbb{N} : \rho_k > \eta_1\} \cup \{0\}.$$

We note that $|\mathcal{T}| = \infty$, and [16]

$$\lim_{k\in\mathcal{T},\, k\to\infty} \|g_k\| \;=\; 0. \tag{3.84}$$

Indeed, if $|\mathcal{T}| < \infty$, then (3.69) would imply $\Upsilon_{k+1} \leq \gamma_1\Upsilon_k$ for all but finitely many k, which is absurd as $\gamma_1 < 1$ and $\{\Upsilon_k\}$ has a positive lower bound (see (3.70)). The limit (3.84) is because

$$f(x_k) - f(x_{k+1}) \;\geq\; \eta_1[h_k(0) - h_k(d_k)] \;\geq\; \frac{1}{2}\alpha\eta_1 \min\left\{\mu\|g_k\|,\, \frac{\kappa\|g_k\|}{L_h}\right\}\kappa\|g_k\| \quad \text{for} \quad k \in \mathcal{T}\setminus\{0\},$$

[16] Note that (3.84) ensures $\liminf \|g_k\| = 0$ for Algorithm 3.1 with (3.3) replace by (3.68)–(3.69). If we further assume $\eta_0 > 0$, the same argument shows $\|g_k\| \to 0$ as long as we change η_1 to η_0 when defining $\mathcal{T}$. If f is convex and lower-bounded, then $\liminf \|g_k\| = 0$ implies $f(x_k) \to f_{\inf}$ and $\|g_k\| \to 0$ as in the proof of Theorem 3.1.

which is rendered by Assumptions 3.3, 3.6 and the fact that $\Delta_k \geq \mu\|g_k\|$ (see (3.71)).

Now we set

$$\ell_k = \max\{\ell \in \mathcal{T} : \ell \leq k\} \quad \text{for each} \quad k \geq 0.$$

Then $\{\ell_k\}_{k\geq 0} \subset \mathcal{T}$ is well defined [17] and $\ell_k \to \infty$ (as $\{\ell_k\}$ is monotone and $\ell_k = k$ for $k \in \mathcal{T}$). So

$$\lim_{k\to\infty} \|g_{\ell_k}\| \;=\; 0.$$

We will complete the proof by showing there exists a constant $c \in (0, \infty)$ such that

$$\|g_k\| \;\leq\; c\|g_{\ell_k}\| \quad \text{for each} \quad k \geq 0. \tag{3.85}$$

Since ∇f is L_f-Lipschitz and $\|x_{k+1} - x_k\| \leq \|s_k\| = \|T_k d_k\| \leq \tau\Delta_k = \tau\Upsilon_k\|g_k\|$, we have

$$\|g_{k+1}\| \leq \|g_k\| + L_f\|x_{k+1} - x_k\| \leq (1 + \tau L_f \Upsilon_k)\|g_k\| \leq \|g_k\| \exp(\tau L_f \Upsilon_k) \quad \text{for each} \quad k \geq 0,$$

and consequently,

$$\|g_k\| \;\leq\; \|g_{\ell_k}\| \prod_{\ell=\ell_k}^{k-1} \exp(\tau L_f \Upsilon_\ell) \;=\; \|g_{\ell_k}\| \exp\left(\tau L_f \sum_{\ell=\ell_k}^{k-1} \Upsilon_\ell\right) \quad \text{for each} \quad k \geq 0. \tag{3.86}$$

For $k \in \mathcal{T}$, (3.86) reduces trivially to $\|g_k\| \leq \|g_{\ell_k}\|$ as $\ell_k = k$. Thus (3.85) is achieved if we can bound $\sum_{\ell=\ell_k}^{k-1} \Upsilon_\ell$ for $k \in \mathbb{N}\setminus\mathcal{T}$. For such a k, the definition of ℓ_k ensures $\{\ell_k + 1, \ldots, k\} \subset \mathbb{N}\setminus\mathcal{T}$, then the update scheme (3.69) implies $\Upsilon_{\ell+1} \;\leq\; \gamma_1\Upsilon_\ell$ for each $\ell \in \{\ell_k + 1, \ldots, k\}$, and hence

$$\sum_{\ell=\ell_k}^{k-1} \Upsilon_\ell \;\leq\; \Upsilon_{\ell_k} + \sum_{j=0}^{\infty} \gamma_1^j \Upsilon_{\ell_k+1} \;=\; \Upsilon_{\ell_k} + \frac{\Upsilon_{\ell_k+1}}{1-\gamma_1} \;\leq\; \frac{2-\gamma_1}{1-\gamma_1}\bar{\Upsilon},$$

the upper bound (3.83) coming into play in the final step. This completes the proof. □

3.6.4 Significance of the algorithmic parameter η_1

In Algorithm 3.1, be the trust region updated according to (3.3) or the more general rule (3.65), the algorithmic parameter η_1 is a threshold for ρ_k such that

$$\rho_k \;>\; \eta_1 \quad \text{guarantees} \quad \Delta_{k+1} \;\geq\; \Delta_k.$$

Note that Δ_{k+1} may *equal* Δ_k when $\rho_k > \eta_1$, which is illustrated in the concrete schemes (3.63) and (3.64). In the implementations of classical trust-region methods, $0 < \eta_1 \leq 1/2$ is typical.

As we have seen in the analysis of Algorithm 3.1, it is essential to ensure the compatibility between the value of η_1 and the geometrical properties of $\{R_k\}$ and $\{T_k\}$ in the sense of Assumption 3.7. The algorithm can fail to converge when such compatibility does not hold, as will be demonstrated by Example 6.1 (see the remarks made after the example). In contrast, the other algorithmic parameters η_0, γ_0, γ_1, and γ_2 are free from any constraints but the basic ones, namely $0 \leq \eta_0 \leq \eta_1$ and $0 < \gamma_0 \leq \gamma_1 < 1 \leq \gamma_2$.

[17] If we did not include 0 into $\mathcal{T}$, then ℓ_k would be undefined for $k < \min\mathcal{T}$ (but it would not lead to an essential difficulty because we could focus on $k \geq \min\mathcal{T}$). Including 0 into $\mathcal{T}$ circumvents such a problem.

In addition, it will turn out in Section 6 that η_1 has a significant impact upon the behavior of classical trust-region methods when the gradient information is inaccurate. Setting η_1 to a small value is favorable to the robustness of Algorithm 3.1. The influences of η_0, γ_0, γ_1, and γ_2, on the other hand, are not as strong under the settings of Section 6. See Subsections 6.3, 6.4, and 6.5 for details.

Similar remarks can be made when Algorithm 3.1 adopts (3.68)–(3.69) to manage the trust region radius, where η_1 works as a threshold for ρ_k such that $\rho_k > \eta_1$ ensures $\Upsilon_{k+1} \geq \Upsilon_k$.

4 Coarse space correction

In this section, we enrich the space transformation framework with another ingredient named coarse space correction. The primary motivation comes from space decomposition methods which we will study in Section 5. In that context, coarse space correction is indispensable for the scalability of algorithms. See Subsection 5.7 for details.

4.1 Incorporating coarse spaces into the space transformation framework

Algorithm 4.1 presents a framework for optimization by space transformation with coarse space. Compared to the space transformation framework in Algorithm 3.1, iteration k of this algorithm introduces a coarse space model H_k in addition to h_k. A coarse space correction δ_k is calculated by minimizing H_k, and the trial step s_k incorporates both d_k and δ_k. In practice, for instance, H_k can be either a low-dimensional simplification of f reflecting its overall geometry without fine structures, or an approximation to f along a subspace that is beneficial to explore.

Algorithm 4.1 Optimization by Space Transformation with Coarse Space (prototype)

Input: $x_0 \in \mathbb{R}^n$.

For $k = 0, 1, 2, \ldots$, iterate the following steps.

1. **Forward Transformation.** Define functions $h_k : \mathbb{R}^{N_k} \to \mathbb{R}$ and $H_k : \mathbb{R}^{M_k} \to \mathbb{R}$ satisfying
$$\nabla h_k(0) \;=\; R_k \nabla f(x_k) \quad \text{and} \quad \nabla H_k(0) \;=\; C_k \nabla f(x_k),$$
with some positive integers N_k, M_k and some matrices $R_k \in \mathbb{R}^{N_k \times n}$, $C_k \in \mathbb{R}^{M_k \times n}$.
2. **Minimization.** Calculate $d_k \approx \operatorname{argmin}\{h_k(d) \colon d \in \mathbb{R}^{N_k}\}$ and $\delta_k \approx \operatorname{argmin}\{H_k(\delta) \colon \delta \in \mathbb{R}^{M_k}\}$.
3. **Backward Transformation.** Define
$$s_k \;=\; \omega_k T_k d_k + (1 - \omega_k) C_k^{\mathsf{T}} \delta_k \tag{4.1}$$
with a certain matrix $T_k \in \mathbb{R}^{n \times N_k}$ and a certain weight $\omega_k \in (0, 1]$.
4. **Update.** Set x_{k+1} to either $x_k + s_k$ or x_k *depending on the quality of* s_k.

In Algorithm 4.1, the coarse space is $\operatorname{range}(C_k^{\mathsf{T}}) \subset \mathbb{R}^n$, but the **Minimization** phase formulates the corresponding subproblem in $\mathbb{R}^{M_k}$ for practicality (normally $M_k \ll n$), matrices C_k and C_k^{T} working as the forward and backward transformations between the coarse space and $\mathbb{R}^{M_k}$. Following the terminology of domain decomposition, we also call C_k and C_k^{T} the *restriction* and *prolongation* operators for the coarse space. Suggested by coarse grid techniques for PDEs, we intentionally set the two operators to be the transposes of each other, which is a notable distinction from R_k and T_k. We will discuss how to design the coarse space and its restriction/prolongation operators in Subsection 5.7.2 in the context of space decomposition.

Given C_k, to have a model H_k that satisfies $\nabla H_k(0) = C_k \nabla f(x_k)$, we can set $H_k(\delta)$ to any function that approximates $f(x_k + C_k^{\mathsf{T}}\delta)$ to the first order or above. A simple example is

$$H_k(\delta) \;=\; f(x_k) + \delta^{\mathsf{T}} C_k \nabla f(x_k) + \frac{1}{2}\delta^{\mathsf{T}}\Big[C_k B_k C_k^{\mathsf{T}}\Big]\delta \quad \text{with} \quad B_k \approx \nabla^2 f(x_k), \tag{4.2}$$

which approximates the second-order Taylor expansion of $f(x_k + C_k^{\mathsf{T}}\delta)$ around $d = 0$ and resembles the commonly used Galerkin approximation (*e.g.*, [41, Page 106]) in coarse grid techniques for PDEs. We can devise B_k in (4.2) by techniques including quasi-Newton [106, 146] and random sketching [113]. Note that $C_k B_k C_k^{\mathsf{T}}$ may be obtained by analyzing f in the coarse space as in [160, Section 2], [157, Section 3], and [158, Section 2] instead of forming B_k in $\mathbb{R}^n$ and multiplying matrices. When $M_k \ll n$, pursuing exact or inexact second or even higher-order information of $f(x_k + C_k^{\mathsf{T}}\delta)$ for $\delta \in \mathbb{R}^{M_k}$ can be much more economical than doing the same with f in $\mathbb{R}^n$, and setting $B_k = \nabla^2 f(x_k)$ or even $H_k(\delta) = f(x_k + C_k^{\mathsf{T}}\delta)$ may become affordable.

The weight ω_k in the **Backward Transformation** phase can be either set to a constant chosen *a priori* or determined adaptively, for example, by a line search. The only constraint is that $\{\omega_k\}$ should be bounded away from zero, which will be stated in Assumption 4.1.

As what we did for Algorithm 2.1 in Section 3, we incorporate trust regions into Algorithm 4.1 to guarantee global convergence, obtaining Algorithm 4.2.

Algorithm 4.2 Optimization by Space Transformation with Coarse Space (trust-region version)

Input: Identical to Algorithm 3.1.

For $k = 0, 1, 2, \ldots,$ iterate the following steps.

1. **Forward Transformation.** Identical to Algorithm 4.1.
2. **Minimization.** Calculate

$$d_k \approx \operatorname{argmin}\big\{h_k(d) : d \in \mathbb{R}^{N_k},\ \|d\| \le \Delta_k\big\} \quad \text{and} \quad \delta_k \approx \operatorname{argmin}\big\{H_k(\delta) : \delta \in \mathbb{R}^{M_k},\ \|\delta\| \le \Delta_k\big\}.$$

3. **Backward Transformation.** Identical to Algorithm 4.1.
4. **Update.** Update x_k and Δ_k according to (3.2) and (3.3) with

$$\rho_k \;=\; \frac{f(x_k) - f(x_k + s_k)}{\omega_k[h_k(0) - h_k(d_k)] + (1 - \omega_k)[H_k(0) - H_k(\delta_k)]}. \tag{4.3}$$

4.2 Analysis of the space transformation framework with coarse space

4.2.1 Assumptions and a lower bound for the reduction ratio

For the analysis of Algorithm 4.2, we adopt the assumptions that Section 3 postulates on f, $\{h_k\}$, $\{d_k\}$, $\{R_k\}$, and $\{T_k\}$. In addition, we impose Assumption 4.1 on the new ingredients $\{H_k\}$, $\{C_k\}$, $\{\delta_k\}$, and $\{\omega_k\}$.

Assumption 4.1 (Assumptions on $\{H_k\}$, $\{C_k\}$, $\{\delta_k\}$, and $\{\omega_k\}$)**.**

1. *For each $k \ge 0$, H_k is continuously differentiable in $\mathbb{R}^{M_k}$, and ∇H_k is L_H-Lipschitz continuous in $\mathbb{R}^{N_k}$ with a constant L_H independent of k.*
2. *There exists a positive constant ν such that $\|C_k\| \le \nu$ for each $k \ge 0$.*
3. *$H_k(\delta_k) \le H_k(0)$ for each $k \ge 0$.*

4. There exists a constant $\omega \in (0, 1]$ such that $\omega_k \geq \omega$ for each $k \geq 0$.

The assumption on $\{H_k\}$ is conventional, while the requirements on $\{C_k\}$ and $\{\delta_k\}$ are nearly trivial. Note that we do not demand δ_k to achieve any kind of sufficient decrease. Consequently, it is insecure to rely solely on $C_k^\mathsf{T}\delta_k$ to define the trial step s_k. That is why the weight of $T_k d_k$ in s_k has to stay bounded away from zero as elaborated in the assumption on $\{\omega_k\}$.

Special attention should be paid to (4.3), which defines the reduction ratio ρ_k of iteration k. Its denominator, namely the *predicted reduction*, is neither $h_k(0) - h_k(d_k)$ nor $H_k(0) - H_k(\delta_k)$ but a weighted sum of the two, the weights being the same as those in the definition (4.1) of the trial step s_k. The structure of ρ_k leads to the following lemma, which states that ρ_k is either not less than 1 or roughly as large as $[f(x_k) - f(x_k + T_k d_k)]/[h_k(0) - h_k(d_k)]$. This observation is critical for us to establish the global convergence of Algorithm 4.2.

Lemma 4.1. *Under Assumptions 3.1, 3.3, and 4.1, when $\Delta_k \leq \|R_k g_k\|/L_h$, Algorithm 4.2 fulfills*

$$\rho_k \;\geq\; \min\left\{1,\;\; \frac{f(x_k) - f(x_k + T_k d_k)}{h_k(0) - h_k(d_k)} - \frac{[L_H + (2\|T_k\|^2 + \|C_k\|^2)L_f]\Delta_k}{\alpha\,\omega\|R_k g_k\|}\right\}. \tag{4.4}$$

Proof. We first note that [18]

$$\frac{a}{b+c} \;\geq\; \min\left\{1,\;\; \frac{a-c}{b}\right\} \tag{4.5}$$

for any real numbers a, b, and c with $b > 0$ and $c \geq 0$. Therefore, setting

$$a \;=\; f(x_k) - f(x_k + s_k), \quad b \;=\; \omega_k[h_k(0) - h_k(d_k)], \quad c \;=\; (1-\omega_k)[H_k(0) - H_k(\delta_k)],$$

it suffices to show that

$$\frac{a-c}{b} \;\geq\; \frac{f(x_k) - f(x_k + T_k d_k)}{h_k(0) - h_k(d_k)} - \frac{[L_H + (2\|T_k\|^2 + \|C_k\|^2)L_f]\Delta_k}{\alpha\,\omega\|R_k g_k\|}. \tag{4.6}$$

Since $b = \omega_k[h_k(0) - h_k(d_k)]$, we have

$$\frac{f(x_k) - f(x_k + T_k d_k)}{h_k(0) - h_k(d_k)} - \frac{a-c}{b} \;=\; \frac{\omega_k[f(x_k) - f(x_k + T_k d_k)] + c - a}{b}. \tag{4.7}$$

To prove (4.6), we will overestimate the right-hand side of (4.7) by examining its denominator and numerator individually. Regarding the denominator b, according to Assumption 3.3 and the facts that $\Delta_k \leq \|R_k g_k\|/L_h$ and $\omega_k > \omega$ (item 4 of Assumption 4.1), we have

$$b \;=\; \omega_k[h_k(0) - h_k(d_k)] \;\geq\; \frac{\alpha\,\omega}{2}\|R_k g_k\|\Delta_k. \tag{4.8}$$

Now we consider the numerator. By Assumptions 3.1 and 4.1 (item 1), we have Taylor expansions

$$\begin{cases} \omega_k[f(x_k) - f(x_k + T_k d_k)] \;\leq\; -\omega_k g_k^\mathsf{T} T_k d_k + \dfrac{\omega_k L_f}{2}\|T_k d_k\|^2, & (4.9)\\[2ex] c \;\leq\; -(1-\omega_k)[\nabla H_k(0)]^\mathsf{T}\delta_k + \dfrac{(1-\omega_k)L_H}{2}\|\delta_k\|^2, & (4.10)\\[2ex] -a \;\leq\; g_k^\mathsf{T} s_k + \dfrac{L_f}{2}\|s_k\|^2. & (4.11) \end{cases}$$

[18] If $a > b + c$, then (4.5) is trivial. Otherwise, $(a-c)(b+c) = ab + (a-b-c)c \leq ab$, implying (4.5).

According to the fact that $s_k = \omega_k T_k d_k + (1-\omega_k) C_k^{\mathsf{T}} \delta_k$ (see (4.1)) and $\nabla H_k(0) = C_k g_k$, the first-order terms in (4.9)–(4.11) *add up to zero*. Hence, taking the sum of (4.9)–(4.11), we have

$$\begin{aligned} \omega_k[f(x_k) - f(x_k + T_k d_k)] + c - a \;&\leq \frac{\omega_k L_f}{2}\|T_k d_k\|^2 + \frac{(1-\omega_k)L_H}{2}\|\delta_k\|^2 + \frac{L_f}{2}\|s_k\|^2 \\ &\leq \frac{L_f}{2}\|T_k\|^2[\Delta_k]^2 + \frac{L_H}{2}[\Delta_k]^2 + \frac{L_f}{2}\|s_k\|^2, \end{aligned} \tag{4.12}$$

the second inequality being because $0 \leq \omega_k \leq 1$. By the definition of s_k and the convexity of $\|\cdot\|^2$,

$$\|s_k\|^2 \;\leq\; \omega_k\|T_k d_k\|^2 + (1-\omega_k)\|C_k^{\mathsf{T}}\delta_k\|^2 \;\leq\; \|T_k\|^2[\Delta_k]^2 + \|C_k\|^2[\Delta_k]^2. \tag{4.13}$$

Inequalities (4.12) and (4.13) lead us to

$$\omega_k[f(x_k) - f(x_k + T_k d_k)] + c - a \;\leq\; \Big(\frac{L_H}{2} + L_f\|T_k\|^2 + \frac{L_f}{2}\|C_k\|^2\Big)[\Delta_k]^2.$$

Combining this inequality with (4.7), (4.8), we obtain (4.6), which completes the proof. □

4.2.2 Global convergence and worst-case complexity

With Lemma 4.1 proved, we can proceed to establish the global convergence and worst-case complexity of Algorithm 4.2. According to the proofs in Subsections 3.4–3.6, we only need to establish for Algorithm 4.2 the counterparts of Lemmas 3.2 and 3.3 with μ defined below in (4.16). Precisely speaking, it suffices to prove

$$\omega_k[h_k(0) - h_k(d_k)] + (1-\omega_k)[H_k(0) - H_k(\delta_k)] \;\geq\; \frac{1}{2}\alpha\kappa\mu\|\tilde{g}_k\|^2 \quad \text{for each} \quad k \geq 0, \tag{4.14}$$

and

$$K \;\leq\; \frac{\log(\gamma_2^{-1}\gamma_1)}{\log\gamma_1}|\mathcal{S}_K| + \frac{\log(\gamma_1\Delta_0^{-1}\mu\|\tilde{g}_K\|)}{\log\gamma_1} \tag{4.15}$$

with

$$\mathcal{S}_K \;=\; \{k \in \mathbb{N} : 0 \leq k \leq K-1 \text{ and } \rho_k > \eta_1\}.$$

The left-hand side of (4.14) is the predicted reduction for iteration k of Algorithm 4.2. Its counterpart in Lemma 3.2 is $h_k(0) - h_k(d_k)$, which is the predicted reduction in the case of Algorithm 3.1.

Theorem 4.1. *Consider Algorithm 4.2 with $\eta_1 < 1$. Under Assumptions 3.1–3.7 and 4.1, Algorithm 4.2 enjoys the global convergence elaborated in Theorems 3.1 and 3.7, and it possesses the worst-case complexity bounds quantified in Theorems 3.2–3.4 with*

$$\mu \;=\; \omega\min\left\{\frac{\Delta_0}{\|g_0\|},\;\; \gamma_0\kappa\left[\frac{\kappa(\lambda-\eta_1)}{\sqrt{5}\tau\beta}\right]^{\frac{1}{p}},\;\; \frac{\alpha\,\omega\gamma_0\kappa(\lambda-\eta_1)}{2[\omega\lambda L_h + L_H + (2\tau^2+\nu^2)L_f]}\right\}. \tag{4.16}$$

Proof. We will establish (4.14) and (4.15) following the road map set up in Subsections 3.2 and 3.3. First of all, as the counterpart of Lemma 3.1, we claim that Algorithm 4.2 satisfies

$$\rho_k \;\geq\; \min\left\{1,\;\; \frac{g_k^{\mathsf{T}} T_k R_k g_k}{\|R_k g_k\|^2}\left(1 - \tan\phi_k\tan\psi_k - \frac{L_h\Delta_k}{\alpha\|R_k g_k\|}\right) - \frac{L_k\Delta_k}{\alpha\|R_k g_k\|}\right\} \tag{4.17}$$

provided that $\phi_k + \psi_k < \pi/2$ and $\Delta_k \leq \alpha\|R_k g_k\|/L_h$, where ϕ_k and ψ_k are still the angles defined in (3.11)–(3.12), and

$$L_k \;=\; \omega^{-1}(L_H + 2L_f\|T_k\|^2 + L_f\|C_k\|^2) + L_f\|T_k\|^2. \tag{4.18}$$

Indeed, it is straightforward to check that the lower bound presented in Lemma 3.1 still holds for $[f(x_k)-f(x_k+T_k d_k)]/[h_k(0)-h_k(d_k)]$, that is to say when $\phi_k + \psi_k < \pi/2$ and $\Delta_k \leq \alpha\|R_k g_k\|/L_h$, Algorithm 4.2 achieves

$$\frac{f(x_k) - f(x_k + T_k d_k)}{h_k(0) - h_k(d_k)} \;\geq\; \frac{g_k^T T_k R_k g_k}{\|R_k g_k\|^2}\left(1 - \tan\phi_k \tan\psi_k - \frac{L_h \Delta_k}{\alpha\|R_k g_k\|}\right) - \frac{L_f\|T_k\|^2\Delta_k}{\alpha\|R_k g_k\|}.$$

Hence (4.17) is true according to (4.4).

The second step is to prove that $\rho_k > \eta_1$ when Δ_k becomes sufficiently small. Since we assume $\eta_1 < 1$, it suffices to ensure that the second term on the right-hand side of (4.17) exceeds η_1. Following the proof of Proposition 3.2, one can see that $\rho_k > \eta_1$ when

$$\frac{\Delta_k}{\|\nabla h_k(0)\|} \;\leq\; \min\left\{\left[\frac{\kappa(\lambda-\eta_1)}{\sqrt{5}\tau\beta}\right]^{\frac{1}{p}},\; \frac{\alpha(\lambda-\eta_1)}{2(\lambda L_h + L)}\right\}, \tag{4.19}$$

where L is an upper bound for $\{L_k\}$ defined in (4.18).[19] Since $0 < \omega \leq 1$, $\|T_k\| \leq \tau$, and $\|C_k\| \leq \nu$, we can take

$$L \;=\; \omega^{-1}[L_H + (3\tau^2 + \nu^2)L_f].$$

Then we can establish a lower bound for the trust-region radius Δ_k. Similar to Proposition 3.3, we can prove that Algorithm 4.2 satisfies

$$\Delta_k \;\geq\; \mu'\|\tilde{g}_k\| \quad \text{for each} \quad k \geq 0 \tag{4.20}$$

with

$$\mu' \;=\; \min\left\{\frac{\Delta_0}{\|g_0\|},\; \gamma_0\kappa\left[\frac{\kappa(\lambda-\eta_1)}{\sqrt{5}\tau\beta}\right]^{\frac{1}{p}},\; \frac{\alpha\gamma_0\kappa(\lambda-\eta_1)}{2(\lambda L_h + L)}\right\}. \tag{4.21}$$

Now we are ready to prove (4.15). Following the proof of Lemma 3.2, we have

$$h_k(0) - h_k(d_k) \;\geq\; \frac{1}{2}\alpha\kappa\mu'\|\tilde{g}_k\|^2. \tag{4.22}$$

Since $\omega \leq \omega_k \leq 1$ and $H_k(0) - H_k(\delta_k) \geq 0$ (Assumption 4.1), it follows from (4.22) that

$$\omega_k[h_k(0) - h_k(d_k)] + (1-\omega_k)[H_k(0) - H_k(\delta_k)] \;\geq\; \frac{1}{2}\alpha\kappa\omega\mu'\|\tilde{g}_k\|^2 \quad \text{for each} \quad k \geq 0,$$

which is identical to (4.14) because $\mu = \omega\mu'$ according to (4.16) and (4.21).

To justify (4.15), we note that (4.20) leads to

$$K \;\leq\; \frac{\log(\gamma_2^{-1}\gamma_1)}{\log\gamma_1}|\mathcal{S}_K| + \frac{\log(\gamma_1\Delta_0^{-1}\mu'\|\tilde{g}_{K-1}\|)}{\log\gamma_1}$$

according to the proof of Lemma 3.3. This implies (4.15) because $\mu \leq \mu'$ and $\log\gamma_1 < 0$.

With (4.14) (sufficient predicted reduction) and (4.15) (significant proportion of iterations with $\rho_k > \eta_1$), we can then replicate the analysis in Subsections 3.4 and 3.5 to establish the global convergence and worst-case complexity for Algorithm 4.2 as desired. □

[19] Observe that the second term on the right-hand side of (4.17) is almost the same as the right-hand side of (3.21). The only difference is that $L_f\|T_k\|^2$ is replaced by L_k. Consequently, (4.19) takes the same form as (3.25) except that $\tau^2 L_f$ is replaced by L, while (4.21) duplicates (3.30) with $\tau^2 L_f$ changed to L as well.

4.3 Remarks

In Algorithm 4.2, the two trust-region subproblems in the **Minimization** phase take the same trust-region radius. This is only for the simplicity of the presentation. In practice, the H_k-subproblem can use a trust-region radius Δ'_k that may differ from Δ_k. In that case, while Δ_k is updated according to the reduction ratio ρ_k defined in (4.17), one can define Δ'_k in *any fashion* that ensures $\{\Delta'_k/\Delta_k\}$ bounded from above along the iterations. Our analysis will hold with minor modifications. Indeed, our proofs resort to the trust-region radius of the H_k-subproblem only in (4.13), where $\|s_k\|^2$ has to be dominated by a constant multiple of $[\Delta_k]^2$. Such a domination holds as long as $\sup_{k\geq 0} \Delta'_k/\Delta_k < \infty$.

As generalizations of (4.1) and (4.17), Algorithm 4.2 can also define

$$\begin{aligned} s_k &= \omega_k T_k d_k + \omega'_k C_k^{\mathsf{T}} \delta_k, \\ \rho_k &= \frac{f(x_k) - f(x_k + s_k)}{\omega_k[h_k(0) - h_k(d_k)] + \omega'_k[H_k(0) - H_k(\delta_k)]}, \end{aligned}$$

where ω_k and ω'_k are nonnegative weights that do not necessarily sum up to 1. Our analysis will hold with minor modifications as long as $\{\omega_k\}$ and $\{\omega'_k\}$ are bounded from above and there exists a positive constant ω such that $\omega_k \geq \omega$ for each $k \geq 0$. In fact, the constraint $\omega_k + \omega'_k = 1$ only affects (4.13) in our analysis. Without such a constraint, we will have

$$\|s_k\|^2 \;=\; \|\omega_k T_k d_k + \omega'_k C_k^{\mathsf{T}} \delta_k\|^2 \;\leq\; 2(\omega_k^2 \|T_k d_k\|^2 + \omega_k'^2 \|C_k^{\mathsf{T}} \delta_k\|^2) \;\leq\; 2(\omega_k^2 \|T_k\|^2 + \omega_k'^2 \|C_k\|^2)[\Delta_k]^2$$

instead of (4.13). Hence one can still dominate $\|s_k\|^2$ by a multiple of $[\Delta_k]^2$ provided that $\{\omega_k\}$ and $\{\omega'_k\}$ are bounded from above.

Before closing this section, we point out that Algorithm 4.1 can be interpreted as a special instance of Algorithm 2.1 by arguments similar to those in Subsection 5.4. However, we cannot cast the theory of Algorithm 2.1 to establish that of Algorithm 4.1, because it would necessitates assumptions much more stringent than Assumption 4.1.

5 Optimization by space decomposition

5.1 Introduction

High-dimensional optimization problems are emerging progressively from various areas such as statistics, data science, and artificial intelligence. They are challenging the optimization community and urging the development of parallel algorithms that are scalable and adapted to modern High Performance Computing architectures. Meanwhile, parallel computing has always been a major theme in the study of numerical partial differential equations (PDEs), where domain decomposition [130, 141, 41] is proven to be a successful methodology yielding highly scalable algorithms. Several critical techniques contribute to the scalability of such algorithms, Restricted Additive Schwarz [14] and coarse space correction [148] being two examples. Such techniques, however, are little explored in parallel optimization algorithms, unless the optimization problem is the discretization of an infinite-dimensional problem.

We will bridge this gap by proposing a space decomposition framework for problem (1.1), which extends the Additive Schwarz type domain decomposition methods to nonlinear optimization. Under this framework, we can explore both Restricted Additive Schwarz and coarse space

correction. Our approach is entirely algebraic and we do not assume the optimization problem discretizes an underlying infinite-dimensional problem.

Indeed, we will see that the space decomposition framework can be regarded as a special instance of the space transformation framework studied in Sections 2–3. Our investigation on the space decomposition framework will be a concrete instance of the discussion in Section 2–3.

This section is organized as follows. In Subsection 5.2, we introduce Additive Schwarz type methods in domain decomposition for linear systems. Subsection 5.3 extends these methods to the optimization problem (1.1), resulting in the space decomposition framework. Subsection 5.4 shows that such a framework is indeed a special instance of the space transformation framework studied in Section 2. The framework in Subsection 5.3 is only conceptual, and there are three critical questions concerning its implementation: how to configure the decomposition/synchronization matrices, how to guarantee the global convergence, and how to incorporate course space correction. The three subsequent subsections answer these questions. Subsection 5.5 elaborates Additive Schwarz type decomposition/synchronization matrices according to well-known strategies in domain decomposition, including Restricted Additive Schwarz (Subsection 5.5), Subsection 5.6 globalizes the space decomposition framework via trust regions by means of specializing the corresponding investigation in Section 3 on the space transformation framework. Coarse space correction is explored in Subsection 5.7 following the exploration in Section 4. Section 5.8 presents numerical experiments with a quite simple implementation of our space decomposition framework. By highly encouraging performance of Restricted Additive Schwarz and coarse space correction on nonlinear optimization problems, we demonstrate the potential of developing scalable parallel optimization algorithms based on our framework.

5.2 Additive Schwarz type methods for linear systems

To motivate our space decomposition framework for optimization, we briefly introduce the Additive Schwarz type methods in domain decomposition for linear systems [14, 51, 80].

Let us consider solving a linear system

$$\mathbf{A}\mathbf{u} \;=\; \mathbf{r} \tag{5.1}$$

by an iterative scheme $\mathbf{u}_{k+1} = \mathbf{u}_k + \Delta\mathbf{u}_k$. The correction step $\Delta\mathbf{u}_k$ should be an approximate solution to the residual system

$$\mathbf{A}\Delta\mathbf{u} \;=\; \mathbf{r} - \mathbf{A}\mathbf{u}_k. \tag{5.2}$$

We pursue such a solution following the *algebraic domain decomposition* methodology (see, for instance, [41, Section 1.3]). For illustrative purpose, we suppose system (5.1) is 4-dimensional with [20] $\mathbf{u} = [v\,;\,w\,;\,y\,;\,z]$, and that the entry-wise formulation of the residual system (5.2) is

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{pmatrix} \begin{pmatrix} \Delta v \\ \Delta w \\ \Delta y \\ \Delta z \end{pmatrix} \;=\; \begin{pmatrix} a_k \\ b_k \\ c_k \\ t_k \end{pmatrix}, \tag{5.3}$$

where $[a_k\,;b_k\,;\,c_k\,;\,t_k]$ represents the residual $\mathbf{r} - \mathbf{A}\mathbf{u}_k$. We group $[v\,;\,w\,;\,y\,;\,z]$ into two blocks

$$[v\,;\,w\,;\,y] \quad \text{and} \quad [w\,;\,y\,;\,z], \tag{5.4}$$

[20] As mentioned in the introduction, the MATLAB-style notation $[a\,;b\,;\cdots;c]$ denotes a *vertical* array with a, b,..., c being the entries.

which are *overlapping* at variables w and y. In domain decomposition for PDEs, overlaps can facilitate information exchange between different subdomains and improve the efficiency of algorithms [41, Section 1.5].

Grouping the variables as (5.4) amounts to decomposing $\mathbb{R}^4$ into two overlapping subspaces [21]

$$\mathcal{X}^1 \;=\; \mathrm{span}\{e_1,\, e_2,\, e_3\} \quad \text{and} \quad \mathcal{X}^2 \;=\; \mathrm{span}\{e_2,\, e_3,\, e_4\},$$

where e_j is the jth canonical coordinate vector.[22] The restriction of (5.3) to $\mathcal{X}^1$ is

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} \Delta v \\ \Delta w \\ \Delta y \end{pmatrix} \;=\; \begin{pmatrix} a_k \\ b_k \\ c_k \end{pmatrix}, \tag{5.5}$$

which seeks for a correction step in $\mathcal{X}^1$ so that the two sides of (5.3) match each other when orthogonally projected to $\mathcal{X}^1$. For simplicity, we suppose that the restricted residual system (5.5) has a unique solution $\big[\Delta v_k^1\,;\, \Delta w_k^1\,;\, \Delta y_k^1\big]$, which gives us a correction step

$$\mathbf{d}_k^1 \;=\; \big[\Delta v_k^1\,;\, \Delta w_k^1\,;\, \Delta y_k^1\,;\, 0\big] \;\in\; \mathcal{X}^1. \tag{5.6}$$

Similarly, we can obtain

$$\mathbf{d}_k^2 \;=\; \big[0\,;\Delta w_k^2\,;\, \Delta y_k^2\,;\, \Delta z_k^2\big] \;\in\; \mathcal{X}^2 \tag{5.7}$$

by solving

$$\begin{pmatrix} A_{22} & A_{23} & A_{24} \\ A_{32} & A_{33} & A_{34} \\ A_{42} & A_{43} & A_{44} \end{pmatrix} \begin{pmatrix} \Delta w \\ \Delta y \\ \Delta z \end{pmatrix} \;=\; \begin{pmatrix} b_k \\ c_k \\ t_k \end{pmatrix}. \tag{5.8}$$

It remains now to define a full-space correction step $\Delta\mathbf{u}_k$ using the subspace steps. Various strategies exist in domain decomposition literatures. The Additive Schwarz (AS) strategy takes the sum of the subspace correction as the full-space correction step, namely

$$\text{AS:}\quad \Delta\mathbf{u}_k \;=\; \big[\Delta v_k^1\,;\, \Delta w_k^1 + \Delta w_k^2\,;\, \Delta y_k^1 + \Delta y_k^2\,;\, \Delta z_k^2\big]. \tag{5.9}$$

However, this may lead to *surplus correction* to the overlapping variables w and y. To avoid such overcorrection, the Restricted Additive Schwarz (RAS) strategy restricts the subspace steps into

$$\big[\Delta v_k^1\,;\, \Delta w_k^1\,;\, 0\,;\, 0\big] \quad \text{and} \quad \big[0\,;\, 0\,;\, \Delta y_k^2\,;\, \Delta z_k^2\big]$$

to annihilate the overlap and then "glues" the restricted steps together, that is

$$\text{RAS:}\quad \Delta\mathbf{u}_k \;=\; \big[\Delta v_k^1\,;\, \Delta w_k^1\,;\, \Delta y_k^2\,;\, \Delta z_k^2\big]. \tag{5.10}$$

RAS (5.10) can be generalized to the Weighted Restricted Additive Schwarz (WRAS) strategy

$$\text{WRAS}:\quad \Delta\mathbf{u}_k \;=\; \big[\Delta v_k^1\,;\, (1-\mathsf{p})\Delta w_k^1 + \mathsf{p}\Delta w_k^2\,;\, \mathsf{q}\Delta y_k^1 + (1-\mathsf{q})\Delta y_k^2\,;\, \Delta z_k^2\big], \tag{5.11}$$

where p and q are weights between 0 and 1 (they may vary along the iterations). WRAS (5.11) avoids overcorrection by averaging the overlapping corrections, reducing to RAS when $\mathsf{p} = \mathsf{q} = 0$.

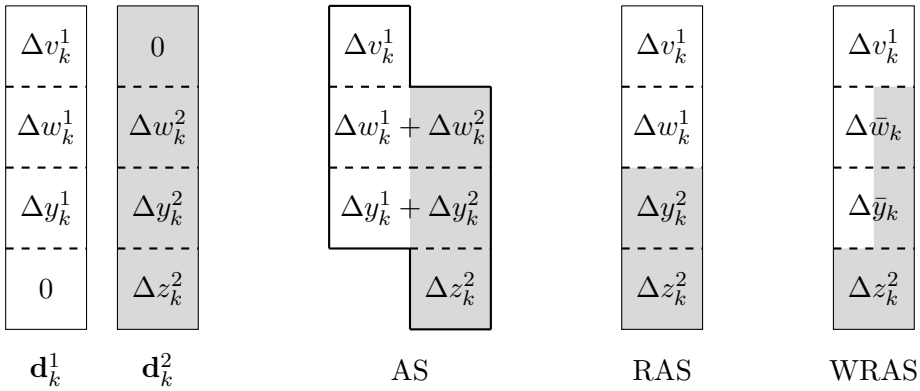


Figure 4: Illustration of AS (5.9), RAS (5.10), and WRAS (5.11).

We illustrate AS (5.9), RAS (5.10), and WRAS (5.11) in Figure 4, where, for WRAS (5.11), we denote $\Delta\bar{w}_k = (1-\mathsf{p})\Delta w_k^1 + \mathsf{p}\Delta w_k^2$ and $\Delta\bar{y}_k = \mathsf{q}\Delta y_k^1 + (1-\mathsf{q})\Delta y_k^2$.

There is another strategy called ASH (Additive Schwarz with Harmonic Extension) [14, 51]. This strategy first modifies the right-hand sides of (5.5) and (5.8) to

$$[a_k\,;\,b_k\,;\,0] \quad \text{and} \quad [0\,;\,c_k\,;\,t_k] \tag{5.12}$$

in order to suppress the overlap. It then solves the modified version of (5.5) and (5.8) to obtain subspace correction steps

$$\tilde{\mathbf{d}}_k^1 \;=\; \big[\Delta\tilde{v}_k^1\,;\,\Delta\tilde{w}_k^1\,;\,\Delta\tilde{y}_k^1\,;\,0\big] \quad \text{and} \quad \tilde{\mathbf{d}}_k^2 \;=\; \big[0\,;\,\Delta\tilde{w}_k^2\,;\,\Delta\tilde{y}_k^2\,;\,\Delta\tilde{z}_k^2\big], \tag{5.13}$$

and finally sets the full-space correction step to the sum of the two, namely

$$\text{ASH}: \quad \Delta\mathbf{u}_k \;=\; \big[\Delta\tilde{v}_k^1\,;\,\Delta\tilde{w}_k^1 + \Delta\tilde{w}_k^2\,;\,\Delta\tilde{y}_k^1 + \Delta\tilde{y}_k^2\,;\,\Delta\tilde{z}_k^2\big]. \tag{5.14}$$

We can define the weighted variant of ASH, named WASH, by replacing (5.12) with

$$\big[a_k\,;\,(1-\mathsf{p})b_k\,;\,\mathsf{q}c_k\big] \quad \text{and} \quad \big[\mathsf{p}b_k\,;\,(1-\mathsf{q})c_k\,;\,t_k\big],$$

where, as in WRAS (5.11), p and q are weights between 0 and 1. Another possibility, called RASH (Symmetrized Restricted Additive Schwarz) [14, 51], is to take

$$\text{RASH}: \quad \Delta\mathbf{u}_k \;=\; \big[\Delta\tilde{v}_k^1\,;\,\Delta\tilde{w}_k^1\,;\,\Delta\tilde{y}_k^2\,;\,\Delta\tilde{z}_k^2\big],$$

which can be regarded as the composite of RAS and ASH, explaining its code name RASH. One can also devise a weighted version of RASH by composing WRAS and WASH.

When there is no overlap, all the methods described above reduce to the block Jacobi Iteration. Here we focus on parallel (additive) methods, although there exist sequential (multiplicative) variants that are similar to the block Gauss-Seidel Iteration. We refer to [14, 51] for more

[21] Recall that we use superscripts to denote the indices for subspaces and objects that are corresponding to subspaces, such as N^i, R^i, T^i, h_k^i, $\mathbf{d}_k^i$, and Δ_k^i.

[22] Notably, the subscript j here is not an iteration counter. This does not follow our general convention about subscripts and superscripts but introduces no confusion.

general descriptions of AS, (W)RAS, (W)ASH, and RASH. See [80] for a survey on Additive Schwarz type domain decomposition methods, and [53, 57] for historical notes.

It is worth noting that, AS, (W)RAS, (W)ASH, and RASH are often used as preconditioners for other iterative algorithms like Krylov subspace methods (see preconditioners M_{RAS}^{-1} and M_{ASH}^{-1} and Algorithm 1 in [14]). Here we consider them as iterative solvers, which is the case in, for instances, [46], [53, Section 3.3], and [41, Section 1.7.1]. Indeed, these iterative solvers can also be regarded as the Richardson Iteration [126] preconditioned by the AS, (W)RAS, (W)ASH, and RASH preconditioners. We also note that, as an iterative solver, (W)RAS belongs to the multisplitting methods defined in [50, Definition 2.2], which are iterative schemes that proceed with multiple splittings of the coefficient matrix $\mathbf{A}$ in parallel.

In general, (W)RAS and (W)ASH *outperform* AS and RASH. According to [14, Remarks 2.4 and 2.5], RAS and ASH lead to preconditioners with similar efficiency (see [51, Theorem 6.1] for a theoretical explanation), and both of them are more efficient than the AS and RASH preconditioners. [46] and [53, Theorem 3.5] explain the superiority of RAS to AS by showing that the former, when applied to the discrete approximation of certain PDEs, is identical to the discretization of the continuous Parallel Schwarz method [23] by Lions [91, Section I.4] and consequently enjoys better convergence properties. Frommer and Szyld [51] show that (W)RAS and (W)ASH lead to convergent iterative schemes when $\mathbf{A}$ is a nonsingular M-matrix (see [51, Theorems 4.4, 6.2, and 6.4]), while it is not the case for RASH (see [51, Example 6.3]); subsequently, the authors suggest that RASH is "less attractive". We can also interpret the advantage of (W)RAS and (W)ASH over AS by observing the case where the coefficient matrix $\mathbf{A}$ of system (5.1) is nearly diagonal. For instance, $\mathbf{A} = \mathbf{D}+\mathbf{E}$, where $\mathbf{D}$ is a nonsingular diagonal matrix and $\|\mathbf{E}\| \ll \min_i |\mathbf{D}_{ii}|$. With such a coefficient matrix, decomposition methods should work well as the system is almost decoupled. It is indeed the case for (W)RAS and (W)ASH, which can generate nearly exact correction steps thanks to the presumed structure of $\mathbf{A}$, while AS, due to the very same structure, will obviously suffer from surplus corrections on the overlap.

5.3 A space decomposition framework for nonlinear optimization

We will extend the methods illustrated in Subsection 5.2 to optimization problem (1.1). We desire particularly a framework that encompasses (W)RAS and (W)ASH due to their good performance. Before doing this, note that for the full-space correction step $\Delta\mathbf{u}_k$ defined by RAS (5.10) or WRAS (5.11), we generally have

$$\Delta\mathbf{u}_k \notin \text{span}\{\mathbf{d}_k^1, \mathbf{d}_k^2, \mathbf{u}_k\},$$

where $\mathbf{d}_k^1$ and $\mathbf{d}_k^2$ are the subspace correction steps in (5.6)–(5.7). This fact distinguishes (W)RAS strikingly from the strategies in Parallel Variable Distribution [49, 131], Parallel Gradient Distribution [93], Parallel Variable Transformation [52], and Parallel Line Search Subspace Correction [42]. For (W)ASH, although the full-space correction step does lie in the linear span of the subspace ones (see (5.13)–(5.14)), none of these existing strategies incorporates a procedure equivalent to modifying the right-hand sides of (5.5) and (5.8), which, in optimization context, is a *first-order modification* to the objective function, as we will elaborate later.

[23] The name "Parallel Schwarz method" was coined by Gander [53]. The term used by Lions [91] was "parallel versions of the Schwarz alternating method".

To find a natural extension to the methods in Subsection 5.2, we consider a strongly convex quadratic function

$$f(\mathbf{u}) \;=\; \frac{1}{2}\mathbf{u}^{\mathsf{T}}\mathbf{A}\mathbf{u} - \mathbf{r}^{\mathsf{T}}\mathbf{u},$$

whose minimization is equivalent to solving $\mathbf{A}\mathbf{u} = \mathbf{r}$. Such equivalence enables us to examine the methods in Subsection 5.2 from the view point of optimization.

We still illustrate the idea by a 4-dimensional case as in Subsection 5.2. We observe that, at iteration k, all the Additive Schwarz type methods indeed define certain subspace models

$$h_k^i : \mathbb{R}^3 \to \mathbb{R} \quad (i = 1, 2), \tag{5.15}$$

calculate subspace steps [24]

$$d_k^i \;=\; \operatorname{argmin}\left\{h_k^i(d) : d \in \mathbb{R}^3\right\} \quad (i = 1, 2), \tag{5.16}$$

and then set

$$\Delta\mathbf{u}_k \;=\; \sum_{i=1}^{2} T^i d_k^i \tag{5.17}$$

with certain matrices $T^i \in \mathbb{R}^{4\times 3}$ $(i = 1, 2)$. To see this, we define matrices

$$U^1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \; U^2 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \; \tilde{U}^1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \; \tilde{U}^2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{5.18}$$

Recall that AS and RAS first solve (5.5) and (5.8), which is equivalent to minimizing

$$h_k^i(d) \;=\; \frac{1}{2}d^{\mathsf{T}}[U^i]^{\mathsf{T}}\mathbf{A}U^i d + d^{\mathsf{T}}[U^i]^{\mathsf{T}}(\mathbf{A}\mathbf{u}_k - \mathbf{r}) \quad (d \in \mathbb{R}^3,\, i = 1, 2). \tag{5.19}$$

With the solutions $d_k^1 = [\Delta v_k^1\,; \Delta w_k^1\,; \Delta y_k^1]$ and $d_k^2 = [\Delta w_k^2\,; \Delta y_k^2\,; \Delta z_k^2]$, AS and RAS then define $\Delta\mathbf{u}_k$ by (5.17) with $T^i = U^i$ and $T^i = \tilde{U}^i$ respectively. On the other hand, ASH and RASH obtain $d_k^1 = [\Delta\tilde{v}_k^1\,; \Delta\tilde{w}_k^1\,; \Delta\tilde{y}_k^1]$ and $d_k^2 = [\Delta\tilde{w}_k^2\,; \Delta\tilde{y}_k^2\,; \Delta\tilde{z}_k^2]$ by solving a modified version of (5.5) and (5.8) with right-hand sides (5.12), which is equivalent to minimizing

$$h_k^i(d) \;=\; \frac{1}{2}d^{\mathsf{T}}[U^i]^{\mathsf{T}}\mathbf{A}U^i d + d^{\mathsf{T}}[\tilde{U}^i]^{\mathsf{T}}(\mathbf{A}\mathbf{u}_k - \mathbf{r}) \quad (d \in \mathbb{R}^3,\, i = 1, 2). \tag{5.20}$$

Then ASH and RASH invoke (5.17) with $T^i = U^i$ and $T^i = \tilde{U}^i$ respectively. WRAS and WASH can be interpreted in the same way as RAS and ASH except that $\tilde{U}^1$ and $\tilde{U}^2$ are replaced with their weighted generalizations

$$W^1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1-\mathsf{p} & 0 \\ 0 & 0 & \mathsf{q} \\ 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad W^2 = \begin{pmatrix} 0 & 0 & 0 \\ p & 0 & 0 \\ 0 & 1-\mathsf{q} & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (0 \le \mathsf{p},\, \mathsf{q} \le 1), \tag{5.21}$$

[24] Note that the d_k^i defined by (5.16) is not identical to the $\mathbf{d}_k^i$ in (5.6)–(5.7). They are indeed linked with each other by $\mathbf{d}_k^i = U^i d_k^i$.

which reduce to $\tilde{U}^1$ and $\tilde{U}^2$ in (5.18) when $\mathsf{p} = \mathsf{q} = 0$.

Subspaces $\{\mathcal{X}^i\}_{i=1}^2$ do not appear explicitly in the process elaborated above. They are represented by $\mathbb{R}^3$. Indeed, h_k^i can be regarded as a model of $f(\mathbf{u}_k + d)$ in $\mathcal{X}^i$. However, for the sake of practicality, the model is formulated in $\mathbb{R}^3$ instead of $\mathcal{X}^i$. A critical feature of h_k^i is that

$$\nabla h_k^i(0) \;=\; R^i \nabla f(\mathbf{u}_k) \tag{5.22}$$

with a certain matrix R^i. Since $\nabla f(\mathbf{u}_k) = \mathbf{A}\mathbf{u}_k - \mathbf{r}$, we can see from (5.19) that $R^i = [U^i]^\mathsf{T}$ for AS and RAS, and from (5.20) that $R^i = [\tilde{U}^i]^\mathsf{T}$ for ASH and RASH. Similarly, WRAS sets $R^i = [U^i]^\mathsf{T}$ and WASH takes $R^i = [W^i]^\mathsf{T}$. Recalling the configurations of $\{T^i\}$ mentioned above, we note that R^i and T^i are generally not the transposes of each other except for AS and RASH, which are known to be less favorable than the other strategies.

Another particular feature of (W)ASH should be stressed: the subspace model functions of (W)ASH generally do not match the objective function in terms of first-order information, but impose a *first-order modification* to the objective function. For example, recall the h_k^i defined in (5.20) serves as a model of $f(\mathbf{u}_k + d)$ in $\mathcal{X}^i$ in the case of ASH. However,

$$\nabla h_k^i(0) \;=\; [\tilde{U}^i]^\mathsf{T}(\mathbf{A}\mathbf{u}_k - \mathbf{r}) \;=\; [\tilde{U}^i]^\mathsf{T}\nabla f(\mathbf{u}_k),$$

which differs from $[U^i]^\mathsf{T}\nabla f(\mathbf{u}_k)$ by excluding the derivative of f with respect to the third component of $\mathbf{u}$, and hence generally does not match the gradient of $f(\mathbf{u}_k + d)$ ($d \in \mathcal{X}^i$) at 0. The purpose of the first-order modification, as we observed in Subsection 5.2, is to avoid surplus correction on the overlap. Importantly, after the first-order modification is imposed, the full-space correction step in (W)ASH is defined by summing the subspace ones directly without any modification, in contrast to the RASH strategy that modifies the subspace corrections again on the overlap and leads to inadequate correction. To the best of our knowledge, there do not exist analogues to (W)ASH in space decomposition methods for general nonlinear optimization. We also note that (5.20) sets $\nabla h_k^i(0)$ using $\tilde{U}^i$ but defines $\nabla^2 h_k^i(0)$ with U^i. Therefore, model h_k^i excludes the derivative information of f with respect to the third component of $\mathbf{u}$, but the curvature information is retained. If $\nabla^2 h_k^i(0)$ were set to $[\tilde{U}^i]^\mathsf{T}\mathbf{A}\tilde{U}^i$, then the information of f in the third variable would be completely absent from h_k^i, missing the point of deploying an overlap between the two blocks of variables. Similarly, WASH uses W^i only to form $\nabla h_k^i(0)$, while $\nabla^2 h_k^i(0)$ is set to $[U^i]^\mathsf{T}\mathbf{A}U^i$ ($i = 1, 2$).

With (5.15)–(5.17) and (5.22), we can easily extend the Additive Schwarz type methods for the linear system (5.1) into a space decomposition framework for the optimization problem (1.1). We formalize the framework in Algorithm 5.1. This framework covers all the strategies introduced in Subsection 5.2. As far as we know, this is the first space decomposition framework that encompasses (W)RAS and (W)ASH for problem (1.1).

Algorithm 5.1 essentially decomposes $\mathbb{R}^n$ into subspaces $\{\mathcal{X}^i\}_{i=1}^m$, yet the decomposition does not appear explicitly. Instead, subspace $\mathcal{X}^i$ is represented by $\mathbb{R}^{N^i}$. The algorithm then works in each subspace (*in parallel*) to obtain subspace steps and updates the iterate by combining the subspace steps.

In the **Decomposition** phase of Algorithm 5.1, the (first-order) local information of f at the current iterate x_k is distributed from $\mathbb{R}^n$ to spaces $\{\mathbb{R}^{N^i}\}_{i=1}^m$ via matrices $\{R^i\}_{i=1}^m$, and then models $\{h_k^i\}_{i=1}^m$ are constructed according to such distributed information. The matrices $\{R^i\}_{i=1}^m$ can be configured according to the decomposition $\{\mathcal{X}^i\}_{i=1}^m$. See Subsection 5.5 for details. In the **Minimization** phase, for each $i \in \{1, 2, \ldots, m\}$, h_k^i is minimized approximately to obtain

Algorithm 5.1 Optimization by Space Decomposition (prototype)

Input: $x_0 \in \mathbb{R}^n$, $m \in \mathbb{N}_+$, $\{N^i\}_{i=1}^m \subset \mathbb{N}_+$; $R^i \in \mathbb{R}^{N^i \times n}$, $T^i \in \mathbb{R}^{n \times N^i}$ $(i = 1, 2, \ldots, m)$.

For $k = 0, 1, 2, \ldots$, iterate the following steps.

1. **Decomposition.** For each $i \in \{1, 2, \ldots, m\}$, define a function $h_k^i : \mathbb{R}^{N^i} \to \mathbb{R}$ satisfying
$$\nabla h_k^i(0) \;=\; R^i \nabla f(x_k).$$
2. **Minimization.** Calculate $d_k^i \approx \operatorname{argmin} \left\{ h_k^i(d) : d \in \mathbb{R}^{N^i} \right\}$ $(i = 1, 2, \ldots, m)$.
3. **Synchronization.** Define
$$s_k \;=\; \sum_{i=1}^m T^i d_k^i.$$
4. **Update.** Set x_{k+1} to either $x_k + s_k$ or x_k *depending on the quality of* s_k.

a step $d_k^i \in \mathbb{R}^{N^i}$. The exact sense of minimizing h_k^i is left vague and will be elaborated in Subsection 5.6 (see Algorithm 5.2 and Assumption 5.2). The **Synchronization** phase maps the subspace steps $\{d_k^i\}_{i=1}^m$ back to $\mathbb{R}^n$ by linear transformations $\{T^i\}_{i=1}^m$ and then sum them up to obtain a full-space step s_k in $\mathbb{R}^n$. The configuration of $\{T^i\}_{i=1}^m$, in conjugation with $\{R^i\}_{i=1}^m$, will be covered in Subsection 5.5. Finally, the **Update** phase updates x_k to x_{k+1} using s_k. This part is also left vague and will be detailed in Algorithm 5.2.

The **Synchronization** phase of Algorithm 5.1 is by design much simpler than the **Decomposition** and **Minimization** phases. Such a feature makes Algorithm 5.1 particularly suitable for parallel implementation, where the **Decomposition** and **Minimization** phases are naturally parallelizable while a complicated **Synchronization** phase would become a bottleneck. The **Update** phase most likely involves objective function evaluation (see Algorithm 5.2 for an implementation), which can be parallelized by exploiting particular problem structures such as partial separability [71, 27, 26].

Algorithm 5.1 specifies only the gradient of h_k^i, which is adequate for the convergence in terms of first-order stationarity (see Subsection 5.6.3). Although higher-order information such as $\nabla^2 h_k^i$ is not needed for the analysis, it will affect the practical performance of the algorithm. To fulfill $\nabla h_k^i(0) = R^i \nabla f(x_k)$, we can set $h_k^i(d)$ to any function that matches $f(x_k + [R^i]^\mathsf{T} d)$ to the first order. Similar to what Section 4 elaborated for the coarse space model H_k, one may set h_k^i to an approximate second-order Taylor expansion of $f(x_k + [R^i]^\mathsf{T} d)$ around $d = 0$, namely
$$h_k^i(d) = f(x_k) + d^\mathsf{T} R^i \nabla f(x_k) + \frac{1}{2} d^\mathsf{T} R^i B_k [R^i]^\mathsf{T} d \quad \text{with} \quad B_k \approx \nabla^2 f(x_k), \tag{5.23}$$
or even define $h_k^i(d) = f(x_k + [R^i]^\mathsf{T} d)$, both being probably far more affordable than doing the same in $\mathbb{R}^n$ if $N^i \ll n$. However, in contrast to the coarse space model, requiring $h_k^i(d)$ to match $f(x_k + [R^i]^\mathsf{T} d)$ beyond the first order is not always a good strategy. Indeed, as in (5.19), setting $h_k^i(d)$ to approximate $f(x_k + [R^i]^\mathsf{T} d)$ to high order is sensible if $\{R^i\}_{i=1}^m$ and $\{T^i\}_{i=1}^m$ are configured according to AS or (W)RAS (see Subsection 5.5 about how to do this), but it is not the case if $\{R^i\}_{i=1}^m$ and $\{T^i\}_{i=1}^m$ follow (W)ASH, as we can see from (5.20). The model $h_k^i(d)$ in (5.20) matches $f(x_k + [R^i]^\mathsf{T} d)$ to the first order but deliberately differs from it in the quadratic term. Instead of (5.23), which fits AS and (W)RAS, (5.20) suggests setting for (W)ASH
$$h_k^i(d) = f(x_k) + d^\mathsf{T} R^i \nabla f(x_k) + \frac{1}{2} d^\mathsf{T} [T^i]^\mathsf{T} B_k T^i d \quad \text{with} \quad B_k \approx \nabla^2 f(x_k),$$

which distinguishes itself from (5.23) due to the discrepancy between T^i and $[R^i]^\mathsf{T}$.

It is critical to observe the relation between the frameworks of space transformation (Algorithm 2.1) and space decomposition (Algorithm 5.1). Obviously, Algorithm 2.1 is a special case of Algorithm 5.1 with $m = 1$. Less obviously yet more importantly, Algorithm 5.1 can also be regarded as a special instance of Algorithm 2.1, as we mentioned in the introduction and will detail in Subsection 5.4. Such an observation reduces the investigation of the space decomposition framework to an application of what we have learnt about the space transformation framework. Following Section 3, we will incorporate trust regions into Algorithm 5.1 in order to guarantee global convergence. This will lead us to Algorithm 5.2, which is in turn a special case of Algorithm 3.1, the trust-region version of Algorithm 5.1. We will then establish the theory of Algorithm 5.2 by casting that of Algorithm 3.1 to this special case. Similarly, we can incorporate coarse space correction into Algorithm 5.1 according to the discussion in Section 4.

Before diving into the analysis, we will elaborate in Subsection 5.5 how to configure the matrices $\{R^i\}_{i=1}^m$ and $\{T^i\}_{i=1}^m$ in practice. We will extend the strategies in the 4-dimensional illustrative example to the n-dimensional case. Since $\nabla h_k^i(0) = R^i \nabla f(x_k)$, we can interpret $h_k^i(d)$ as a first-order model of $f(x_k + [R^i]^\mathsf{T} d)$. Similar to (2.1), one may expect that setting $T^i = [R^i]^\mathsf{T}$ is sensible. However, it is in general not the best choice, as we can infer from the quadratic example discussed in the beginning of the current subsection.

Remark 5.1. *For simplicity, the **Minimization** phase of Algorithm 5.1 denotes the variable of h_k^i by d* regardless of *k or i. By writing $h_k^i(d)$, we imply that $d \in \mathbb{R}^{N^i}$. Therefore, for the symbol d without any subscript or superscript, its dimension depends on whose variable it is standing for, which will always be clearly indicated by context. See also Remark 2.1 in Section 2.*

Remark 5.2. *One can generalize Algorithm 5.1 by allowing m, $\{R^i\}_{i=1}^m$, and $\{T^i\}_{i=1}^m$ to vary along the iterations. Our discussions can be easily extended to cover such a generalization.*

5.4 Space decomposition is a special instance of space transformation

Now we show that the space decomposition framework presented in Algorithm 5.1 is indeed a special instance of the space transformation framework given in Algorithm 2.1. Define [25]

$$N_k = \sum_{i=1}^m N^i. \tag{5.24}$$

For each $d \in \mathbb{R}^{N_k}$, decompose d as

$$d = \left[d^1 ; d^2 ; \cdots ; d^m\right] \in \mathbb{R}^{N_k} \tag{5.25}$$

so that $d^i \in \mathbb{R}^{N^i}$ $(i = 1, 2, \ldots, m)$. Define

$$h_k(d) = \sum_{i=1}^m h_k^i(d^i). \tag{5.26}$$

[25] Since Algorithm 5.1 fixes $\{N^i\}_{i=1}^m$ along the iterations (just for simplicity; see Remark 5.2), the N_k defined in (5.24) does not depend on k, but we keep the index k to align with the notation of Algorithm 2.1.

Then h_k is a function in $\mathbb{R}^{N_k}$. Indeed, it is *separable* with respect to the decomposition of d given in (5.25) (even though we do not assume the separability of f in $\mathbb{R}^n$). Hence

$$\nabla h_k(0) \;=\; \Big[\nabla h_k^1(0)\,;\,\nabla h_k^2(0)\,;\,\cdots\,;\,\nabla h_k^m(0)\Big]. \tag{5.27}$$

Furthermore, since $\nabla h_k^i(0) = R^i \nabla f(x_k)$, $\nabla h_k(0)$ is the following transformation of $\nabla f(x_k)$:

$$\nabla h_k(0) = R_k \nabla f(x_k)$$

with

$$R_k = \big[R^1\,;\,R^2\,;\,\cdots\,;\,R^m\big] \in \mathbb{R}^{N_k \times n}. \tag{5.28}$$

Now let

$$d_k = \big[d_k^1\,;\,d_k^2\,;\,\cdots\,;\,d_k^m\big] \in \mathbb{R}^{N_k}. \tag{5.29}$$

Since d_k^i approximately minimizes h_k^i, we can regard d_k as an approximate minimizer of h_k. In addition, the trial step s_k defined in Algorithm 5.1 can be written as $s_k = T_k d_k$ with

$$T_k = \big[T^1\,,\,T^2\,,\,\cdots\,,\,T^m\big] \in \mathbb{R}^{N_k \times n}. \tag{5.30}$$

Hence Algorithm 5.1 is an instance of Algorithm 2.1 with N_k, h_k, R_k, T_k, and d_k specified above.

5.5 Additive Schwarz type decomposition/synchronization matrices

We will refer to $\{R^i\}_{i=1}^m$ and $\{T^i\}_{i=1}^m$ in Algorithm 5.1 as *decomposition matrices* and *synchronization matrices* respectively. They correspond to the *restriction* and *prolongation* operators in domain decomposition. As mentioned in Subsection 5.3, these matrices can be chosen based on a decomposition of $\mathbb{R}^n$. Although Algorithm 5.1 is applicable to general decompositions of $\mathbb{R}^n$, this subsection focuses on the special case where $\mathbb{R}^n$ is decomposed according to a partition of the coordinate index set $\{1, 2, \ldots, n\}$. Under such a setting, we introduce how to define the decomposition/synchronization matrices according to the Additive Schwarz type methods in domain decomposition. Implementing Algorithm 5.1 with such matrices, we generalize the Additive Schwarz type domain decomposition methods to nonlinear optimization problem (1.1).

5.5.1 Decomposition, restriction, multiplicity, and several useful matrices

We assume that $\mathbb{R}^n$ is decomposed into $\{\mathcal{X}^i\}_{i=1}^m$ with

$$\mathcal{X}^i \;=\; \mathrm{span}\,\big\{e_j : j \in X^i\big\} \qquad (i = 1, 2, \ldots, m),$$

where $\{X^i\}_{i=1}^m$ is a given *partition* of $\{1, 2, \ldots, n\}$, meaning that

$$\begin{cases} \emptyset \neq X^i \subset \{1, 2, \ldots, n\} \quad \text{for each} \quad i \in \{1, 2, \ldots, m\}, \\ \bigcup_{i=1}^m X^i \;=\; \{1, 2, \ldots, n\}. \end{cases}$$

As in algebraic domain decomposition methods [41, Section 1.3], our discussion here will be entirely algebraic and assume no geometrical/physical background for $\{X^i\}_{i=1}^m$ (as well as its restriction that will be introduced later). However, if such background information is available, we should take it into consideration when designing $\{X^i\}_{i=1}^m$. For instance, if the coupling pattern of the variables can be described by a graph, we can configure $\{X^i\}_{i=1}^m$ as in [14, Section 2].

Definition 5.1. If the partition $\{X^i\}_{i=1}^m$ satisfies $X^i \cap X^{i'} = \emptyset$ for all $i, i' \in \{1, 2, \ldots, m\}$ with $i \neq i'$, then we say that $\{X^i\}_{i=1}^m$ is *non-overlapping*; otherwise, it is said to be *overlapping*.

Definition 5.2. Let $\{\tilde{X}^i\}_{i=1}^m$ be a partition of $\{1, 2, \ldots, n\}$. We call $\{\tilde{X}^i\}_{i=1}^m$ a *restriction* of $\{X^i\}_{i=1}^m$ if $\{\tilde{X}^i\}_{i=1}^m$ is non-overlapping and $\tilde{X}^i \subset X^i$ for each $i \in \{1, 2, \ldots, m\}$.

Given an integer $j \in \{1, 2, \ldots, n\}$ and a set $X \subset \{1, 2, \ldots, n\}$, we henceforth use $\mathbb{1}\big(j \in X\big)$ to denote the following binary value that indicates whether $j \in X$ is true or false:

$$\mathbb{1}\big(j \in X\big) \;=\; \begin{cases} 1 & \text{if } j \in X, \\ 0 & \text{else.} \end{cases}$$

Definition 5.3. Let $\{X^i\}_{i=1}^m$ be a partition of $\{1, 2, \ldots, n\}$. For each $j \in \{1, 2, \ldots, n\}$,

$$\theta_j \;=\; \sum_{i=1}^{m} \mathbb{1}\big(j \in X^i\big)$$

is called the *multiplicity* of $\{X^i\}_{i=1}^m$ at j. Moreover, we call

$$\theta \;=\; \max_{1 \le j \le n} \theta_j$$

the *multiplicity*[26] of the partition $\{X^i\}_{i=1}^m$.

The number θ_j is an integer between 1 and m, and it counts how many times j appears in the partition $\{X^i\}_{i=1}^m$. Clearly, $1 \le \theta \le m$, and $\theta = 1$ if and only if $\{X^i\}_{i=1}^m$ is non-overlapping, while $\theta = m$ if and only if there exists a $j \in \{1, 2, \ldots, n\}$ belonging to all X^i for $i \in \{1, 2, \ldots, m\}$. The multiplicity θ is critical in the analysis of Algorithm 5.2, especially when $\{R^i\}_{i=1}^m$ and $\{T^i\}_{i=1}^m$ are configured according to RAS (5.36) or WRAS (5.37) (see Theorem 5.1).

Example 5.1. *Suppose that* $n = 4$. *Let* $X^1 = \{1, 2, 3\}$, $X^2 = \{2, 3, 4\}$, $\tilde{X}^1 = \{1, 2\}$, *and* $\tilde{X}^2 = \{3, 4\}$. *Then* $\{X^i\}_{i=1}^2$ *is an overlapping particular of* $\{1, 2, 3, 4\}$ *with multiplicity* 2, *while* $\{\tilde{X}^i\}_{i=1}^2$ *is a non-overlapping one. Moreover,* $\{\tilde{X}^i\}_{i=1}^2$ *is a restriction of* $\{X^i\}_{i=1}^2$. *They are the partitions used in Subsection 5.3.*

Given a partition $\{X^i\}_{i=1}^m$ of $\{1, 2, \ldots, n\}$ and its restriction $\{\tilde{X}^i\}_{i=1}^m$, we now introduce several matrices that will be the building blocks for the decomposition/synchronization matrices.

For each $i \in \{1, 2, \ldots, m\}$, we define U^i to be the matrix whose columns constitute an enumeration of $\{e_j : j \in X^i\}$ with j in the ascending order. We formulate this definition as

$$U^i \;=\; \big[\, e_j \,\big]_{j \in X^i}. \tag{5.31}$$

With the same notation, we define

$$\tilde{U}^i \;=\; \big[\, \mathbb{1}\big(j \in \tilde{X}^i\big)\, e_j \,\big]_{j \in X^i}. \tag{5.32}$$

In other words, $\tilde{U}^i$ enumerates $\{\mathbb{1}\big(j \in \tilde{X}^i\big)\, e_j : j \in X^i\}$ with j in the ascending order. In addition,

$$W^i \;=\; \big[\, w_j^i e_j \,\big]_{j \in X^i}, \tag{5.33}$$

[26] [41, Definition 5.4] calls it "multiplicity of intersections".

where $\{w_j^i : 1 \le i \le m,\ 1 \le j \le n\} \subset [0,1]$ is a set of numbers satisfying[27]

$$\sum_{i=1}^{m} w_j^i \, \mathbb{1}\big(j \in X^i\big) \;=\; 1 \quad \text{for each} \quad j \in \{1, 2, \ldots, n\}. \tag{5.34}$$

Alternatively, one can define W^i as $U^i D^i$, where D^i is the $|X^i| \times |X^i|$ diagonal matrix whose diagonal enumerates $\{w_j^i\}_{j \in X^i}$ with j in the ascending order. [41, Definition 1.11] calls $\{D^i\}_{i=1}^m$ a set of *partition of unity matrices* for $\{X^i\}_{i=1}^m$.

Matrices $\{\tilde{U}^i\}_{i=1}^m$ can be regarded as a particular instance of $\{W^i\}_{i=1}^m$ with $w_j^i = \mathbb{1}\big(j \in \tilde{X}^i\big)$. Another commonly used instance sets $w_j^i = \theta_j^{-1}$ regardless of i, which indeed leads to

$$W^i \;=\; \left[\sum_{\ell=1}^{m} U^\ell [U^\ell]^\mathsf{T}\right]^{-1} U^i,$$

because $\sum_{\ell=1}^m U^\ell [U^\ell]^\mathsf{T}$ is an $n \times n$ diagonal matrix whose (j,j) entry is θ_j (see Lemma E.1).

For $\{X^i\}_{i=1}^m$ and $\{\tilde{X}^i\}_{i=1}^m$ in Example 5.1, the corresponding $\{U^i\}_{i=1}^m$, $\{\tilde{U}^i\}_{i=1}^m$, and $\{W^i\}_{i=1}^m$ are given in (5.18) and (5.21).

5.5.2 Additive Schwarz type decomposition/synchronization matrices

Let $\{X^i\}_{i=1}^m$ be the partition of $\{1, 2, \ldots, n\}$ discussed before and $\{\tilde{X}^i\}_{i=1}^m$ be a restriction of $\{X^i\}_{i=1}^m$. With the help of the matrices $\{U^i\}_{i=1}^m$, $\{\tilde{U}^i\}_{i=1}^m$, and $\{W^i\}_{i=1}^m$, following the Additive Schwarz type decomposition/synchronization strategies [14, 46, 50, 51] and the illustrations in Subsection 5.3, we can define the decomposition/synchronization matrices $\{R^i\}_{i=1}^m$ and $\{T^i\}_{i=1}^m$ for Algorithm 5.1 as follows.

$$\text{AS:} \quad R^i = [U^i]^\mathsf{T}, \quad T^i = U^i, \tag{5.35}$$

$$\text{RAS:} \quad R^i = [U^i]^\mathsf{T}, \quad T^i = \tilde{U}^i, \tag{5.36}$$

$$\text{WRAS:} \quad R^i = [U^i]^\mathsf{T}, \quad T^i = W^i, \tag{5.37}$$

$$\text{ASH:} \quad R^i = [\tilde{U}^i]^\mathsf{T}, \quad T^i = U^i, \tag{5.38}$$

$$\text{WASH:} \quad R^i = [W^i]^\mathsf{T}, \quad T^i = U^i, \tag{5.39}$$

$$\text{RASH:} \quad R^i = [\tilde{U}^i]^\mathsf{T}, \quad T^i = \tilde{U}^i. \tag{5.40}$$

Note that RAS is a special case of WRAS, and ASH is a special case of WASH.

We stress that (W)RAS and (W)ASH *do not necessarily maintain* $T^i = [R^i]^\mathsf{T}$ when $\{X^i\}_{i=1}^m$ is overlapping, while AS and RASH insist on such an equality. The first two are much more favorable in domain decomposition, as we mentioned in Subsection 5.2. In Subsection 5.8.2, we will demonstrate numerically that the same preference still holds when these decomposition/synchronization strategies are applied to nonlinear optimization problems without any PDE background. Comparing AS and RAS as illustrations, our experiment will show that the latter is clearly superior to the former.

[27] The subscript j of w_j^i is inherited from e_j, the jth canonical coordinate vector in $\mathbb{R}^n$. It is not an iteration counter, which violates our general convention about subscripts and superscripts but introduces no confusion.

5.6 Globalizing the space decomposition framework by trust regions

5.6.1 A trust-region version of the space decomposition framework

Algorithm 2.1 is not implementable before further elaborations on the **Minimization** and **Update** phases, which will play critical roles in the global convergence of the resultant algorithm. Since Algorithm 2.1 embraces Algorithm 5.1 as a special case, we can implement the latter according to the trust-region version of the former. This leads us to Algorithm 5.2, where the **Minimization** phase calculates d_k^i by approximately solving a trust-region subproblem $\min_{\|d\|\le\Delta_k^i} h_k^i(d)$, and the **Update** phase proceeds in a typical trust-region fashion according to the reduction ratio ρ_k defined in (5.42).

Algorithm 5.2 Optimization by Space Decomposition (trust-region version)

Input: $x_0 \in \mathbb{R}^n$, $\Delta_0 > 0$; $\eta_0 \ge 0$, $\eta_1 > 0$ with $\eta_0 \le \eta_1$; γ_0, γ_1, γ_2 with $0 < \gamma_0 \le \gamma_1 < 1 \le \gamma_2$; $m \in \mathbb{N}_+$, $\{N^i\}_{i=1}^m \subset \mathbb{N}_+$; $R^i \in \mathbb{R}^{N^i \times n}$, $T^i \in \mathbb{R}^{n \times N^i}$ $(i = 1, 2, \ldots, m)$.

For $k = 0, 1, 2, \ldots$, iterate the following steps.

1. **Decomposition.** Identical to Algorithm 5.1.
2. **Minimization.** For each $i \in \{1, 2, \ldots, m\}$, set

$$\Delta_k^i \;=\; \frac{\left\|\nabla h_k^i(0)\right\|}{\sqrt{\sum_{i=1}^m \left\|\nabla h_k^i(0)\right\|^2}} \cdot \Delta_k \tag{5.41}$$

   and calculate

$$d_k^i \;\approx\; \operatorname{argmin}\left\{h_k^i(d) : d \in \mathbb{R}^{N^i},\, \|d\| \le \Delta_k^i\right\}.$$

3. **Synchronization.** Identical to Algorithm 5.1.
4. **Update.** Update x_k and Δ_k according to (3.2) and (3.3) with

$$\rho_k \;=\; \frac{f(x_k) - f(x_k + s_k)}{\sum_{i=1}^m \left[h_k^i(0) - h_k^i(d_k^i)\right]}. \tag{5.42}$$

Special attention should be paid to the configuration (5.41) of the subspace trust-region radii $\{\Delta_k^i\}_{i=1}^m$. Recall that Algorithm 5.2 implicitly decomposes $\mathbb{R}^n$ into subspaces $\{\mathcal{X}^i\}_{i=1}^m$, and h_k^i is essentially a model for $f(x_k + d)$ in $\mathcal{X}^i$. The size of $\nabla h_k^i(0)$ in turn measures the first-order optimality of f at x_k along $\mathcal{X}^i$, and the algorithm determines $\{\Delta_k^i\}_{i=1}^m$ according to such optimality measurements. A subspace with lower optimality (*i.e.*, a bigger value of $\|\nabla h_k^i(0)\|$) receives a longer trust-region radius, because it is both necessary and possible to make larger progress in this subspace.

Inheriting the relation between Algorithms 2.1 and 5.1, Algorithm 5.2 can be regarded as a special instance of Algorithm 3.1 with N_k, h_k, R_k, T_k, and d_k specified in Subsection 5.4. Indeed, according to (5.29) and (5.41),

$$\|d_k\| \;=\; \Big(\sum_{i=1}^m \|d_k^i\|^2\Big)^{\frac{1}{2}} \;\le\; \Big(\sum_{i=1}^m [\Delta_k^i]^2\Big)^{\frac{1}{2}} \;=\; \Delta_k.$$

Therefore, Δ_k essentially works as a trust-region radius in $\mathbb{R}^{N_k}$, and d_k defined by (5.29) can be interpreted as an approximate solution to the trust-region subproblem $\min_{\|d\|\leq\Delta_k} h_k(d)$ with h_k given by (5.26). Moreover, with such definitions of h_k and d_k, the reduction ratio ρ_k defined in (3.1) turns out to be exactly the one specified in (5.42).

In light of the aforesaid relationship between Algorithm 3.1 and 5.2, we can cast the theory of the former to establish the global convergence and worst-case complexity bounds for the latter, as we will do in Subsection 5.6.3, where we will present the global convergence and worst-case complexities of Algorithm 5.2 with Additive Schwarz type decomposition/synchronization matrices $\{R^i\}_{i=1}^m$ and $\{T^i\}_{i=1}^m$ introduced in Subsection 5.5.

Note that one can modify Algorithm 5.2 to manage the trust-region radius Δ_k by (3.65) or (3.68)–(3.69) instead of (3.3), where $\nabla h_k(0)$ is given in (5.27) for the usage of (3.68). According to Subsection 3.6, such modifications will not lead to essential differences in our theory of Algorithm 5.2: replacing (3.3) with (3.65) only slightly affects the multiplicative constants in the worst-case complexities bounds, while switching the rule to (3.68)–(3.69) does not affect anything. We also mention that, if Algorithm 5.2 manages Δ_k by (3.68)–(3.69) instead of (3.3), then (5.41) will reduce to

$$\Delta_k^i \;=\; \Upsilon_k \|\nabla h_k^i(0)\|.$$

We point that [72] also proposed an Additive Schwarz type trust-region method (APTS). However, the decomposition/synchronization matrices in APTS (restriction/interpolation operators following the terminology in [72]) are transposes of each other, as one can see from Section 3.1.4, equation (3.2.1), and equation (4.1.2) of [72].

5.6.2 Fulfilling Assumptions 3.2–3.7

In order to cast the theory of Algorithm 3.1 to Algorithm 5.2 with Additive Schwarz type decomposition/synchronization matrices, this subsection will discuss how the assumptions postulated in Subsection 3.2 can be satisfied in this case. All the propositions in this subsection will be proved in Appendix D.

We first postulate Assumption 5.1 on the subspace models $\{h_k^i\}$, which can ensure Assumption 3.2 for h_k defined in (5.26), as is detailed in Propositions 5.1.

Assumption 5.1 (Smoothness of subspace models)**.** *For each $k \geq 0$ and $i \in \{1, 2, \ldots, m\}$, the model h_k^i is continuously differentiable, and ∇h_k^i is Lipschitz continuous in $\mathbb{R}^{N^i}$ with a Lipschitz constant $L_h > 0$ independent of k and i.*

Proposition 5.1. *Under Assumption 5.1, for each $k \geq 0$, the model h_k defined by (5.26) is continuously differentiable and ∇h_k is L_h-Lipschitz continuous in $\mathbb{R}^{N_k}$ with N_k defined in (5.24).*

As for the subspace trust-region steps $\{d_k^i\}$, we impose Assumption 5.2. Then Assumptions 3.3 and 3.4 are guaranteed for d_k defined in (5.29), as we will elaborate in Proposition 5.2.

Assumption 5.2 (Sufficient decreases of subspace trust-region steps)**.** *For each $k \geq 0$ and each $i \in \{1, 2, \ldots, m\}$, the subspace step d_k^i satisfies*

$$h_k^i(d_k^i) \;\leq\; h_k^i(\bar{d}_k^i), \tag{5.43}$$

where

$$\bar{d}_k^i \;=\; -\min\left\{\frac{\Delta_k^i}{\|\nabla h_k^i(0)\|},\;\frac{1}{L_h}\right\}\nabla h_k^i(0). \tag{5.44}$$

Here we define $0/0 = 0$ *in case* $\|\nabla h_k^i(0)\| = 0$ (*which necessitates* $\Delta_k^i = 0$ *according to* (5.41)).

Similar to the step $\bar{d}_k$ defined in (3.7), it holds for $\bar{d}_k^i$ that

$$\bar{d}_k^i \;=\; \operatorname{argmin}\Big\{h_k^i(0) + d^{\mathsf{T}}\nabla h_k^i(0) + \frac{L_h}{2}\|d\|^2 : d \in \mathbb{R}^{N^i},\ \|d\| \le \Delta_k^i\Big\},$$

whose objective function majorizes $h_k^i(d)$. Akin to the case of $\bar{d}_k$, the step $\bar{d}_k^i$ is less demanding than the classical Cauchy step, namely a minimizer of $h_k^i(d)$ on the half line $\{-t\nabla h_k^i(0) : t \ge 0\}$ subject to $\|d\| \le \Delta_k$, and Assumption 5.2 holds as long as the subspace steps achieve the Cauchy decreases. See the comments made on $\bar{d}_k$ and d_k^C on page 15.

Proposition 5.2. *Under Assumptions 5.1 and 5.2, for the model* h_k *defined in* (5.26) *and the step* d_k *defined in* (5.29), *it holds that*

$$h_k(0) - h_k(d_k) \;\ge\; \frac{1}{2}\|\nabla h_k(0)\| \min\left\{\Delta_k,\ \frac{\|\nabla h_k(0)\|}{L_h}\right\} \tag{5.45}$$

and

$$\sin\phi_k \;\le\; \sqrt{\frac{2L_h\Delta_k}{\|\nabla h_k(0)\|}}, \tag{5.46}$$

where ϕ_k *is the angle between* d_k *and* $-\nabla h_k(0)$ *defined in* (3.11).

Proposition 5.2 tells us that Assumptions 5.1–5.2 validate Assumption 3.3 with $\alpha = 1$ and Assumption 3.4 with $\beta = 2L_h$ and $p = 1/2$. Note that $\alpha = 1$ is achieved because Assumption 5.2 demands (for simplicity) (5.43) for each subspace trust-region step.

Now let us examine Assumptions 3.5–3.7 for the transformations R_k and T_k defined by (5.28) and (5.30) using the Additive Schwarz type decomposition/synchronization matrices $\{R^i\}_{i=1}^m$ and $\{T^i\}_{i=1}^m$ given in (5.35)–(5.40). To this end, we define

$$\boldsymbol{\tau}(\{T_k\}) = \sup_{k\ge 0}\|T_k\|, \quad \boldsymbol{\kappa}(\{R_k\}) = \inf_{k\ge 0}\inf_{g\ne 0}\frac{\|R_k g\|}{\|g\|}, \quad \boldsymbol{\lambda}(\{R_k, T_k\}) = \inf_{k\ge 0}\inf_{g\ne 0}\frac{g^{\mathsf{T}}T_k R_k g}{\|R_k g\|^2}. \tag{5.47}$$

Proposition 5.3 presents some bounds for these quantities, justifying Assumptions 3.5–3.7.

Proposition 5.3. *Let* θ *be the multiplicity of* $\{X^i\}_{i=1}^m$ *defined in Definition 5.3. Suppose that* $\{R^i\}_{i=1}^m$ *and* $\{T^i\}_{i=1}^m$ *are defined by one of AS* (5.35), *WRAS* (5.37), *WASH* (5.39), *and RASH* (5.40). *Then for the transformations* R_k *and* T_k *formed according to* (5.28) *and* (5.30), *we have the following conclusions on the values of* $\boldsymbol{\tau}$, $\boldsymbol{\kappa}$, *and* $\boldsymbol{\lambda}$ *defined in* (5.47).

1. *AS:* $\boldsymbol{\tau} = \sqrt{\theta}$, $\boldsymbol{\kappa} \ge 1$, $\boldsymbol{\lambda} = 1$.
2. *WRAS:* $\boldsymbol{\tau} \le 1$, $\boldsymbol{\kappa} \ge 1$, $\boldsymbol{\lambda} \ge \theta^{-1}$.
3. *WASH:* $\boldsymbol{\tau} = \sqrt{\theta}$, $\boldsymbol{\kappa} \ge \sqrt{\theta^{-1}}$, $\boldsymbol{\lambda} \ge 1$.
4. *RASH:* $\boldsymbol{\tau} = 1$, $\boldsymbol{\kappa} = 1$, $\boldsymbol{\lambda} = 1$.

We recall that RAS (5.36) is a special case of WRAS (5.37), and ASH (5.38) is a special case of WASH (5.39). Thus Proposition 5.3 covers RAS and ASH as well.

5.6.3 Global convergence and worst-case complexity bounds

According to the theory of Algorithm 3.1 in Section 3, with the help of Propositions 5.1–5.3, we can establish immediately the global convergence and worst-case complexity bounds for Algorithm 5.2 when it uses the Additive Schwarz type decomposition/synchronization matrices specified in Subsection 5.5.

Theorem 5.1. *Consider Algorithm 5.2. Suppose that* $\{R^i\}_{i=1}^m$ *and* $\{T^i\}_{i=1}^m$ *are defined according to a partition* $\{X^i\}_{i=1}^m$ *of* $\{1, 2, \ldots, n\}$*. Let one of the following situations be true.*

1. $\{R^i\}_{i=1}^m$ *and* $\{T^i\}_{i=1}^m$ *are defined by AS (5.35), ASH (5.38), WASH (5.39), or RASH (5.40), and the algorithmic parameter* η_1 *is less than* 1*.*
2. $\{R^i\}_{i=1}^m$ *and* $\{T^i\}_{i=1}^m$ *are defined by RAS (5.36) or WRAS (5.37), and the algorithmic parameter* η_1 *is less than* θ^{-1}*, where* θ *is the multiplicity of* $\{X^i\}_{i=1}^m$ *defined in Definition 5.3.*

Then under Assumptions 3.1, 5.1, and 5.2, Algorithm 5.2 enjoys the global convergence elaborated in Theorems 3.1 and 3.7, and it possesses the worst-case complexity bounds quantified in Theorems 3.2–3.4.

Theorem 5.1 tells us that Algorithm 5.2 equipped with Additive Schwarz type decomposition/synchronization matrices enjoys the same convergence as Algorithm 3.1 provided that the parameter η_1 is *compatible with the decomposition/synchronization strategy*. In the cases of AS, (W)ASH, and RASH, the constraint on η_1 is $0 < \eta_1 < 1$, which is a common requirement in trust-region methods. Things become more interesting when Algorithm 5.2 adopts (W)RAS, in which case the multiplicity θ of $\{X^i\}_{i=1}^m$ comes into play and $0 < \eta_1 < \theta^{-1}$ turns out sufficient.

5.7 Coarse space correction

Algorithm 5.1 will likely slow down when the number of subspaces increases. This is because the **Minimization** phase minimizes f along the subspaces separately, losing the coupling information between them. This phenomenon is well studied in domain decomposition, and coarse space correction [148, 92] is the remedy. The idea is to enrich the decomposition $\{\mathcal{X}^i\}_{i=1}^m$ by a so-called coarse space $\mathcal{X}_k^0$ that *overlaps* with each $\mathcal{X}^i$, representing global information that is otherwise scattered among the subspaces. We expect that minimizing f in $x_k + \mathcal{X}_k^0$ will transmits information across the subspaces and thus recovering some information lost when the coupling between $\{\mathcal{X}^i\}_{i=1}^m$ is ignored.

Both theoretically and numerically, it has been proved that coarse spaces can substantially enhance domain decomposition algorithms for PDEs. Indeed, such a coarse component can render convergence bounds that are independent of the number of subdomains [11, 148] and hence guarantee scalability in parallel computing. The idea of coarse space correction has also achieved remarkable success in PDE optimization [8, 9]. Thus we are motivated to integrate this technique into our space decomposition framework so that such a technique can be exploited in general nonlinear optimization as well.

5.7.1 Incorporating coarse spaces into the space decomposition framework

Algorithm 5.3 formulates a framework that incorporates coarse space correction into optimization by space decomposition. In the algorithm, the coarse space $\mathcal{X}_k^0$ is not shown explicitly but represented by $\mathbb{R}^{M_k}$. Matrices C_k and C_k^{T} work as the restriction/prolongation operators for

the coarse space. Note that they are the transposes of each other, which is different from the configuration of the decomposition/synchronization matrices.

Algorithm 5.3 Optimization by Space Decomposition with Coarse Space (prototype)

Input: Identical to Algorithm 5.1.

For $k = 0, 1, 2, \ldots,$ iterate the following steps.

1. **Decomposition.** Define functions $\{h_k^i\}_{i=1}^m$ as Algorithm 5.1 does. In addition, define a function $H_k : \mathbb{R}^{M_k} \to \mathbb{R}$ satisfying
$$\nabla H_k(0) \;=\; C_k \nabla f(x_k)$$
with a certain positive integer M_k and a certain matrix $C_k \in \mathbb{R}^{n \times M_k}$.
2. **Minimization.** Calculate steps $\{d_k^i\}_{i=1}^m$ as Algorithm 5.1 does. In addition, calculate
$$\delta_k \;\approx\; \operatorname{argmin} \left\{ H_k(\delta) : \delta \in \mathbb{R}^{M_k} \right\}.$$
3. **Synchronization.** Define
$$s_k \;=\; \omega_k \sum_{i=1}^m T^i d_k^i + (1 - \omega_k) C_k^{\mathsf{T}} \delta_k \tag{5.48}$$
with some matrices $T^i \in \mathbb{R}^{n \times N^i}$ $(i = 1, 2, \ldots, m)$ and a certain weight $\omega_k \in (0, 1]$.
4. **Update.** Set x_{k+1} to either $x_k + s_k$ or x_k *depending on the quality of* s_k.

Reusing the arguments in Subsection 5.4, we can show that Algorithm 5.3 is a special instance of Algorithm 4.1 with N_k, h_k, R_k, d_k, and T_k specified in Subsection 5.4. Hence we can globalize Algorithm 5.3 by trust regions according to Algorithm 4.2 (the trust-region version of Algorithm 4.1), obtaining Algorithm 5.4.

In Algorithm 5.4, the trust-region subproblem for the coarse space model H_k takes a full trust-region radius as shown in (5.49), while the subspace models $\{h_k^i\}_{i=1}^m$ share a single trust-region radius according to (5.41). Such a difference is due to the distinct roles played by H_k and $\{h_k^i\}_{i=1}^m$: H_k is supposed to reflect (coarsely) the behavior of $f(x_k + d)$ for d in $\mathbb{R}^n$, while each h_k^i depicts $f(x_k + d)$ only for d in a subspace of $\mathbb{R}^n$. This distinction is also manifested in the way that s_k integrates the coarse space step δ_k and the subspace steps $\{d_k^i\}_{i=1}^m$: in (5.48), δ_k acts as a counterpart to the ensemble of $\{d_k^i\}_{i=1}^m$ but not any individual of them.

The global convergence and worst-case complexity theory of Algorithm 5.4 can be established by specializing that of Algorithm 4.2. In particular, we have Theorem 5.2, which follows directly from Theorem 4.1 with the help of Propositions 5.1–5.3.

Theorem 5.2. *Consider Algorithm 5.4. Suppose that the matrices* $\{R^i\}_{i=1}^m$, $\{T^i\}_{i=1}^m$ *and the algorithmic parameter* η_1 *are configured as in Theorem 5.1. Then under Assumptions 3.1, 4.1, 5.1, and 5.2, Algorithm 5.4 enjoys the global convergence elaborated in Theorems 3.1 and 3.7, and it possesses the worst-case complexity bounds quantified in Theorems 3.2–3.4.*

The convergence rate (worst-case complexity) in Theorem 5.2 is standard. We have yet to investigate how the coarse space correction can expedite the convergence. Such an effect will be clearly demonstrated by the numerical experiments in Subsections 5.8.3–5.8.4, although the theoretical analysis will be left for future study.

Algorithm 5.4 Optimization by Space Decomposition with Coarse Space (trust-region version)

Input: Identical to Algorithm 5.2.

For $k = 0, 1, 2, \ldots$, iterate the following steps.

1. **Decomposition.** Identical to Algorithm 5.3.
2. **Minimization.** Calculate steps $\{d_k^i\}_{i=1}^m$ as Algorithm 5.2 does. In addition, calculate
$$\delta_k \approx \operatorname{argmin}\left\{H_k(\delta) : \delta \in \mathbb{R}^{M_k}, \|\delta\| \le \Delta_k\right\}. \tag{5.49}$$
3. **Synchronization.** Identical to Algorithm 5.3.
4. **Update.** Update x_k and Δ_k according to (3.2) and (3.3) with
$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{\omega_k \sum_{i=1}^m \left[h_k^i(0) - h_k^i(d_k^i)\right] + (1-\omega_k)[H_k(0) - H_k(\delta_k)]}. \tag{5.50}$$

5.7.2 Devising coarse spaces

Designing an effective coarse space is nontrivial. It is still under active investigation even in domain decomposition for PDEs [148, 92, 41, 55] after years of exploration. Good coarse spaces rely on the geometrical/physical background or structure of the problem. For example, if the solution is known to be sparse, then it is natural to define a coarse space to be the space spanned by the canonical basis vectors corresponding to the best available estimated support set of the solution. For generic problems, there are still some problem-independent algebraic coarse spaces to try. Here we suggest some possibilities.

The first possibility is to choose a linear space spanned by subspace steps of *previous* iterations. For instance, we can set
$$\mathcal{X}_k^0 = \operatorname{span}\left\{T^i d_{l_k}^i\right\}_{i=1}^m \tag{5.51}$$
as the coarse space for iteration k, where l_k can be either $k-1$ (*i.e.*, the last iteration) or $\max\{l \in \mathbb{N} : l \le k-1 \text{ and } \rho_l > \eta_0\}$ (*i.e.*, the last *successful* iteration).[28] Since $d_{l_k}^i$ inhabits in $\mathbb{R}^{N^i}$, coarse space (5.51) relies on $T^i \in \mathbb{R}^{n \times N^i}$ to map $d_{l_k}^i$ to $\mathbb{R}^n$. This coarse space resembles but still differs from the basic idea of [49, 93, 131, 52, 42], which explore the space spanned by the subspace steps of the *current* iteration, yet they do not use such a space as a coarse space. More importantly, (5.51) defines $\mathcal{X}_k^0$ by the subspace steps of a *previous* iteration, which makes the **Minimization** phase of Algorithms 5.3–5.4 parallelizable. One can also choose $\mathcal{X}_k^0 = \operatorname{span}\left\{[R^i]^\mathsf{T} d_{l_k}^i\right\}_{i=1}^m$. It is similar to (5.51), but $d_{l_k}^i$ is mapped to $\mathbb{R}^n$ by $[R^i]^\mathsf{T}$ instead of T^i.

The second possibility is to define a coarse space based on previous full-space steps. Exploring steps of past iterations is a common way of accelerating iterative schemes, momentum-type accelerated algorithms [114, 103] being prominent examples. Here we can simply take
$$\mathcal{X}_k^0 = \operatorname{span}\{s_{l_k}\} \tag{5.52}$$
as a coarse space, where l_k again signifies $k-1$ or $\max\{l \in \mathbb{N} : l \le k-1 \text{ and } \rho_l > \eta_0\}$. Despite its simplicity, this coarse space worked quite well in our tests (see Subsections 5.8.3–5.8.4). Alternatively, we can decompose s_{l_k} into components according to the space decomposition $\{\mathcal{X}^i\}_{i=1}^m$

[28] For the second strategy, when $\{l \in \mathbb{N} : l \le k-1 \text{ and } \rho_l > \eta_0\} = \emptyset$, we can set $l_k = k-1$. For both strategies, when $k = 0$, the coarse space has to be chosen ad hoc or set to $\{0\}$.

and use such components to span a coarse space. In particular, for the Additive Schwarz type methods introduced in Subsection 5.5, we can either choose $\mathcal{X}_k^0 = \text{span}\left\{U^i[U^i]^\mathsf{T} s_{l_k}\right\}_{i=1}^m$, or set

$$\mathcal{X}_k^0 \;=\; \text{span}\left\{U^i[\tilde{U}^i]^\mathsf{T} s_{l_k}\right\}_{i=1}^m \;=\; \text{span}\left\{\tilde{U}^i[\tilde{U}^i]^\mathsf{T} s_{l_k}\right\}_{i=1}^m. \tag{5.53}$$

Note that each entry of s_{l_k} appears exactly once in the vectors $\left\{U^i[\tilde{U}^i]^\mathsf{T} s_{l_k}\right\}_{i=1}^m$ due to $\{\tilde{U}^i\}_{i=1}^m$ (see (5.32)), while it can be repeated in $\left\{U^i[U^i]^\mathsf{T} s_{l_k}\right\}_{i=1}^m$ if the decomposition $\{\mathcal{X}^i\}_{i=1}^m$ is overlapping. For this reason, we suggest decomposing s_{l_k} by $\left\{U^i[\tilde{U}^i]^\mathsf{T} s_{l_k}\right\}_{i=1}^m$, and $\left\{U^i[W^i]^\mathsf{T} s_{l_k}\right\}_{i=1}^m$ is another possibility (see (5.33) for the definition of W^i).

All the coarse spaces described above can be further enriched by gradient directions if they are available. For instance, instead of (5.52), we can define

$$\mathcal{X}_k^0 \;=\; \text{span}\{s_{l_k},\, -g_k\}, \tag{5.54}$$

the motivation coming from the subspace viewpoint on conjugate gradient method [160], which suggests exploring the subspace spanned by the *current* gradient and *previous* steps.

Recall that Algorithms 5.3–5.4 implement coarse space correction by restriction/prolongation operators C_k and C_k^T. For coarse space (5.51), a possible definition for C_k is

$$C_k \;=\; \left[\frac{T^1 d_{l_k}^1}{\|T^1 d_{l_k}^1\|} \;,\; \cdots \;,\; \frac{T^m d_{l_k}^m}{\|T^m d_{l_k}^m\|}\right].$$

If (nearly) zero vectors occur in $\{T^i d_{l_k}^i\}_{i=1}^m$, we exclude them from C_k. This definition fulfills item 2 of Assumption 4.1, which is the only assumption that Theorem 5.2 imposes on the coarse space restriction/prolongation operators. The other aforementioned coarse spaces can be implemented by devising C_k in a similar way. For (5.53), the columns of such C_k constitute an orthonormal basis of $\mathcal{X}_k^0$, because $\left\{U^i[\tilde{U}^i]^\mathsf{T} s_{l_k}\right\}_{i=1}^m$ are mutually orthogonal.

Coarse spaces (5.51)–(5.54) all worked well in our experiments (see Subsections 5.8.3–5.8.4). Intriguingly, the one-dimensional space (5.52) performed remarkably on the problems we tried. However, we stress that coarse space correction is a highly problem-dependent technique. It will be no surprise if the suggestions made here turn out ineffective for certain problems.

5.8 Numerical illustrations

The space decomposition framework elaborated in Algorithms 5.1–5.4 contains important ingredients that have been proved crucial in PDEs, particularly the overlapping strategies like (W)RAS and coarse space correction. The purpose of this subsection is to demonstrate numerically the significance of such ingredients even if the optimization problem is *not PDE-based*.

We put forward a note of caution before presenting the experiments: our space decomposition framework is not proposed as a specific algorithm but a general foundation for designing and interpreting algorithms that utilize space decomposition methodology. When applied to a particular type of problems, the framework has to be adapted as much as possible to the problem structure (*e.g.*, separability, sparsity, scaling), taking into account the geometrical/physical background if such information is available, as will be illustrated in Subsection 5.8.4. This is the same as domain decomposition, which is general framework to be tailored when a certain class of problems are given (*e.g.*, [24, Sections 9–11]). The aim of our examples is to illustrate the key ingredients rather than discovering the optimal implementation of our framework, which depends on particular applications and we leave it for future research.

5.8.1 General settings of the experiments

Our experiments are based on an unsophisticated implementation of Algorithms 5.2 and 5.4 with the Additive Schwarz type decomposition/synchronization matrices introduced in Subsection 5.5. As illustrations, we focus on AS (5.35) and RAS (5.36). We elaborate here all the key points in the implementation that will lead to the results presented later.

To implement Algorithms 5.2 and 5.4 with AS (5.35) and RAS (5.36) on an n-dimensional problem, we need first to define a partition $\{X^i\}_{i=1}^m$ of $\{1, 2, \ldots, n\}$ and a restriction $\{\tilde{X}^i\}_{i=1}^m$ of $\{X^i\}_{i=1}^m$, where $m \geq 2$ is the number of subspaces. Here we simply define them by dividing $\{1, 2, \ldots, n\}$ into balanced groups.[29] Assume for simplicity that m divides n. We begin with

$$\tilde{X}^i \;=\; \Big\{\frac{n}{m}(i-1)+1,\; \ldots,\; \frac{n}{m}i\Big\}, \quad i = 1, 2, \ldots, m, \tag{5.55}$$

which splits $\{1, 2, \ldots, n\}$ into m groups of size n/m following the natural order, and then let X^i be $\big\{(\min \tilde{X}^i) - n_{\mathrm{o}}/2,\; \ldots,\; (\max \tilde{X}^i) + n_{\mathrm{o}}/2\big\}$, introducing overlaps of size n_{o} between consecutive groups. Here, again for simplicity, we assume n_{o} to be an even number, and require $0 \leq n_{\mathrm{o}} \leq n/m$ so that overlaps are limited to neighboring groups. Exceptions should be made for X^i when $i = 1$ and $i = m$, where we take $X^1 = \big\{1,\; \ldots,\; (\max \tilde{X}^1) + n_{\mathrm{o}}/2\big\}$ and $X^m = \big\{(\min \tilde{X}^m) - n_{\mathrm{o}}/2,\; \ldots,\; n\big\}$ to keep X^i within $\{1, 2, \ldots, n\}$. Overall, we set

$$X^i \;=\; \Big\{\frac{n}{m}(i-1)+1-\frac{n_{\mathrm{o}}}{2},\; \ldots,\; \frac{n}{m}i+\frac{n_{\mathrm{o}}}{2}\Big\} \bigcap \{1, 2,\; \ldots,\; n\}, \quad i = 1, 2, \ldots, m. \tag{5.56}$$

According to Definitions 5.1–5.3, $\{X^i\}_{i=1}^m$ is a partition of $\{1, 2, \ldots, n\}$, $\{\tilde{X}^i\}_{i=1}^m$ is a restriction of $\{X^i\}_{i=1}^m$, and the multiplicity of $\{X^i\}_{i=1}^m$ is

$$\theta \;=\; \begin{cases} 1 & \text{if } n_{\mathrm{o}} = 0, \\ 2 & \text{if } 1 \leq n_{\mathrm{o}} \leq n/m. \end{cases}$$

With $\{X^i\}_{i=1}^m$ and $\{\tilde{X}^i\}_{i=1}^m$, the decomposition/synchronization matrices $\{R^i\}_{i=1}^m$ and $\{T^i\}_{i=1}^m$ can be constructed according to AS (5.35) or RAS (5.36).

In the **Decomposition** phase of Algorithm 5.2, according to (5.23), we simply set

$$h_k^i(d) = f(x_k) + d^{\mathsf{T}} R^i \nabla f(x_k) + \frac{1}{2} d^{\mathsf{T}} R^i \nabla^2 f(x_k) [R^i]^{\mathsf{T}} d, \quad d \in \mathbb{R}^{N^i} \quad (N^i = |X^i|).$$

The **Minimization** phase of Algorithm 5.2 solves its subproblem by the Toint-Steihaug truncated conjugate gradient method described in [30, Algorithm 7.5.1], which achieves the Cauchy decrease and fulfills Assumption 5.2. For trail step acceptance, the **Update** phase of Algorithm 5.2 chooses $\eta_0 = 0$. We use (3.64) to update trust-region radius with $\gamma_1 = 1/2$, $\gamma_2 = 2$, $\eta_1 = \lambda/10$, and $\eta_2 = 3\lambda/4$, where λ is the upper bound imposed on η_1 by Theorem 5.1, namely

$$\lambda \;=\; \begin{cases} 1 & \text{for AS}, \\ \theta^{-1} & \text{for RAS}. \end{cases}$$

[29] Similar to domain decomposition, our space decomposition framework is intended to be implemented in parallel when applied to large-scale problems. In such a setting, it is essential to limit load imbalance between tasks executed in parallel. Thus it is rational to balance the number of variables in each subspace, assuming that the cost of each subproblem in the **Minimization** phase of Algorithm 5.2 is roughly proportional to the dimension of the subspace. Such an assumption holds for the problems tested here. In practice, problem-dependent and possibly iteration-dependent partitions should be devised according to the actual cost of the subproblems.

In other words, $\lambda = 1/2$ for RAS with $n_o \geq 1$ and $\lambda = 1$ otherwise. Note that we should calculate $\|d_k\|$ by $\sqrt{\sum_{i=1}^m \|d_k^i\|^2}$ in (3.64), because $d_k = \left[d_k^1; d_k^2; \cdots; d_k^m\right]$ when we interpret Algorithm 5.2 as a special case of Algorithm 3.1, as we can recall from (5.29). Δ_0 is set to 1.

Algorithm 5.4 is implemented by integrating coarse spaces into the implementation of Algorithm 5.2. It suffices to specify C_k, H_k, δ_k, and ω_k. Matrix C_k is defined according to one of the coarse spaces (5.51)–(5.54) recommended in Subsection 5.7.2. Then following (4.2), we choose

$$H_k(\delta) \;=\; f(x_k) + \delta^{\mathsf{T}} C_k \nabla f(x_k) + \frac{1}{2}\delta^{\mathsf{T}}\left[C_k \nabla^2 f(x_k) C_k^{\mathsf{T}}\right]\delta, \quad \delta \in \mathbb{R}^{M_k}.$$

To obtain δ_k, we solve the trust-region subproblem in the coarse space still by the Toint-Steihaug truncated conjugate gradient method. For ω_k, we simply pick a constant value 3/4. Assumption 4.1 is fulfilled under these configurations of C_k, H_k, δ_k, and ω_k. Incorporating coarse spaces does not change the schemes for trial step acceptance and trust-region radius update, but the reduction ratio ρ_k should be calculated by (5.50) instead of (5.42).

Coarse spaces (5.51)–(5.54) need to specify l_k. Both $l_k = \max\{l \in \mathbb{N} : l \leq k-1 \text{ and } \rho_l > \eta_0\}$ and $l_k = k-1$ were tried in our (rather limited) tests. Unexpectedly, the two strategies led to little difference. Thus $l_k = k-1$ is chosen for simplicity ($k \geq 1$). At iteration 0, we define $C_0 = 0$ and $\omega_0 = 0$ for coarse spaces (5.51)–(5.53), while $C_0 = -g_0/\|g_0\|$ and $\omega_0 = 3/4$ for (5.54).

5.8.2 A comparison between AS and RAS

We first try our implementation of Algorithm 5.2 on a set of test problems, compare the behavior of AS and RAS, and manifest the superiority of RAS over AS when the partition $\{X^i\}_{i=1}^m$ is overlapping. By the performance of RAS, we will demonstrate the potential of designing highly scalable parallel optimization algorithms based on our space decomposition framework.

The test problems are taken from CUTEst [62] and Powell [119]. The first column of Table 2 contains their names, each accompanied by a bracketed letter indicating the sparsity pattern of the problem's Hessian matrix with the code below.

1. A: arrowhead matrix with all nonzeros in the diagonal and the last row/column;
2. B: band matrix;
3. D: double arrowhead matrix with nonzeros in the diagonal, the first and last rows/columns;
4. F: full matrix without particular sparsity.

These patterns reflect the coupling structures of corresponding problems. Although such structures are *not* exploited in our implementation of Algorithm 5.2, they affect the behavior of the algorithm. The formulations and standard starting points of these problems can be found in [119] (`arwhead`, `chrosen`, `penalty1`, and `vardim`) or [62] (all but `chrosen`). Problems `arwhead`, `engval1`, `powellsg`, `power`, `tridia`, and `vardim` are convex, while the others are nonconvex.

The test problems are with adjustable dimension. As illustrations, we solved the 10^3- and 10^4-dimensional versions of them by our implementation of Algorithm 5.2 with the number m of subspaces being 10 and 10^2 respectively. The overlap size n_o was set to 0, 2, 4, or 6. Both AS and RAS were tried when defining the decomposition/synchronization matrices. The two strategies are identical when $n_o = 0$. As a reference, we also executed Algorithm 5.2 with no decomposition ($m = 1$), our implementation reducing to a standard trust-region method in $\mathbb{R}^n$. Table 2 presents the number of iterations for all the variants of Algorithm 5.2 to fulfill

Overlap size	0	2		4		6		No decomposition
Strategy	AS/RAS	AS	RAS	AS	RAS	AS	RAS	
arwhead (A)	5	7	5	7	5	7	5	5
	5	5	5	5	5	5	5	5
broydn3d (B)	20	22	9	28	7	29	7	6
	22	33	10	30	9	33	8	8
broydnbd (B)	8	15	8	15	8	17	8	8
	10	15	9	16	10	18	10	9
chrosen (B)	33	190	26	197	27	217	24	22
	29	139	28	165	34	132	27	31
cosine (B)	140	7478	105	–	110	–	104	4514
	347	–	356	–	578	–	247	3378
cragglvy (B)	37	23	15	23	15	25	15	15
	39	19	16	26	15	22	14	14
engval1 (B)	11	17	10	20	10	22	10	10
	13	21	12	25	12	28	12	12
errinros (B)	68	404	72	1979	68	4113	67	90
	152	155	159	2538	136	1421	145	82
freuroth (B)	9	138	9	318	9	1551	9	8
	10	30	10	24	10	22	10	9
powellsg (B)	15	28	15	132	19	149	15	15
	16	29	16	182	22	197	16	16
tridia (B)	19	24	10	25	9	35	8	6
	23	26	12	27	10	32	9	7
eg2 (D)	8	33	8	23	9	25	14	8
	12	18	13	16	13	13	9	7
penalty1 (F)	19	19	19	19	19	19	19	23
	20	24	20	21	20	21	20	28
power (F)	8	9	9	9	8	9	9	14
	9	14	8	17	8	17	8	16
vardim (F)	10	10	10	10	10	10	10	14
	15	12	14	12	10	13	10	15

Table 2: Number of iterations for AS/RAS to solve some test problems (bracketed letter: coupling pattern; odd row: 10^3 variables, 10 subspaces; even row: 10^4 variables, 10^2 subspaces).

convergence criterion $\|g_k\| \leq 10^{-6}\|g_0\|$, a "–" being displayed if such a condition was not achieved within 10^4 iterations.

First of all, as expected, RAS (5.36) rather than AS (5.35) turned out to be the correct strategy to define the decomposition/synchronization matrices in the overlapping cases ($n_o > 0$). Therefore, RAS is an appropriate *overlapping strategy* while AS is not. The surplus correction of AS (5.35) was detrimental for all the problems except `arwhead`, `penalty1`, and `vardim`, where the two strategies performed similarly. The comparison between AS and RAS matches what is known for linear systems arising from PDEs (see Subsection 5.2 and the references therein), although the problems in Table 2 have absolutely *no PDE background*.

For `broydn3d`, `cragglvy`, `tridia`, as well as 10^4-dimensional `vardim`, introducing overlaps in the decomposition significantly reduced the number of iterations provided that RAS was the overlapping strategy. Moreover, for these problems, augmenting the overlap size expedited the convergence of RAS, resembling the behavior of parallel Schwarz domain decomposition methods for certain types of PDEs and linear systems (see, *e.g.*, [41, Section 1.5] and [51, Section 5]). It will be interesting to investigate such a phenomenon theoretically.

With RAS being the overlapping strategy, Algorithm 5.2 worked well on the test problems despite the naïveness of our implementation. In particular, the algorithm *scaled* nicely when both the dimension and number of subspaces were enlarged by a factor of 10. Its scalability is also reflected by the surprising fact that, even with 10^2 subspaces, RAS did not need much more iterations than the full-space method to converge for these problems, including the fully coupled `penalty1`, `power`, and `vardim`. For problem `cosine`, whose Hessian is tridiagonal, RAS even greatly outperformed the full-space method. Due to the considerably lower dimension of the subspaces, one can expect for a large class of problems that establishing models and solving subproblems in the subspaces are much more economical than doing the same in the full space, the latter becoming even *impossible* when the problem is huge. Moreover, the subspace computations are parallel among the subspaces by the design of our framework. Thus the nice scalability of RAS can lead to considerable speedup when it is implemented on parallel machines.

The performance of RAS on these test problems implies that RAS is naturally well adapted to the coupling structures of such problems. It is however *unrealistic* to expect RAS can work equally well on all types of problems. This urges us to characterize analytically the problems that RAS can handle, opening a broad avenue for future research.

5.8.3 Improvement by coarse space correction: an algebraic example

Our second experiment is to test Algorithm 5.4 and illustrate the significance of coarse space correction. We will show by an example that, when the objective function has relative strong coupling among the variables, Algorithm 5.4 with appropriate coarse spaces can considerably outperform Algorithm 5.2, and that coarse spaces can evidently improve the scalability of our space decomposition framework when the number of subspaces increases.

We focus on coarse spaces (5.51)–(5.54) introduced in Subsection 5.7.2, which are problem-independent and purely algebraic. For convenience, we assign the following code names to them: SS (Subspace Steps) for (5.51), FS (Full-space Step) for (5.52), DF (Decomposition of Full-space step) for (5.53), and CG (Conjugate Gradient) for (5.54). In addition, Nil indicates that no coarse space is in use, where we define $C_k = 0$ and $\omega_k = 1$ in Algorithm 5.4 so that it reduces to Algorithm 5.2. In all cases, we use RAS with overlap size $n_o = 2$ to set the decomposition/synchronization matrices.

	chrosen					chrosen RC					chrosen SRC				
	Nil	SS	FS	DF	CG	Nil	SS	FS	DF	CG	Nil	SS	FS	DF	CG
2	31	18	22	22	24	32	26	24	23	26	32	27	25	24	25
4	22	18	24	24	25	34	24	24	23	27	35	24	25	24	27
8	29	24	24	27	25	51	26	23	24	33	52	26	25	25	33
16	28	23	24	23	23	70	31	27	30	35	72	33	33	30	37
32	22	19	21	19	20	89	37	31	35	40	100	31	34	33	40
64	24	19	20	19	20	120	33	32	32	41	106	36	35	34	40
128	24	20	21	20	21	135	37	40	35	40	116	38	41	33	42
256	24	20	21	20	22	145	37	39	33	42	117	40	40	36	42

Table 3: Number of iterations for Algorithm 5.4 with RAS and $n_o = 2$ to solve `chrosen` and its variants with 6400 variables (columns: coarse spaces; rows: number of subspaces).

To test Algorithm 5.4 and the coarse spaces, we use the `chrosen` (chained Rosenbrock) problem from Powell [119] and its variants that we define later. The objective function of `chrosen` is

$$f(x) \;=\; \sum_{i=1}^{n-1} \left[4(u_i - u_{i+1}^2)^2 + (1 - u_{i+1})^2\right], \quad x = [\,u_1\,;\,u_2\,;\,\cdots\,;\,u_n\,] \in \mathbb{R}^n.$$

The standard starting point is $x_0 = [-1\,;\,-1\,;\,\cdots\,;\,-1]$.

Table 3 shows in its first part (under "`chrosen`") the numbers of iterations for our implementation of Algorithm 5.4 to achieve $\|g_k\| \leq 10^{-6}\|g_0\|$ on the 6400-dimensional[30] `chrosen` problem with various coarse spaces (corresponding to columns) and different numbers of subspaces (corresponding to rows). We see that Algorithm 5.4 scaled quite well when the number of subspaces increased more than 100 times (2 to 256). It was the case even for the version with no coarse space, the implementation being identical to Algorithm 5.2. The coarse spaces typically reduced the number of iterations, yet the effect was not significant.

According to these results, one may infer that coarse spaces are beneficial but not obligatory. However, such an inference will turn out invalid if we introduce more coupling into the objective function. To see this, we consider a perturbation of `chrosen` defined by

$$f_V(x) \;=\; f(x) + \frac{1}{2} x^{\mathsf{T}} V V^{\mathsf{T}} x, \quad x \in \mathbb{R}^n,$$

where $V \in \mathbb{R}^{n\times r}$ is a constant matrix with r columns ($r \ll n$) whose 2-norms are $\sigma > 0$. By the term $(x^{\mathsf{T}} V V^{\mathsf{T}} x)/2$, we intend to embed simple yet nontrivial coupling into f_V in addition to the `chrosen` objective function f. Note that $\operatorname{rank}(VV^{\mathsf{T}}) \leq r$, and the eigenvalues of VV^{T} are located in $[0,\, r\sigma^2]$. We propose two probabilistic schemes for generating V (once generated, V is fixed so that $f_V(x)$ is a deterministic function of x), resulting in two variants of `chrosen`.

1. `chrosen` with Random Coupling (`chrosen` RC). The columns of V are independent samples from the uniform distribution on $\sigma\,\mathbb{S}^{n-1}$, where $\mathbb{S}^{n-1}$ is the unit sphere in $\mathbb{R}^n$. Each sample can be obtained by drawing a vector from the standard normal distribution in $\mathbb{R}^n$ and scaling its 2-norm to σ. Then, almost surely, VV^{T} is a full matrix and hence couples all the variables.

[30] We choose $n = 6400$ so that, for simplicity, it is always divisible by the number of subspaces.

2. `chrosen` with Sparse Random Coupling (`chrosen` SRC). The columns of V are generated independently as follows: draw a vector of n independent random variables taking values -1, 1, and 0 with probability $p/2$, $p/2$, and $1-p$ respectively, and scale its 2-norm to σ. Here $p \in (0,1)$ is a constant. In this case, VV^{T} is sparse with roughly rp^2n^2 nonzeros.

Setting $r = 2$ and $\sigma = 5$, we generated 10 independent instances of `chrosen` RC, solved them by our implementation of Algorithm 5.4 with various combinations of coarse space and number of subspaces, and recorded for each combination the average number of iterations (rounded to the nearest integer) to achieve $\|g_k\| \leq 10^{-6}\|g_0\|$. Table 3 presents these values in its second part below "`chrosen` RC", columns corresponding to different coarse spaces while rows corresponding to different numbers of subspaces. The same experiment was conducted on `chrosen` SRC with $r = 2$, $\sigma = 5$, and $p = 10^{-1}$, results shown in the third part of Table 3.

Compared with `chrosen`, Algorithm 5.4 took more iterations to solve `chrosen` RC and `chrosen` SRC due to the additional coupling. Nevertheless, Algorithm 5.4 still scaled well on `chrosen` RC and `chrosen` SRC when the number of subspaces varied from 2 to 256. Even without coarse space, the iteration number grew less than five times on average while the number of subspaces increased more than 100 times. However, with coarse spaces, the performance was evidently better: the problems were solved with considerably smaller iteration numbers, which grew at a visibly slower rate when the number of subspaces was increasing.

To conclude, coarse spaces (5.51)–(5.54) worked well on `chrosen` and its variants. Their effectiveness was more evident when additional coupling was injected into the `chrosen` objective function. Little difference was detected among the four of them. However, recall that coarse space correction is a rather problem-dependent technique. The generic coarse spaces tested here will change their behavior when problems change, and they will be defeated by problem-specific strategies when problem structure is available to exploit, as we will see in next experiment.

5.8.4 Exploiting problem structure: a geometrical example

The final experiment will illustrate how to implement Algorithms 5.2 and 5.4 according to the particularity of a given problem. For an example with geometrical background, we design a structure-aware decomposition and a problem-dependent coarse space, and show that such an adapted implementation can greatly outperform the generic one introduced in Subsection 5.8.1.

Our illustrative example is a simple instance of Plateau's problem in $\mathbb{R}^3$, namely to find a surface of minimal area with a prescribed boundary [44]. Being applied in a wide variety of sciences, this problem admits analytical solutions only in special cases, and numerical methods are needed in general [43, 142]. As a simple illustration, we consider the following instance:

$$\begin{aligned} &\min_{\Phi} \quad \iint_{[0,\,1]^2} \Big[1 + \|\nabla\Phi(u,v)\|^2\Big]^{\frac{1}{2}} \,\mathrm{d}u\,\mathrm{d}v \\ &\text{s.t.} \quad \Phi(u,v) = \begin{cases} 0 & u \in [0,\,1],\ \ v = 0 \text{ or } 1, \\ \sin(4\pi v) + \dfrac{1}{10}\sin(120\pi v) & v \in [0,\,1],\ \ u = 0 \text{ or } 1. \end{cases} \end{aligned} \tag{5.57}$$

Discretizing $[0,\,1]^2$ by a regular grid $\mathbf{G} = \{(\mathbf{i}h_n, \mathbf{j}h_n) \mid 0 \leq \mathbf{i}, \mathbf{j} \leq n+1\}$ with $h_n = 1/(n+1)$, we obtain a discretization of problem (5.57), whose objective function is

$$\frac{1}{2}\sum_{\mathbf{i}=0}^{n}\sum_{\mathbf{j}=0}^{n}\Big[h_n^2+(z_{\mathbf{i},\mathbf{j}}-z_{\mathbf{i}+1,\mathbf{j}})^2+(z_{\mathbf{i}+1,\mathbf{j}+1}-z_{\mathbf{i}+1,\mathbf{j}})^2\Big]^{\frac{1}{2}}+\Big[h_n^2+(z_{\mathbf{i},\mathbf{j}}-z_{\mathbf{i},\mathbf{j}+1})^2+(z_{\mathbf{i}+1,\mathbf{j}+1}-z_{\mathbf{i},\mathbf{j}+1})^2\Big]^{\frac{1}{2}},$$

where $z_{\mathbf{i},\mathbf{j}}$ is the value of Φ at grid node $(\mathbf{i}h_n, \mathbf{j}h_n)$. Due to the boundary conditions in (5.57), the discretized problem is n^2-dimensional with $\{z_{\mathbf{i},\mathbf{j}} \mid 1 \le \mathbf{i}, \mathbf{j} \le n\}$ being the decision variables.

To tackle the discretized problem by Algorithms 5.2 and 5.4, we can concatenate $\{z_{\mathbf{i},\mathbf{j}}\}$ into a vector in $\mathbb{R}^{n^2}$ and invoke the generic implementation of these algorithms in Subsection 5.8.1. A better strategy is to pursue a problem-specific implementation, which is explained below.

We partition the variable indices $\{(\mathbf{i}, \mathbf{j}) \mid 1 \le \mathbf{i}, \mathbf{j} \le n\}$ in a way respecting the two-dimensional structure of the variables. Take integers $m_1 \ge 1$, $m_2 \ge 1$, and $n_\mathrm{o} \ge 0$ so that, for simplicity, m_1 and m_2 divide n, and n_o is even. Following (5.55)–(5.56), set $\{X^i\}_{i=1}^{m_1}$ and $\{Y^j\}_{j=1}^{m_2}$ to partition $\{1, 2, \ldots, n\}$, both with overlap size n_o, their restrictions being $\{\tilde{X}^i\}_{i=1}^{m_1}$ and $\{\tilde{Y}^j\}_{j=1}^{m_2}$ respectively. Then we take $\{X^i \times Y^j \mid 1 \le i \le m_1, 1 \le j \le m_2\}$ as an $m_1 \times m_2$ partition of $\{(\mathbf{i}, \mathbf{j}) \mid 1 \le \mathbf{i}, \mathbf{j} \le n\}$, and $\{\tilde{X}^i \times \tilde{Y}^j \mid 1 \le i \le m_1, 1 \le j \le m_2\}$ as its restriction. The multiplicity of this partition is

$$\theta \;=\; \begin{cases} 1 & \text{if } n_\mathrm{o} = 0 \text{ or } m_1 = m_2 = 1, \\ 2 & \text{if } n_\mathrm{o} \ge 1 \text{ and } \max\{m_1, m_2\} > \min\{m_1, m_2\} = 1, \\ 4 & \text{else.} \end{cases}$$

It is not essential to require that $\{X^i\}_{i=1}^{m_1}$ and $\{Y^j\}_{j=1}^{m_2}$ have the same overlap size, yet we do have to ensure $n_\mathrm{o} \le \min\{n/m_1, n/m_2\}$ so that overlaps are limited to neighboring groups.

To devise a coarse space, we note that (5.57) leads to a quasi-linear elliptic boundary value problem by its Euler-Lagrange equation (see, *e.g.*, [44, equation (1.2)]). Hence it is sensible to trial the HEM (Harmonically Enriched Multiscale) coarse space [56], which renders optimal coarse spaces for certain *linear* elliptic problems. For each $(i, j) \in \{1, \ldots, m_1\} \times \{1, \ldots, m_2\}$ and each overlapped index $(\mathbf{i}, \mathbf{j}) \in X^i \times Y^j$, define a function Ψ on $\mathbf{G}$ by assigning $\Psi = 1$ at $(\mathbf{i}h_n, \mathbf{j}h_n)$, setting $\Psi = 0$ outside $(X^i \times Y^j)h_n$, and extending these values to $\mathbf{G}$ harmonically. Our coarse space is spanned by these discrete functions, each of which constitutes a column of the restriction matrix C_k. The dimension of this coarse space[31] is approximately $2nn_\mathrm{o}(m_1+m_2)$, which is much smaller than the dimension n^2 of the variable space as long as $m_1, m_2 \ll n$ and n_o is small. To further reduce the dimension, we subsample the discrete functions by taking only one out of every four of them, leading to a coarse space whose dimension is about $nn_\mathrm{o}(m_1 + m_2)/2$.

We solved a 128^2-dimensional discretization of problem (5.57) by Algorithm 5.4 using both the generic implementation in Subsection 5.8.1 and the problem-specific one elaborated above. We tried coarse spaces SS (5.51), FS (5.52), DF (5.53), CG (5.54), and HEM, the last one equipped only in the problem-specific implementation. The case without coarse space was also tested, Algorithm 5.4 being then reduced to Algorithm 5.2. The starting point was 0. Table 4 displays the numbers of iterations for the two implementations to fulfill $\|g_k\| \le 10^{-6}\|g_0\|$ with coarse spaces varying across the columns and number of subspaces varying across the rows. A "–" is displayed if such a condition was not achieved within 10^4 iterations. To have m subspaces, m being 2, 4, ..., 256 as in the table, the generic implementation applied (5.55)–(5.56) to vector $x = [\, z_{1,1}\,;\, \cdots\,;\, z_{n,1}\,;\, z_{1,2}\,;\, \cdots\,;\, z_{n,2}\,;\, \cdots\,;\, z_{1,n}\,;\, \cdots\,;\, z_{n,n}\,] \in \mathbb{R}^{n^2}$ (where $n = 128$), while the problem-specific implementation took a $\sqrt{m} \times \sqrt{m}$ or $\sqrt{m/2} \times \sqrt{2m}$ two-dimensional partition depending on whether $\sqrt{m}$ is an integer or not. Both implementations set $n_\mathrm{o} = 2$, yet in the second case the overlaps between the variable groups were indeed larger than n_o due to the two-dimensional structure of the partition. RAS was always the overlapping strategy.

[31] There are around $nn_\mathrm{o}(m_1+m_2-2)$ overlapped indices, most belonging to two groups while very few associated with four. Thus the process of defining the discrete function Ψ generates a bit more than $2nn_\mathrm{o}(m_1 + m_2 - 2)$ functions, that is to say approximately $2nn_\mathrm{o}(m_1 + m_2)$ of them.

	Generic implementation					Problem-specific implementation					
	Nil	SS	FS	DF	CG	Nil	SS	FS	DF	CG	HEM
2	243	133	52	52	48	158	63	33	33	50	39
4	318	212	67	67	62	114	63	34	34	51	35
8	–	357	102	98	111	140	89	42	41	65	39
16	1104	683	129	126	179	173	134	53	52	95	38
32	2000	1209	196	182	300	252	195	68	67	136	39
64	–	2022	325	249	536	341	259	80	81	175	57
128	–	–	–	–	–	502	357	97	104	242	60
256	–	–	374	384	275	650	436	156	133	271	81

Table 4: Number of iterations for two implementations of Algorithm 5.4 to solve a discretized Plateau's problem with 128^2 variables (columns: coarse spaces; rows: numbers of subspaces).

Above all, we highlight the outstanding performance of the problem-specific implementation with HEM coarse space. When the number of subspaces was 64, 128, or 256, this implementation enjoyed more than 100 times of speedup in terms of iteration number compared to the generic version without coarse space, which did not achieve the converge criterion even after 10^4 iterations. Comparing the HEM column with all the others in Table 4, it is reasonable to conclude that both the structure-aware decomposition and the problem-dependent coarse space contributed to such a speedup, which confirms the significance of structure-exploiting in the implementation of our space decomposition framework.

The problem-specific implementation outperformed the generic one in all situations as anticipated. The performance of the latter was however acceptable so long as it was equipped with coarse spaces FS, DF, or CG, exception made when there were 128 subspaces. This extremely bad exception is indeed a quite good demonstration for the importance of structure-aware decomposition. Applying partition (5.56) to $x = [\, z_{1,1}\,;\,\cdots\,;\,z_{n,1}\,;\,z_{1,2}\,;\,\cdots\,;\,z_{n,2}\,;\,\cdots\,;\,z_{1,n}\,;\,\cdots\,;\,z_{n,n}\,]$ with $m = n = 128$ and $n_{\mathrm{o}} = 2$, we see for each $\mathbf{j} \in \{2, \cdots, n-1\}$ that the $\mathbf{j}$th group of variables are $\{z_{n,\mathbf{j}-1},\, z_{1,\mathbf{j}},\, z_{2,\mathbf{j}},\, \ldots,\, z_{n,\mathbf{j}},\, z_{1,\mathbf{j}+1}\}$, variable $z_{\mathbf{i},\mathbf{j}}$ losing its coupling with $z_{\mathbf{i},\mathbf{j}-1}$ and $z_{\mathbf{i},\mathbf{j}+1}$ unless $\mathbf{i} = 1$ or $\mathbf{i} = n$. Since $z_{\mathbf{i},\mathbf{j}}$ couples with at most four variables in this problem, such a partition misses roughly half of the coupling information overall. This is significantly worse than the case with $m = n/2 = 64$, where about $1/4$ of the coupling is ignored while $3/4$ retained, which probably contributed to the sudden deterioration of the algorithm from $m = 64$ to $m = 128$.

For both the generic implementation and the problem-specific one, coarse spaces FS, DF, and CG performed much better than SS. The first three worked (unexpectedly) well even with the generic implementation in the 256-subspace case, while SS failed to converge within 10^4 iterations. Such a clear distinction was not observed in Subsection 5.8.3, which testifies the problem-dependent nature of coarse spaces.

We close this subsection by emphasizing that, despite the importance of geometrical structure in this example, our space decomposition framework is *not* particularly designed for this type of problems and has absolutely no dependence on such structure. The generality of our framework has been manifested clearly by the theory in Subsections 5.6–5.7 and by the experiments in Subsections 5.8.2–5.8.3, providing us with a solid basis and a strong motivation to explore our framework on more applications in the investigations yet to come.

6 Optimization based on inaccurate gradients

6.1 Introduction

As elaborated in Subsection 1.2, optimization base on inaccurate gradients is becoming increasingly relevant due to modern applications. However, most classical gradient-based optimization algorithms are designed to work with accurate gradients. We are thus obliged to investigate the behavior of existing algorithms when they are fed with inaccurate gradient information.

Such investigations have been conducted on trust-region methods under a variety of assumptions (see [97, 138, 15, 17, 75] and [30, Section 8.4]), where global convergence is established when the inaccurate gradients satisfy certain consistency conditions (see [97, inequality (4.3)]) or the gradient inaccuracy is below some threshold (see [138, inequality (14)], [15, inequality (5)], [75, Remark 4.1], and [30, Assumption AM.3b]). However, these existing results are asymptotic. To gain deeper insights into the impact of gradient inaccuracy on trust-region methods, we need non-asymptotic results, namely global convergence rates or worst-case complexity bounds.

Interestingly, our analysis on the space transformation framework provides results of this type without extra effort. Suppose that the inaccurate gradient at x_k is $\hat{g}_k$. Then by setting

$$R_k \;=\; \frac{\hat{g}_k g_k^{\mathsf{T}}}{\|g_k\|^2} \;\in\; \mathbb{R}^{n\times n}, \tag{6.1}$$

the inaccurate gradient can be regarded as a linear transformation of the accurate one, that is

$$\hat{g}_k \;=\; R_k g_k.$$

In this way, trust-region methods using inaccurate gradients can be encompassed by Algorithm 3.1 as a particular instance. We note that in this case

$$T_k \;=\; I \;\in\; \mathbb{R}^{n\times n}. \tag{6.2}$$

Therefore, by casting the theory of Algorithm 3.1 to this special instance, we can effortless establish a relative complete theory on the behavior of trust-region methods based on inaccurate gradients. As in [15, 16] and [30, Section 8.4], we focus on the scenario where $\hat{g}_k$ approximates g_k with a bounded relative error, which is a quite common scenario in practice if the gradient evaluation involves solving lower-level problems given finite computing resource and finite precision arithmetic [15, 59, 78]. We will establish the global convergence of trust-region methods in this scenario and, more importantly, obtain their worst-case complexity bounds. It will turn out that gradient inaccuracy, as long as confined within certain limits, does not affect the order of the complexity bounds; in addition, its impact on the multiplicative constants in the bounds is quite mild and completely decided by algorithmic parameters while being *independent* of the optimization problem. On the other hand, once the gradient inaccuracy exceeds the aforementioned limits, the behavior of trust-region methods will deteriorate in a dramatic way that is clearly detectable in computation.

Surely, our results can be obtained by directly investigating $\hat{g}_k$ without considering the transformation R_k. However, the space transformation framework provides a new way of interpreting gradient inaccuracy: rather than regarding the inaccuracy as an error, we perceive it as a transformation. This new perspective reveals similarities between otherwise utterly distinct topics. In addition, research effort is economized by answering seemingly unrelated questions within one single framework.

The structure of this section is as follows. Subsection 6.2 formulates in Algorithm 6.1 a trust-region framework that solves (1.1) based on inaccurate gradients. The framework is identical to classical trust-region methods except that the gradients used in trust-region models are not assumed to be accurate. Directly applying the theory of our space transformation framework, Subsections 6.3 and 6.4 establish the global convergence and worst-case complexities of Algorithm 6.1 when the relative gradient error is below certain bounds that depend solely on the algorithmic parameter η_1 (Theorems 6.1 and 6.2). Subsections 6.3 and 6.4 measure relative gradient error in two different ways, and they are compared and unified in Subsection 6.5, which finds the largest admissible region of inaccurate gradients that guarantees Algorithm 6.1 to converge. Subsection 6.6 discusses briefly the behavior of gradient descent with Armijo line search and inaccurate gradients, showing that such a line-search method is indeed a trust-region method in disguise and hence covered by our theory. Subsection 6.7 will demonstrate numerically the sharpness of the bounds imposed on gradient inaccuracy by Subsections 6.3–6.5, with a focus on Theorem 6.1. Some remarks are included in Subsection 6.8. In addition, as a simple application of our theoretical study, Appendix H shows that the trust-region algorithms built in GNU Octave [32] are unstable with respect to gradient inaccuracy and provides a simple remedy.

Before starting, we emphasize that our aim here is to study the behavior of trust-region methods when inaccurate gradients are given. We will not discuss how to produce such gradients or how to detect the magnitude of gradient inaccuracy, both being extremely important in practice. There is no uniform answer to the former as it is problem-dependent and we refer to [108, Chapter 8] and [100] for discussions, while the latter is investigated in-depth by [99, 101].

6.2 A trust-region framework using inaccurate gradients

Algorithm 6.1 presents a trust-region framework for solving (1.1) based on inaccurate gradient information. Compared with classical trust-region methods, the only difference is that the model gradient $\nabla h_k(0)$ may not be $\nabla f(x_k)$.

Algorithm 6.1 A trust-region framework using inaccurate gradients

Input: Identical to Algorithm 3.1.

For $k = 0, 1, 2, \ldots$, iterate the following steps.

1. Define a model $h_k : \mathbb{R}^n \to \mathbb{R}$ with $\nabla h_k(0) = \hat{g}_k$ for a certain vector $\hat{g}_k \in \mathbb{R}^n$.
2. Calculate $d_k \approx \operatorname{argmin} \{h_k(d) : \|d\| \leq \Delta_k\}$.
3. Define $\rho_k = [f(x_k) - f(x_k + d_k)]/[h_k(0) - h_k(d_k)]$. Update Δ_k according to (3.3) and set

$$x_{k+1} = \begin{cases} x_k & \text{if } \rho_k \leq \eta_0, \\ x_k + d_k & \text{if } \rho_k > \eta_0. \end{cases} \tag{6.3}$$

With R_k and T_k defined in (6.1) and (6.2), Algorithm 6.1 is clearly a special instance of Algorithm 3.1. By casting the theory of Algorithm 3.1 to Algorithm 6.1, we will depict the behavior of Algorithm 6.1 when the relative error in $\hat{g}_k$ is bounded. There exist multiples ways of quantifying such a relative error: it can be defined as $\|\hat{g}_k - g_k\|/\|g_k\|$ according to [78, Section 1.2],

[32] GNU Octave [45] is an open-source clone of MATLAB that is practically used in industry and engineering, *e.g.*, [90]. See `www.gnu.org/software/octave/` for details.

or measured by $\|\hat{g}_k - g_k\|/\|\hat{g}_k\|$ as in [15] and [30, Section 8.4]. We will consider both cases in the following two subsections, and then compare and unify the results in Subsection 6.5.

One can choose any matrix R_k that validates $R_k g_k = \hat{g}_k$ to conduct the analysis, and (6.1) is not the only possibility.[33] Different choices of R_k will not affect the resultant theory, because R_k never appears alone but always in the form of $R_k g_k$ in our analysis of Algorithm 3.1.

Instead of the updating scheme (3.3), one can modify Algorithm 6.1 to manage the trust-region radius Δ_k by (3.65) or by (3.68)–(3.69). According to Subsection 3.6, such modifications will not lead to essential differences in the theory of Algorithm 6.1.

The algorithmic parameter η_1 plays a significant role in our theory. Note that we will always require $\eta_1 < 1$ in the analysis of Algorithm 6.1. This requirement is entirely conventional in classical trust-region methods, although it was not needed in our investigation on Algorithm 3.1, where Assumption 3.7 was imposed instead.

6.3 Inaccurate gradients with bounded relative error of type I

6.3.1 Global convergence and worst-case complexity

Consider the scenario where the relative error $\|\hat{g}_k - g_k\|/\|g_k\|$ is bounded. Casting the theory in Section 3 to Algorithm 6.1, we obtain immediately the following theorem.

Theorem 6.1. *Consider Algorithm 6.1 with $\eta_1 < 1$. Suppose that*

$$\|\hat{g}_k - g_k\| \;\le\; \zeta\|g_k\| \quad \text{for each} \quad k \ge 0, \tag{6.4}$$

where ζ is a constant such that

$$0 \;\le\; \zeta \;<\; \min\{\eta_1^{-1} - 1,\; 1\}. \tag{6.5}$$

Then under Assumptions 3.1–3.4, Algorithm 6.1 enjoys the global convergence elaborated in Theorems 3.1 and 3.7, and it possesses the worst-case complexity bounds quantified in Theorems 3.2–3.4, with κ and μ taking the particular values

$$\kappa(\zeta) \;=\; 1 - \zeta, \tag{6.6}$$

$$\mu(\zeta) \;=\; \min\left\{ \frac{\Delta_0}{\|g_0\|},\; \gamma_0\kappa(\zeta)\left[\frac{\kappa(\zeta)\,[(1+\zeta)^{-1} - \eta_1]}{\sqrt{5}\beta}\right]^{\frac{1}{p}},\; \frac{\alpha\gamma_0\kappa(\zeta)\,[(1+\zeta)^{-1} - \eta_1]}{2[(1+\zeta)^{-1}L_h + L_f]} \right\}. \tag{6.7}$$

To prove Theorem 6.1, we only need to verify that R_k and T_k defined in (6.1) and (6.2) fulfill Assumptions 3.5–3.7 with appropriate values of τ, κ, and λ. We will do this in Appendix F.

After verifying Assumptions 3.5–3.7, Theorems 3.5, 3.6, and 3.8 are also applicable here: when Algorithm 6.1 follows the more general rule (3.65) to update Δ_k, Theorem 6.1 holds after a minor modification to μ; if Algorithm 6.1 maintains Δ_k using (3.68)–(3.69) instead of (3.3), then Theorem 6.1 holds after replacing Theorem 3.7 with its counterpart Theorem 3.8.

The upper bound $\min\{\eta_1^{-1} - 1, 1\}$ in (6.5) cannot be improved in the context of Theorem 6.1, otherwise the global convergence of Algorithm 6.1 is not guaranteed. This will be explained in the remarks succeeding Example 6.1 and demonstrated numerically in Subsection 6.7.

[33] Another instance is $R_k = I + \dfrac{(\hat{g}_k - g_k)g_k^{\mathsf{T}}}{\|g_k\|^2}$.

6.3.2 Impact of gradient inaccuracy

In the typical case of $\eta_1 \leq 1/2$, Theorem 6.1 tells us that Algorithm 6.1 converges as long as $\zeta < 1$ in (6.4). In other words, gradient inaccuracy does not jeopardize the global convergence of the algorithm as long as the magnitude of the inaccuracy is uniformly below that of the true gradient (*i.e.*, the *signal-to-noise ratio* is uniformly higher than 1 : 1, or expressed in decibels, 0 dB).

The complexity part of Theorem 6.1 sheds light on how the gradient inaccuracy affects the non-asymptotic behavior of Algorithm 6.1. Let us take the $\mathcal{O}(\epsilon^{-2})$ bound of Algorithm 6.1 in the general nonconvex case as an example. According to Theorems 3.2 and 6.1, the worst-case complexity of Algorithm 6.1 is

$$k_\epsilon^g \;\leq\; K_\epsilon^g(\zeta) \;\equiv\; \frac{2\log(\gamma_2^{-1}\gamma_1)}{\alpha\,\kappa(\zeta)\,\mu(\zeta)\,\eta_1\log\gamma_1}(f_0 - f_{\inf})\epsilon^{-2} + \frac{\log[\gamma_1\Delta_0^{-1}\mu(\zeta)\epsilon]}{\log\gamma_1}.$$

Note that $K_\epsilon^g(0)$ is exactly the worst-case complexity bound provided by Theorem 3.2 for Algorithm 6.1 when the gradient information is indeed accurate (*i.e.*, the case with $R_k = T_k = I$). Compared with $K_\epsilon^g(0)$, the bound $K_\epsilon^g(\zeta)$ is enlarged by

$$\frac{K_\epsilon^g(\zeta)}{K_\epsilon^g(0)}$$

times due to the gradient inaccuracy. Since the first term in $K_\epsilon^g(\zeta)$ is dominant when ϵ is small, it is reasonable to expect that

$$\frac{K_\epsilon^g(\zeta)}{K_\epsilon^g(0)} \;\leq\; \frac{\kappa(0)\,\mu(0)}{\kappa(\zeta)\,\mu(\zeta)} \tag{6.8}$$

as long as ϵ is not too large. This is indeed true when $\epsilon \leq \gamma_1^{-1}\mathrm{e}^{-1}\|g_0\|$. In this case,

$$0 \;<\; \gamma_1\Delta_0^{-1}\mu(\zeta)\epsilon \;\leq\; \gamma_1\Delta_0^{-1}\mu(0)\epsilon \;\leq\; \mathrm{e}^{-1} \quad \text{(note that } \mu(0) = \mu \leq \Delta_0/\|g_0\| \text{ by (3.30))}.$$

Since function $t \mapsto t\log t$ is decreasing when $0 < t \leq \mathrm{e}^{-1}$, we know that

$$[\gamma_1\Delta_0^{-1}\mu(\zeta)\epsilon]\log[\gamma_1\Delta_0^{-1}\mu(\zeta)\epsilon] \;\geq\; [\gamma_1\Delta_0^{-1}\mu(0)\epsilon]\log[\gamma_1\Delta_0^{-1}\mu(0)\epsilon],$$

which implies

$$\frac{\log[\gamma_1\Delta_0^{-1}\mu(\zeta)\epsilon]}{\log\gamma_1} \;\leq\; \frac{\mu(0)}{\mu(\zeta)}\cdot\frac{\log[\gamma_1\Delta_0^{-1}\mu(0)\epsilon]}{\log\gamma_1} \;\leq\; \frac{\mu(0)\,\kappa(0)}{\mu(\zeta)\,\kappa(\zeta)}\cdot\frac{\log[\gamma_1\Delta_0^{-1}\mu(0)\epsilon]}{\log\gamma_1} \tag{6.9}$$

because $\log\gamma_1 < 0$, $\log[\gamma_1\Delta_0^{-1}\mu(\zeta)\epsilon] < 0$, and $\kappa(0)/\kappa(\zeta) \geq 1$. With (6.9), one can verify (6.8) easily. Plugging the values of $\kappa(\zeta)$ and $\mu(\zeta)$ given by (6.6) and (6.7) into (6.8), we obtain

$$\frac{K_\epsilon^g(\zeta)}{K_\epsilon^g(0)} \;\leq\; \frac{1}{(1-\zeta)^2}\max\left\{\left[\frac{(1+\zeta)(1-\eta_1)}{(1-\zeta)[1-\eta_1(1+\zeta)]}\right]^{\frac{1}{p}},\; \frac{(1+\zeta)(1-\eta_1)}{1-\eta_1(1+\zeta)}\right\}. \tag{6.10}$$

Note that, in addition to the relative error magnitude ζ, the upper bound presented on the right-hand side of (6.10) depends solely on the algorithmic constant η_1 and the order p in Assumption 3.4. Remarkably, this bound is *independent of the optimization problem* (1.1) itself.

The same analysis can be done for the $\mathcal{O}(\epsilon^{-1})$ and $\mathcal{O}(\log(\epsilon^{-1}))$ bounds of Algorithm 6.1 in the convex and strongly convex cases, and it turns out that the factor (6.10) are applicable to these two cases as well, provided that ϵ is not too large.

As an illustration, consider the scenario that the gradient evaluation has *only one correct significant digit*.[34] Then $\zeta = 1/2$ in the worst case. Suppose that $\eta_1 = 1/4$, which is typical, and $p = 1/2$, which is normal in practice according to Proposition 3.1. Then the worst-case complexities of Algorithm 6.1 have the same orders as when the gradient is accurate; moreover, the enlargement factor (6.10) is at most

$$\frac{(1+\zeta)^2(1-\eta_1)^2}{(1-\zeta)^4[1-\eta_1(1+\zeta)]^2} \;<\; 52,$$

which is clearly independent of problem (1.1) and quite *benign*.

6.4 Inaccurate gradients with bounded relative error of type II

6.4.1 Global convergence and worst-case complexity

Now we switch our attention to the relative error measured by $\|\hat{g}_k - g_k\|/\|\hat{g}_k\|$. Parallel to Theorem 6.1, we have the following theorem.

Theorem 6.2. *Consider Algorithm 6.1 with $\eta_1 < 1$. Suppose that*

$$\|\hat{g}_k - g_k\| \;\le\; \zeta\|\hat{g}_k\| \quad \text{for each} \quad k \ge 0, \tag{6.11}$$

where ζ is a constant such that

$$0 \;\le\; \zeta \;<\; 1 - \eta_1. \tag{6.12}$$

Then under Assumptions 3.1–3.4, Algorithm 6.1 enjoys the global convergence elaborated in Theorems 3.1 and 3.7, and it possesses the worst-case complexity bounds quantified in Theorems 3.2–3.4, with κ and μ taking the particular values

$$\begin{aligned}
\kappa(\zeta) \;&=\; (1+\zeta)^{-1}, \\
\mu(\zeta) \;&=\; \min\left\{ \frac{\Delta_0}{\|g_0\|},\; \gamma_0\kappa(\zeta)\left[\frac{\kappa(\zeta)\,(1-\zeta-\eta_1)}{\sqrt{5}\beta}\right]^{\frac{1}{p}},\; \frac{\alpha\gamma_0\kappa(\zeta)\,(1-\zeta-\eta_1)}{2[(1-\zeta)L_h + L_f]} \right\}.
\end{aligned}$$

Theorem 6.2 is proved in Appendix F by verifying Assumptions 3.5–3.7 with appropriate values of τ, κ, and λ. After such verification, Theorems 3.5, 3.6, and 3.8 are also applicable here: if Algorithm 6.1 follows the more general rule (3.65) to update Δ_k, then Theorem 6.2 holds after a minor modification to μ; if Algorithm 6.1 maintains Δ_k using (3.68)–(3.69) instead of (3.3), then Theorem 6.2 holds after replacing Theorem 3.7 with its counterpart Theorem 3.8.

The upper bound $1 - \eta_1$ in (6.12) cannot be improved in the context of Theorem 6.2, otherwise the global convergence of Algorithm 6.1 is not guaranteed. See the remarks succeeding Example 6.1 for the reason.

[34] Let $\hat{x}$ be an approximation to a real number x. According to [78, Section 1.2], there exist different ways to define "$\hat{x}$ has one significant digit". One possible definition is that $\hat{x}$ and x can be rounded to the same number with a single significant digit, where rounding is the operation of replacing a given number by the nearest number with a desired amount of significant digits. Another definition requires $|x - \hat{x}|$ to be less than half a unit in the first significant digit of x. In both cases, $|\hat{x} - x| \le |x|/2$. We say that $\hat{g}_k$ has one correct significant digit if each component of $\hat{g}_k$ has one correct significant digit as an approximation to the corresponding component of g_k. This is corresponding to the *componentwise relative error* widely used in error/perturbation analysis [78, Section 1.2].

Conditions (6.11)–(6.12) were proposed by [15] to establish the global convergence (but not the complexity) of trust-region methods when the gradients are inaccurate. [30, Theorems 8.4.5 and 8.4.7] also proved such convergence under a similar circumstance. Assumption 3.4 is not needed in [30], because an assumption stronger than (6.11)–(6.12) is imposed upon the gradient inaccuracy. Using our notation, [30] requires that the constant ζ in (6.11) satisfies

$$0 \;\leq\; \zeta \;<\; \frac{\alpha}{2}(1-\eta_1), \tag{6.13}$$

which can be seen from Assumptions AN.1, AA.1, and AM.3b of [30]. Since $\alpha \in (0,1]$, inequality (6.13) is strictly stronger than (6.12). When the trust-region step d_k reduces the model h_k at least as much as the Cauchy step does, which is normally the case in practice, Assumption 3.4 holds automatically (see Proposition 3.1) but inequality (6.13) is still stricter than (6.12).

Although (6.12) and (6.13) differ by only a constant factor $\alpha/2$, the practical implication of such a difference is not negligible. Taking squares,[35] we can see for all $t \in [0,\ 1)$ that

$$\|\hat{g}_k - g_k\| < t\|\hat{g}_k\| \quad\Longleftrightarrow\quad \left\|\hat{g}_k - \frac{g_k}{1-t^2}\right\| < \frac{t}{1-t^2}\,\|g_k\|. \tag{6.14}$$

Hence, for a given g_k, $\{\hat{g}_k : \|\hat{g}_k - g_k\| < t\|\hat{g}_k\|\}$ specifies a ball with radius $t\|g_k\|/(1-t^2)$. Enlarging t from $\mathsf{t} = \alpha(1-\eta_1)/2$ to $\mathsf{T} = 1-\eta_1$, the volume of this ball is multiplied by

$$\left(\frac{\mathsf{T}}{1-\mathsf{T}^2}\cdot\frac{1-\mathsf{t}^2}{\mathsf{t}}\right)^n \;>\; \left(\frac{\mathsf{T}}{\mathsf{t}}\right)^n = \left(\frac{2}{\alpha}\right)^n \;\geq\; 2^n.$$

Therefore, if $\hat{g}_k$ is generated by some random perturbation of g_k, it can be significantly easier to fulfill $\|\hat{g}_k - g_k\| < (1-\eta_1)\|\hat{g}_k\|$ than achieving $\|\hat{g}_k - g_k\| < \alpha(1-\eta_1)\|\hat{g}_k\|/2$.

However, we emphasize that the discussion in [30] is indeed much more general, covering much broader situations, while what we mention here is but its specialization to our context.

6.4.2 Impact of gradient inaccuracy

Theorem 6.2 tells us that Algorithm 6.1 converge as long as the noise contained in the gradient information takes a proportion uniformly lower than $1-\eta_1$, which is 75% if $\eta_1 = 1/4$.

The complexity bounds revealed in Theorem 6.2 tell us again how the gradient inaccuracy affects the worst-case iteration complexities of 6.1. Following the argument in Subsection 6.3.2, by checking $[\kappa(0)\mu(0)]/[\kappa(\zeta)\mu(\zeta)]$, we will see that the gradient inaccuracy subject to (6.11) enlarges the complexity bounds by a factor of at most

$$(1+\zeta)^2 \max\left\{\left[\frac{(1+\zeta)(1-\eta_1)}{1-\zeta-\eta_1}\right]^{\frac{1}{p}},\ \frac{1-\eta_1}{1-\zeta-\eta_1}\right\}, \tag{6.15}$$

provided that the threshold ϵ for stationarity is not too large. Again, the factor (6.15) is entirely determined by the relative error magnitude ζ, the algorithmic parameter η_1, and the order p in Assumption 3.4, while being completely *independent of the optimization problem* (1.1).

As an illustration, suppose that $\eta_1 = 1/4$, $p = 1/2$, and $\zeta = 55\%$, allowing the noise in the gradient information to have higher magnitude than the true gradient (a scenario not covered by Theorem 6.1), the factor in (6.15) is less than 82.

[35] $\|\hat{g}_k - g_k\| < t\|\hat{g}_k\| \Longleftrightarrow (1-t^2)\|\hat{g}_k\|^2 - 2g_k^{\mathsf{T}}\hat{g} + \|g_k\|^2 < 0 \Longleftrightarrow \left\|\sqrt{1-t^2}\hat{g}_k - \frac{g_k}{\sqrt{1-t^2}}\right\|^2 < \frac{t^2}{1-t^2}\|g_k\|^2$.

6.5 The largest admissible region of inaccurate gradients

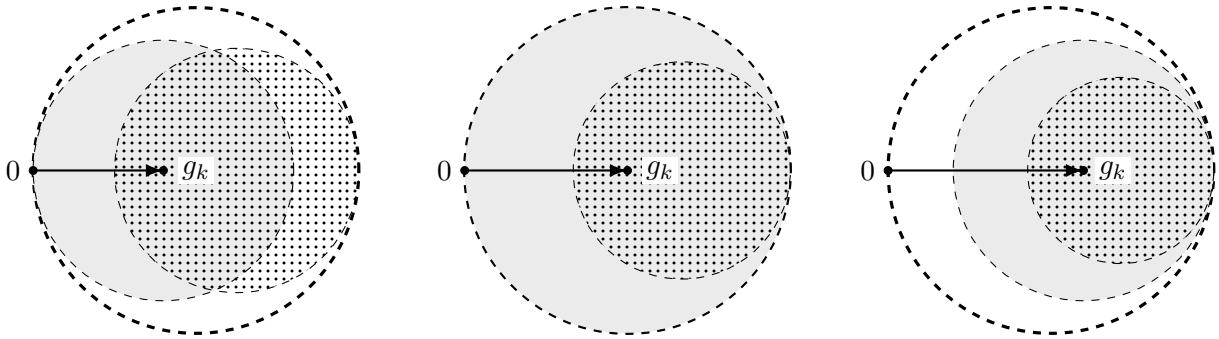
Theorems 6.1 and 6.2 depict the behavior of Algorithm 6.1 based on two different ways of measuring relative error. It is natural to compare the two theorems and see whether one of them is stronger. To this end, let us consider the following regions of inaccurate gradient $\hat{g}_k$:

$$\{\hat{g}_k : \|\hat{g}_k - g_k\| < \min\{\eta_1^{-1} - 1,\, 1\}\|g_k\|\}, \tag{6.16}$$
$$\{\hat{g}_k : \|\hat{g}_k - g_k\| < (1 - \eta_1)\|\hat{g}_k\|\}, \tag{6.17}$$

which are entirely specified by the true gradient g_k and the algorithmic parameter η_1. They are the upper limits for the regions of $\hat{g}_k$ presented in Theorems 6.1 and 6.2. Region (6.16) is a ball. This is also the case for region (6.17): setting $t = 1 - \eta_1$ in (6.14), we have

$$\|\hat{g}_k - g_k\| < (1 - \eta_1)\|\hat{g}_k\| \quad \Longleftrightarrow \quad \left\|\hat{g}_k - \frac{g_k}{2\eta_1 - \eta_1^2}\right\| < \frac{1 - \eta_1}{2\eta_1 - \eta_1^2}\, \|g_k\|.$$

For $n = 2$, Figure 5 illustrates the regions (6.16) and (6.17) when η_1 varies.[36] If $0 < \eta_1 < 1/2$, which is typically true in practice, neither one of two regions contains the other as a subset, and hence neither Theorems 6.1 nor 6.2 is stronger than the other; when $1/2 \le \eta_1 < 1$, region (6.16) contains region (6.17), and hence Theorem 6.1 is stronger than Theorem 6.2.



(a) $0 < \eta_1 < \frac{1}{2}$ (b) $\eta_1 = \frac{1}{2}$ (c) $\frac{1}{2} < \eta_1 < 1$

Legend:

$\|\hat{g}_k - g_k\| < \min\{\eta_1^{-1} - 1, 1\}\|g_k\|$ $\|\hat{g}_k - g_k\| < (1 - \eta_1)\|\hat{g}_k\|$ $\left\|\hat{g}_k - \frac{g_k}{2\eta_1}\right\| < \frac{\|g_k\|}{2\eta_1}$

Figure 5: Admissible regions for the inaccurate gradient $\hat{g}_k$ with different values of η_1.

Now consider the circumscribed circle of the union of (6.16) and (6.17) assuming that $n = 2$ and $0 < \eta_1 < 1/2$. As illustrated in Figure 5(a), under such a setting, the leftmost point in the union is 0, which comes from (6.16); the rightmost one[37] is g_k/η_1, which comes from (6.17).

[36] As illustrations, Figure 5 takes $\eta_1 = 2/5$ and $3/5$ in subfigures (a) and (c) respectively. Note that the subfigures do not visualize g_k with the same length. Region (6.18) is also shown in Figure 5 in addition to (6.16) and (6.17). When $\eta_1 = 1/2$, regions (6.16) and (6.18) become identical as in subfigure (b).

[37] For any $\hat{g}_k$ in (6.17), we have $\|g_k\| > \eta_1\|\hat{g}_k\|$ by triangle inequality, and hence $\|\hat{g}_k\| < \|g_k\|/\eta_1$. Therefore, in (6.17), the point farthest away from 0 is g_k/η_1, which, when $n = 2$, gives the rightmost point in Figure 5(a).

Hence the circumscribed circle is $\{\hat{g}_k : \|\hat{g}_k - g_k/(2\eta_1)\| = \|g_k\|/(2\eta_1)\}$, and the region it encloses is

$$\left\{\hat{g}_k : \left\|\hat{g}_k - \frac{g_k}{2\eta_1}\right\| < \frac{\|g_k\|}{2\eta_1}\right\}, \tag{6.18}$$

as illustrated in Figure 5. It turns out that region (6.18) is indeed the *largest admissible region* for the inaccurate gradient $\hat{g}_k$ in Algorithm 6.1, for any $n \geq 1$ and any $\eta_1 \in (0, 1)$. Roughly speaking, if $\hat{g}_k$ stays *uniformly* inside region (6.18) for each $k \geq 0$, then Algorithm 6.1 behaves essentially the same as when the gradients are accurate (see Theorem 6.3); in contrast, if the inequality in (6.18) is violated for all $k \geq 0$, then Algorithm 6.1 is not globally convergent in general (see Example 6.1). It is worth noting that all the three regions (6.16), (6.17), and (6.18) are *entirely decided* by η_1 and the accurate gradient g_k, and they all expand when η_1 shrinks. In this sense, a *smaller* value of η_1 enables Algorithm 6.1 to tolerate *higher* inaccuracy in gradients.

Theorem 6.3. *If there exists a constant $\zeta \in [0, 1)$ such that*

$$\left\|\hat{g}_k - \frac{g_k}{2\eta_1}\right\| \leq \frac{\zeta}{2\eta_1}\|g_k\| \quad \text{for each} \quad k \geq 0, \tag{6.19}$$

then the transformations $\{R_k\}$ and $\{T_k\}$ defined in (6.1) and (6.2) fulfill Assumptions 3.5–3.7 with $\tau = 1$, $\kappa = (1 - \zeta)/(2\eta_1)$, and $\lambda = 2\eta_1/(1 + \zeta)$. Consequently, under Assumptions 3.1–3.4, Algorithm 6.1 enjoys the global convergence elaborated in Theorems 3.1 and 3.7, and it possess the worst-case complexity bounds quantified in Theorems 3.2–3.4.

The proof of Theorem 6.3 is in Appendix F. We can also cast Theorems 3.5, 3.6, and 3.8 to this case: if Algorithm 6.1 updates Δ_k following (3.65) instead of (3.3), then Theorem 6.3 holds; if Algorithm 6.1 maintains Δ_k using (3.68)–(3.69) instead of (3.3), then Theorem 6.3 holds with Theorem 3.7 replaced by Theorem 3.8.

Example 6.1. *Apply Algorithm 6.1 to function $f(x) = \|x\|^2/2$. Suppose that, for each $k \geq 0$, the model is $h_k(d) = \hat{g}_k^{\mathsf{T}} d + \|d\|^2/2$ with a* nonzero *vector $\hat{g}_k \in \mathbb{R}^n$ outside region (6.18), namely*

$$\left\|\hat{g}_k - \frac{g_k}{2\eta_1}\right\| \geq \frac{\|g_k\|}{2\eta_1}, \tag{6.20}$$

and d_k solves the trust-region subproblem $\min_{\|d\| \leq \Delta_k} h_k(d)$ exactly. Then $\{x_k\}$ converges, and

$$\left\|\lim_{k\to\infty} x_k - x_0\right\| \leq \frac{\Delta_0}{1 - \gamma_1}. \tag{6.21}$$

Hence $\{x_k\}$ cannot converge to the unique stationary point 0 unless x_0 is sufficiently close to it.

Indeed, (6.20) makes $\rho_k > \eta_1$ unattainable, and hence $\{\Delta_k\}$ decays geometrically according to (3.3), forcing $\{x_k\}$ to converge to a point in a neighborhood of x_0. Details are explained below as the justification for Example 6.1.

Justification. Let k an arbitrary iteration index. Taking squares, we see (6.20) equivalent to

$$g_k^{\mathsf{T}} \hat{g}_k \leq \eta_1 \|\hat{g}_k\|^2. \tag{6.22}$$

Since $h_k(d) = \hat{g}_k^T d + \|d\|^2/2$ and d_k solves $\min_{\|d\| \le \Delta_k} h_k(d)$ exactly, we have $d_k = -t_k \hat{g}_k$ for a certain scalar $t_k > 0$. Hence (6.22) leads to

$$-d_k^T g_k \;\le\; -\eta_1 d_k^T \hat{g}_k. \tag{6.23}$$

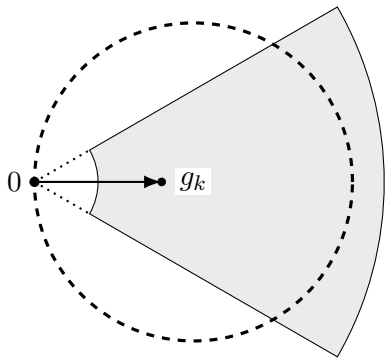
Since f and h_k are quadratic, their second-order Taylor expansions are exact, that is

$$f(x_k) - f(x_k + d_k) \;=\; -d_k^T g_k - \frac{1}{2}\|d_k\|^2 \quad \text{and} \quad h_k(0) - h_k(d_k) \;=\; -d_k^T \hat{g}_k - \frac{1}{2}\|d_k\|^2.$$

Combining these expansions with (6.23) and noting that $\eta_1 < 1$, we have

$$f(x_k) - f(x_k + d_k) \;\le\; \eta_1 [h_k(0) - h_k(d_k)]. \tag{6.24}$$

Since $h_k(0) - h_k(d_k) > 0$, (6.24) shows that $\rho_k > \eta_1$ is never achieved. Hence by the updating rule (3.3) of Δ_k, we have $\Delta_k \le \gamma_1^k \Delta_0$ for each iteration. Therefore, $\{x_k\}$ is a Cauchy sequence and hence convergent, with (6.21) justified by $\sum_{k=0}^{\infty} \Delta_k \le \sum_{k=0}^{\infty} \gamma_1^k \Delta_0 = \Delta_0/(1-\gamma_1)$. □



Legend:

◯ Region defined by (6.25) with $c > \eta_1^{-1}$ ⋯ Region defined by (6.18)

Figure 6: Inaccurate gradients fulfilling (6.25) may not be in the admissible region (6.18).

One may expect that Algorithm 6.1 converges provided that $\hat{g}_k$ and g_k make an acute angle uniformly smaller than $\pi/2$ and $\|\hat{g}_k\|/\|g_k\|$ stays bounded away from zero and infinity, namely

$$g_k^T \hat{g}_k \;\ge\; a\|g_k\|\|\hat{g}_k\| \quad \text{and} \quad b\|g_k\| \;\le\; \|\hat{g}_k\| \;\le\; c\|g_k\| \tag{6.25}$$

for each $k \ge 0$ with certain positive constants a, b, and c. Yet Example 6.1 shows that such an expectation is wrong, because the region defined by (6.25) is not a subset of (6.18) if the constant c is larger than η_1^{-1} (see Figure 6 for an illustration with $\eta_1 = 2/5$, $a = \sqrt{3}/2$, $b = 1/2$, and $c = 3$). Recalling that Algorithm 6.1 is a special case of Algorithm 3.1 with R_k and T_k defined in (6.1) and (6.2), we also see that the length and angle conditions (3.4)–(3.5) cannot ensure the convergence of Algorithm 3.1. Further more, with R_k and T_k in (6.1) and (6.2), the reformulation (6.22) of (6.20) becomes exactly

$$g_k^T T_k R_k g_k \;\le\; \eta_1 \|R_k g_k\|^2.$$

Thus Example 6.1 shows that Algorithm 3.1 may fail to converge without Assumption 3.7.

Example 6.1 also confirms that the constants $\min\{\eta_1^{-1}-1, 1\}$ and $1-\eta_1$ in Theorems 6.1 and 6.2 are *not improvable*. If we replace them with larger values, then the regions specified in (6.4) and (6.11) will not entirely lie in region (6.18) (see Figure 5), and then the global converge of Algorithm 6.1 is not guaranteed. This will be demonstrated numerically in Subsection 6.7.

6.6 Gradient descent with Armijo line search and inaccurate gradients

One may suspect that the admissible region (6.18) is unique to trust-region methods and will not emerge if line search is in use. However, we will show that the same region comes naturally into play if one examines gradient descent based on Armijo line search and inaccurate gradients. Indeed, gradient descent based on Armijo line search turns out a disguised trust-region method.

For a later-revealed reason, let us use t instead of k to denote an iteration counter. At iteration t, an Airmijo line-search method chooses a search direction p_t, and then sets $x_{t+1} = x_t + \Upsilon_t p_t$ with a step size Υ_t fulfilling the Armijo condition, namely

$$f(x_t + \Upsilon_t p_t) \;<\; f(x_t) + \eta_1 \Upsilon_t p_t^{\mathsf{T}} g_t. \tag{6.26}$$

Typically, Υ_t is obtained by backtracking (*e.g.*, [108, Algorithm 3.1]). A gradient descent method will take $p_t = -g_t$, and condition (6.26) then reduces to $f(x_t - \Upsilon_t g_t) < f(x_t) - \eta_1 \Upsilon_t \|g_t\|^2$. Therefore, we can formulate a simple gradient descent algorithm using Armijo line search and inaccurate gradients into Algorithm 6.2.

Algorithm 6.2 An Armijo line-search gradient descent algorithm using inaccurate gradients

Input: $x_0 \in \mathbb{R}^n$, $\Upsilon_0 > 0$, $\eta_1 \in (0, 1)$.

For $t = 1, 2, 3, \ldots$, iterate the following steps.

1. Choose a certain vector $\hat{g}_t \in \mathbb{R}^n$.
2. Set $x_{t+1} = x_t - \Upsilon_t \hat{g}_t$, where Υ_t is the largest number in $\{\Upsilon_{t-1}, 2^{-1}\Upsilon_{t-1}, 2^{-2}\Upsilon_{t-1}, \ldots\}$ with

$$f(x_t - \Upsilon_t \hat{g}_t) \;<\; f(x_t) - \eta_1 \Upsilon_t \|\hat{g}_t\|^2. \tag{6.27}$$

Terminate if no such Υ_t exists.

The right-hand side of (6.27) involves $\hat{g}_t$ instead of g_t, the latter being unavailable in this context. Thus Algorithm 6.2 differs from an iterative process that sets $x_{t+1} = x_t + \Upsilon_t p_t$ with a direction $p_t = -\hat{g}_t \approx -g_t$ and a step size Υ_t satisfying (6.26), which does converge provided that $\hat{g}_t$ and g_t make an acute angle bounded away from $\pi/2$ and $\|\hat{g}_t\|/\|g_t\|$ remains bounded away from zero. Such a process is not feasible without access to accurate gradients due to (6.26).

If Algorithm 6.2 does not always take the inaccurate gradient from the admissible region (6.18) (k changed to t), then it can fail to converge due to the following simple observation.

Proposition 6.1. *If f is* convex, *Algorithm 6.2 terminates once it encounters an iteration with*

$$\left\| \hat{g}_t - \frac{g_t}{2\eta_1} \right\| \;\geq\; \frac{\|g_t\|}{2\eta_1}. \tag{6.28}$$

Proof. As (6.22), we can reformulate (6.28) into $g_t^{\mathsf{T}} \hat{g}_t \leq \eta_1 \|\hat{g}_t\|^2$. Hence, for a convex function f, once (6.28) happens, we have

$$f(x_t - \Upsilon \hat{g}_t) - f(x_t) \;\geq\; -\Upsilon \hat{g}_t^{\mathsf{T}} g_t \;\geq\; -\Upsilon \eta_1 \|\hat{g}_t\|^2 \quad \text{for all} \;\; \Upsilon > 0.$$

Therefore, the Armijo condition (6.27) is not achievable and Algorithm 6.2 terminates. □

In the scenario described by Proposition 6.1, Algorithm 6.2 fails unless x_t is stationary. This is still true if we replace (6.27) with any condition even stronger (the backtracking is not essential), including the (strong) Wolfe conditions and the Goldstein condition [108, Section 3.1].

On the other hand, if the inaccurate gradient stays *uniformly* inside the admissible region in the sense of (6.19), then the convergence theory of Algorithm 6.2 is covered by Subsection 6.5. Indeed, it is straightforward to verify that Algorithm 6.2 can be reformulated into Algorithm 6.3, which is a particular instance of Algorithm 6.1 except that Δ_k is maintained by (3.68)–(3.69) rather than (3.3). Algorithm 6.3 enjoys the global convergence in Theorems 3.1 and 3.8 as well as the worst-case complexities in Theorems 3.2–3.4 according to the comments after Theorem 6.3.

Algorithm 6.3 A trust-region reformulation of Algorithm 6.2

Input: $x_0 \in \mathbb{R}^n$, $\Upsilon_0 > 0$, $\eta_1 \in (0, 1)$.

For $k = 0, 1, 2, \ldots$, iterate the following steps.

1. Set $h_k(d) = d^{\mathsf{T}}\hat{g}_k$ and $\Delta_k = \Upsilon_k\|\hat{g}_k\|$.
2. Calculate $d_k = \operatorname{argmin}\left\{h_k(d) : \|d\| \leq \Delta_k\right\}$.
3. Define $\rho_k = [f(x_k) - f(x_k + d_k)]/[h_k(0) - h_k(d_k)]$. Set

$$x_{k+1} = \begin{cases} x_k & \text{if } \rho_k \leq \eta_1, \\ x_k + d_k & \text{if } \rho_k > \eta_1, \end{cases} \quad \text{and} \quad \Upsilon_k = \begin{cases} \Upsilon_k/2 & \text{if } \rho_k \leq \eta_1, \\ \Upsilon_k & \text{if } \rho_k > \eta_1. \end{cases}$$

Algorithms 6.2 and 6.3 do not count iterations at the same pace: the former defines an iteration as the full backtracking process for achieving the Armijo condition, while the latter takes each trial in the backtracking as an individual iteration. For Algorithm 6.3, defining k_t as the counter of the tth iteration fulfilling $\rho_k > \eta_1$, then x_{k_t} and Υ_{k_t} correspond respectively to x_t and Υ_t in Algorithm 6.2. Thus we use different symbols to denote their iteration counters.

To summarize, although Algorithm 6.2 is of line-search type, its global convergence requires the inaccurate gradient to stay in the admissible region (6.18). If the inaccurate gradient stays uniformly inside (6.18), then Algorithm 6.2 converges globally with virtually the same worst-case complexities as the accurate-gradient case. Consequently, Algorithm 6.2 tolerates relative gradient error below $\min\{\eta_1^{-1} - 1, 1\}$ measured by $\|\hat{g}_t - g_t\|/\|g_t\|$ or $1 - \eta_1$ by $\|\hat{g}_t - g_t\|/\|\hat{g}_t\|$, both bounds being not improvable, and Theorems 6.1–6.2 hold for Algorithm 6.2 after minor modifications to $\kappa(\zeta)$ and $\mu(\zeta)$.

Therefore, in the smooth yet not necessarily convex case, we have effortlessly established a relatively complete theory for gradient descent with Armijo line search and inaccurate gradients. It demonstrates again the versatility of our space transformation framework.

6.7 Numerical experiment

Example 6.1 indicates that we cannot improve the bounds imposed by Theorems 6.1–6.3 on inaccurate gradients, but the objective function in Example 6.1 is extremely special. Now we will show that such bounds are also clearly observable for other functions in numerical computations.

As an illustration, we will focus on Theorem 6.1. The theorem reveals that, in terms of $\|\hat{g}_k - g_k\|/\|g_k\|$, Algorithm 6.1 tolerates any relative error uniformly below $\min\{\eta_1^{-1} - 1, 1\}$.

If this bound is sharp, then the performance of Algorithm 6.1 should *deteriorate dramatically* once it is violated. We will demonstrate such a phenomenon by observing the effectiveness the algorithm with different values of η_1 on problems with various levels of relative gradient error.

6.7.1 Quantifying the effectiveness of an algorithm

Before our experiment, we need first to clarify and quantify what we mean by the effectiveness of an algorithm. Suppose that an algorithm $\mathcal{A}$ is applied to the minimization of a smooth function f without constraints, the output being x_{out}. Then it is reasonable to say that $\mathcal{A}$ is effective on f up to an optimality tolerance $\varepsilon \in (0, 1)$ if

$$\|\nabla f(x_{\text{out}})\| \;\leq\; \varepsilon \|\nabla f(x_0)\|,$$

yet we need to measure the effectiveness with finer granularity. To this end, note that the decimal form of $\|\nabla f(x_{\text{out}})\|/\|\nabla f(x_0)\|$ contains roughly $\left[-\log_{10}\left(\|\nabla f(x_{\text{out}})\|/\|\nabla f(x_0)\|\right)\right]$ leading zeros counted from the digit before the decimal point, while the optimality tolerance ε indicates that we indeed desire $(-\log_{10}\varepsilon)$ such zeros. Hence we define the ε-effectiveness of $\mathcal{A}$ on f as

$$\mathsf{e}_\varepsilon(\mathcal{A}, f) \;=\; \min\left\{1,\; \frac{1}{\log \varepsilon} \log\left[\frac{\|\nabla f(x_{\text{out}})\|}{\|\nabla f(x_0)\|}\right]\right\},$$

which is the proportion of leading zeros achieved by $\mathcal{A}$. Clearly, $\mathsf{e}_\varepsilon(\mathcal{A}, f) = 1$ if and only if $\mathcal{A}$ is effective on f up to ε. The ε-effectiveness of $\mathcal{A}$ on a set $\mathcal{F}$ of functions is defined as

$$\mathsf{E}_\varepsilon(\mathcal{A}, \mathcal{F}) \;=\; \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} \mathsf{e}_\varepsilon(\mathcal{A}, f). \tag{6.29}$$

When ε and $\mathcal{F}$ are fixed in a context, for brevity, we will call $\mathsf{E}_\varepsilon(\mathcal{A}, \mathcal{F})$ the effectiveness of $\mathcal{A}$.

6.7.2 General settings of the experiment

Focusing on Theorem 6.1, the relative gradient error at x_k is always defined as $\|\hat{g}_k - g_k\|/\|g_k\|$ and we will not repeat this fact. Since Theorem 6.1 only covers the situation where the error is below a certain magnitude ζ along *all the iterations*, it is reasonable to investigate its sharpness by observing the scenario where

$$\frac{\|\hat{g}_k - g_k\|}{\|g_k\|} \;=\; \zeta \quad \text{for each} \quad k \geq 0.$$

To have such a scenario, given ζ and x, we *artificially* generate an inaccurate gradient for f by

$$\hat{g}(x) \;=\; \nabla f(x) + \zeta \|\nabla f(x)\|\, e, \tag{6.30}$$

where e is a sample from the uniform distribution on $\mathbb{S}^{n-1}$, drawn independently for different x. It does not matter whether $\hat{g}$ produces the same result or not when repeatedly evaluated at a single x, as the algorithm tested here will not involve gradient re-evaluation at any point.

We implement a vanilla version of Algorithm 6.1. Our experiment is mainly concerned with the trust-region radius update, for which we adopt a simple yet commonly used scheme, namely

$$\Delta_{k+1} \;=\; \begin{cases} \gamma_1 \Delta_k & \text{if } \rho_k \leq \eta_1, \\ \Delta_k & \text{if } \eta_1 < \rho_k \leq \eta_2, \\ \gamma_2 \Delta_k & \text{if } \rho_k > \eta_2, \end{cases} \tag{6.31}$$

with parameters η_1 and η_2 between 0 and 1. Various values of $\eta_1 \in (0, 1)$ will be tested, and η_2 typically should be larger than η_1 and $3/4 \leq \eta_2 \leq 1$, so we set $\eta_2 = \max\{3/4, \min\{3\eta_1/2, 1\}\}$. To define the trust-region model h_k, we follow the suggestion in [108, Section 6.2] and take

$$h_k(d) \;=\; d^{\mathsf{T}}\hat{g}(x_k) + \frac{1}{2}d^{\mathsf{T}}B_k d$$

with B_k updated by the SR1 quasi-Newton method. Specifically, as in [108, Algorithm 6.2],

$$B_{k+1} \;=\; \frac{(y_k - B_k d_k)(y_k - B_k d_k)^{\mathsf{T}}}{d_k^{\mathsf{T}}(y_k - B_k d_k)} \quad \text{if} \quad |d_k^{\mathsf{T}}(y_k - B_k d_k)| \;\geq\; r\|d_k\|\|y_k - B_k d_k\|, \tag{6.32}$$

where $r \in (0, 1)$ is prescribed, and y_k is the change in gradient corresponding to d_k, namely

$$y_k = \hat{g}(x_k + d_k) - \hat{g}(x_k).$$

Note that (6.32) is applied even if $x_{k+1} = x_k$. If the condition in (6.32) fails, then set $B_{k+1} = B_k$. The update is restarted by setting $B_{k+1} = B_0$ when (6.32) leads to a matrix with elements above a safeguarding value L. Otherwise, $\|B_k\|$ may become increasingly large (gradient inaccuracy promotes such a process), violating Assumption 3.2, and hence slows down or jeopardizes the convergence as in the case with accurate gradients. Limited-memory approaches [107] could also be tried but the Hessian of h_k is not our focus. After defining Δ_k and h_k, we solve the trust-region subproblem by the Toint-Steihaug truncated conjugate gradient method [30, Algorithm 7.5.1].

Our implementation sets $\eta_0 = 0$ in (6.3) for trial step acceptance. In addition, we let $\Delta_0 = 1$, $\gamma_1 = 1/2$, $\gamma_2 = 2$, $B_0 = I$, $r = 10^{-4}$, and $L = 10^4$. For the termination of the algorithm, we choose positive parameters $\hat{\varepsilon}$, $K_{\max}$ and $\Delta_{\min}$, and halt the iteration once

$$\|\hat{g}_k\| \;\leq\; \hat{\varepsilon}\|\hat{g}_0\| \quad \text{or} \quad k \;\geq\; K_{\max} \quad \text{or} \quad \Delta \;\leq\; \Delta_{\min},$$

outputting $x_{\text{out}} = x_k$. Note that $(1 - \zeta)\|\nabla f(x)\| \leq \|\hat{g}(x)\| \leq (1 + \zeta)\|\nabla f(x)\|$, where ζ is the relative gradient error. Therefore, given a desired optimality tolerance ε, we set

$$\hat{\varepsilon} \;=\; \varepsilon(1 + \zeta)^{-1}(1 - \zeta)$$

so that $\|\nabla f(x_{\text{out}})\| \leq \varepsilon\|\nabla f(x_0)\|$ is achieved if the algorithm is terminated due to $\|\hat{g}_k\| \leq \hat{\varepsilon}\|\hat{g}_0\|$ rather than reaching the maximal iteration number $K_{\max}$ or minimal trust-region radius $\Delta_{\min}$, which are respectively set to $500n$ and machine epsilon in the implementation.

6.7.3 Results of the experiment

We test the effectiveness of Algorithm 6.1 with optimality tolerance $\varepsilon = 10^{-6}$. The problem set $\mathcal{F}$ is obtained by perturbing the problems listed in Table 2 with dimension $n = 20$. For each of these problems and each $\zeta \in \{0, 1/50, 2/50, \ldots, 49/50, 1\}$, we contaminate the gradient according to (6.30), generating a perturbed problem with relative gradient error ζ. Given a ζ, the problem set $\mathcal{F}$ contains 10 independently perturbed instances for each problem in Table 2, summing up to 150 problems with relative gradient error ζ. For each $\eta_1 \in \{1/50, 2/50, \ldots, 49/50\}$, we tested the corresponding implementation of Algorithm 6.1 on these problems, recording its effectiveness defined in (6.29).

Figure 7 plots the effectiveness of Algorithm 6.1 when the algorithmic parameter η_1 was varying from 0 to 1. It displays individually the results obtained when the error magnitude ζ

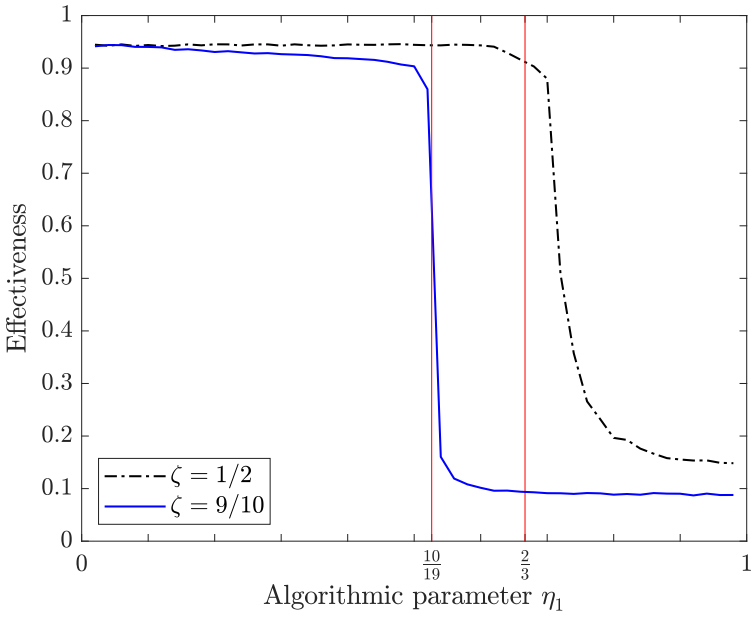


Figure 7: Effectiveness of Algorithm 6.1 plotted against algorithmic parameter η_1 (vertical lines: theoretical upper bounds of η_1 when relative gradient error $\zeta = 1/2$ or $9/10$).

was set to $1/2$ or $9/10$. According to Theorem 6.1, for Algorithm 6.1 to tolerate relative gradient error of magnitude ζ without losing convergence, it suffices to maintain $\zeta < \min\{\eta_1^{-1},\ 1\}$, or

$$\eta_1 \ < \ (1+\zeta)^{-1}. \tag{6.33}$$

Figure 7 indicates such a bound for $\zeta = 1/2$ and $\zeta = 9/10$ by vertical lines. As predicted by the theory, the algorithm behaved stably when the bound was respected, while suffering from a dramatic drop in its effectiveness once (6.33) was violated. In particular, when the error magnitude was high, namely $\zeta = 9/10$, we see a striking match between the theory and the numerical result. The match was less sharp when the error magnitude was lower, yet the rapid deterioration in the effectiveness was still detectable once (6.33) was violated.

Accompanying Figure 7, Figure 8 plots the effectiveness of Algorithm 6.1 when the error magnitude ζ was varying from 0 to 1. It displays the result for three implementations of the algorithm with $\eta_1 = 1/10$, $\eta_1 = 1/2$, and $\eta_1 = 2/3$ respectively. For each of these implementations, recall that the algorithm tolerates relative gradient error of magnitude ζ so long as

$$\zeta \ < \ \min\{\eta_1^{-1} - 1,\ 1\}. \tag{6.34}$$

This upper bound is $1/2$ for $\eta_1 = 2/3$, as indicated by the vertical line in Figure 8, and it becomes 1 when $\eta_1 = 1/2$ or $\eta_1 = 1/10$. Again, we observe a significant deterioration in the effectiveness of the algorithm once (6.34) was violated for $\eta_1 = 2/3$. For $\eta_1 = 1/2$, which is the critical value in (6.34), the situation was different, and the effectiveness started to deteriorate when the error magnitude was quite close to but smaller than 1. Such a distinction is not covered by our theory in this current work, which is an interesting motivation for future investigation. Yet the result with $\eta_1 = 1/2$ does not contradict Theorem 6.1, as the convergence of Algorithm 6.1 is ensured only when $\zeta < 1$ but not $\zeta = 1$, the latter of which can lead to divergence according to

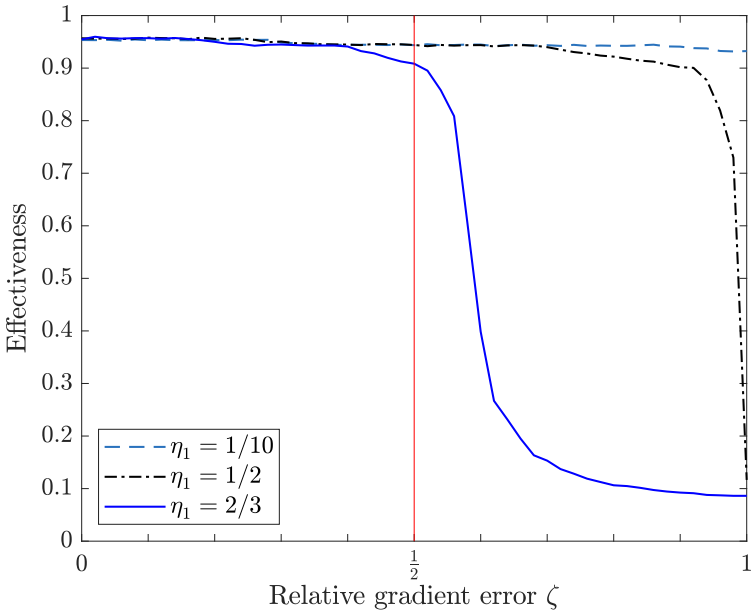


Figure 8: Effectiveness of Algorithm 6.1 plotted against relative gradient error (vertical line: theoretical upper bound of tolerable relative gradient error when $\eta_1 = 2/3$).

Example 6.1. We can at least say that the behavior of the algorithm reflected the bound (6.34) both when $\eta_1 = 2/3$ and $\eta_1 = 1/2$. For $\eta_1 = 1/10$, which is a value used in practice, the effectiveness of Algorithm 6.1 was always close to 1 without significant variation along with the error magnitude, matching Theorem 6.1 and confirming the robustness of practical trust-region methods with respect to gradient inaccuracy. Since the maximal iteration number was limited by $K_{\max} = 10^4$, it is not expected to observe the effectiveness of Algorithm 6.1 being exactly 1.

As a summary, the numerical behavior of Algorithm 6.1 in our experiment reflected quite well the bound $\{\eta_1^{-1} - 1,\ 1\}$ imposed by Theorem 6.1 on the relative gradient error $\|\hat{g}_k - g_k\|/\|g_k\|$. The effectiveness of Algorithm 6.1 deteriorated rapidly when such a bound was violated, manifesting the sharpness of the result.

6.7.4 Choosing η_1 in practice

Our theory in Subsections 6.3–6.5 and the experiment above confirm an intriguing and useful fact: the algorithmic parameter η_1 determines how much relative gradient error is tolerable for Algorithm 6.1, and a smaller value of η_1 implies higher tolerance to gradient inaccuracy. Such a conclusion can already be inferred from [15] and [30, Theorems 8.4.5 and 8.4.7]. In light of Theorems 6.1–6.3 and our numerical results, the decisive role of η_1 becomes crystal clear.

In practice, if the gradient evaluation is inaccurate with a known magnitude of relative error, then we can choose η_1 accordingly. To be precise, suppose that Algorithm 6.1 uses the gradient information provided by an inaccurate oracle $\hat{g} : \mathbb{R}^n \to \mathbb{R}^n$ that satisfies

$$\frac{\|\hat{g}(x) - \nabla f(x)\|}{\|\nabla f(x)\|} \leq \zeta < 1, \tag{6.35}$$

which is the case in our experiment, then it is sufficient to chose

$$\eta_1 \ < \ (1+\zeta)^{-1};$$

if $\hat{g}$ satisfies

$$\frac{\|\hat{g}(x) - \nabla f(x)\|}{\|\hat{g}(x)\|} \ \leq \ \zeta \ < \ 1,$$

then we can chose

$$\eta_1 \ < \ 1 - \zeta.$$

For instance, if the gradient evaluation has only one correct significant digit, then $\zeta \leq 1/2$ in (6.35), and hence setting $\eta_1 < 2/3$ (recall that $\eta_1 \leq 1/2$ is common in practice) can ensure Algorithm 6.1 to converge globally with *essentially the same* worst-case complexities as when the gradient evaluation is accurate. Even if the magnitude of relative error is unknown, Theorems 6.1–6.3 and our numerical experiment suggest that setting η_1 to a small value is favorable for avoiding performance deterioration caused by gradient inaccuracy.

This principle of taking a small η_1 is well documented in [61, Section 4.4], and our theory serves as a footnote for it. Nevertheless, not every widely used implementation of trust-region methods follows this principle. For example, the trust-region algorithms in GNU Octave [45] violate this principle and hence suffer when gradients are inaccurate. Appendix H details the problem with a simple remedy, illustrating how our theoretical study can make practical impacts.

As mentioned in Subsection 6.6, gradient descent with Armijo line search is a disguised trust-region method. Thus the comments made here are also applicable to it.

6.8 Remarks

The theorems in Subsection 6.3 and 6.4 (as well as Subsection G.2 in Appendix G) reveal that gradient inaccuracy influences the worst-case iteration complexity bounds of Algorithm 6.1 in a fairly benign way, provided that the inaccuracy does not exceed certain thresholds. Such an inaccuracy does not affect the *orders* of the bounds in terms of criticality measures, and the *multiplicative constants* in the bounds are enlarged modestly. Thus trust-region methods as described in Algorithm 6.1 are quite *robust with respect to gradient inaccuracy*. While such robustness has been known for a long time [97, 138, 15, 17, 75, 81], Theorems 6.1 and 6.2 refine our knowledge by quantifying the complexity bounds.

Moreover, recalling the discussions in Subsection 1.2, we should note that an inaccurate gradient may be much easier to obtain than the accurate one, or much cheaper to transmit in distributed computing environments. Since the iteration complexities of Algorithm 6.1 are fairly *stable* with respect to gradient inaccuracy, we may expedite the resolution of optimization problems by *deliberately* exploiting inaccurate gradient evaluations. Such a strategy would lose its theoretical basis if we did not establish the stability of the iteration complexities.

7 Discussions and conclusions

We developed a new and versatile space transformation framework for unconstrained optimization. At each iteration, the framework transmits information about the objective function to an auxiliary space where a step is obtained according to the transmitted information, and then updates the iterate by transferring this step back to the original space where the objective function

is initially posed. To obtain an algorithm that is convergent regardless of the starting point, we chose the trust-region globalization technique. We refrained from stating similar results with the Levenberg-Marquardt approach [88, 94, 109, 77], where d_k would be defined by

$$d_k \;\approx\; \operatorname{argmin}\left\{h_k(d) + \frac{\sigma_k}{2}\|d\|^2 : d \in \mathbb{R}^{N_k}\right\}$$

instead of (2), the update of trust-region radius being replaced by adjustment of σ_k. With such a globalization strategy, it is indeed possible to prove that our transformation framework still achieves global convergence as well as the $\mathcal{O}(\epsilon^{-2})$, $\mathcal{O}(\epsilon^{-1})$ and $\mathcal{O}(\log(\epsilon^{-1}))$ complexity bounds. For the cubic or high-order regularization approaches [70, 105, 18, 19, 7, 36, 21], the global convergence is provable, yet the improved complexity bounds such as the $\mathcal{O}(\epsilon^{-3/2})$ one for nonconvex problems in [19] would involve higher-order conditions in addition to $\nabla h_k(0) = R_k \nabla f(x_k)$ and hence necessitate the transformation of higher-order information. Extending our framework to such higher-order cases requires further investigation. Within the trust-region realm, [35] devises a strategy that also renders an $\mathcal{O}(\epsilon^{-3/2})$ complexity bound in the nonconvex case. It will be interesting to study how our framework can integrate such a strategy. Refined complexity analysis as proposed by [34] can also be considered within our framework.

The forward/backward transformations R_k and T_k in the current framework are linear. If R_k and T_k are nonlinear in Algorithm 3.1, then, as long as they are sufficiently smooth, one can still establish a lower bound for the reduction ratio ρ_k as in Lemma 3.1, but R_k and T_k will have to be replaced by their Fréchet derivatives, which are linear transformations. This is possible because such a lower bound only needs to hold when Δ_k is sufficiently small, in which case R_k and T_k can be well approximated with linear transformations defined by their derivatives. Once this bound is established, following the roadmap set in Section 3, the theory on global convergence and worst-case complexities can mostly be reproduced. This is why we decided to focus on linear transformations in our analysis. Nevertheless, nonlinear transformations can be useful in practice, because they do occur naturally, for example, in Space Mapping methods [4, 5, 144] that will be commented below. One can also generalize our framework to the case where the decision variable space $\mathcal{X}$ and the auxiliary space $\mathcal{Y}_k$ are both Banach spaces [143]. In such a context, as mentioned in Section 2, the forward transformation R_k should be a mapping between the dual spaces $\mathcal{X}'$ and $\mathcal{Y}_k'$. To extend our theory to such a much more general setting, we expect that more assumptions would be needed on the regularities of the spaces and the transformations.

We specialized the transformation framework into a space decomposition framework that enabled us to extend the parallel Schwarz method for PDEs to unconstrained smooth yet possibly nonconvex optimization. It would however be important, as it happens for domain decomposition in PDEs, to adapt the many components of our framework to specific applications. For instance, the decomposition should be designed to exploit the coupling pattern of variables and also balance the workload among subspaces for the sake of parallel performance. It will also be reasonable to consider updating the subspace trust-region radii separately according to individual reduction ratios defined for each subspace, as in [30, Section 10.2] and [28, 27, 26]. The best coarse spaces are known to be application dependent in PDEs, and we expect the same will happen for optimization. Exploiting coarse models or low-fidelity yet economical surrogates has been the central idea of Space Mapping [4, 5, 144] and surrogate-based optimization methods [69, 111], with a remarkable success achieved in engineering. It is thus worth investigating how such models and surrogates can guide us to design coarse spaces and coarse space models.

In numerical PDEs, domain decomposition methods are mainly used as *preconditioners* rather than iterative solvers [11, 14, 147]. As an initial attempt, we only applied Additive

Schwarz type methods and coarse space correction as iterative solvers. To use them as preconditioners in the context of nonlinear optimization, one may adopt the methodology of ASPIN [13], for instance. This will lead us to the investigation of *nonlinear preconditioning* [40, 54] within our space decomposition framework, which is another fascinating topic.

Our space decomposition method is synchronous in the sense that the reconstruction step waits for the optimization in all subspaces to be terminated. This would become a drawback when the computational load is much imbalanced among the processors in charge of the subspace minimization. Therefore, it is natural to consider an asynchronous extension of this method where each subspace uses the latest information from other subspaces to build its subproblem and conducts the optimization without waiting for each other (see [1], [149, Section 4] and [112] for examples), which will be theoretically interesting with significant practical impacts.

As an introduction to frameworks and their fundamental theory, we did not go into details about applications but only tried academic test problems in our experiments. Many real-world problems naturally enjoy structures that are favorable to space decomposition, such as the partial separability in Networks Optimization [140], the (overlapping) group structure in (Overlapping) Group Lasso [96, 82], and the layer structure in Deep Neural Networks [76]. To apply our framework to such problems, extensions are needed, for instance, to accommodate constraints and to deal with nonsmooth problems, which are possible directions for future research.

As mentioned in Subsection 1.2, optimization based on inaccurate function/gradient information is drawing considerable research attention, and there exist already an abundant amount of investigations along this direction. In particular, [3, 67, 23, 145] also study the behavior of trust-region methods when the function/gradient evaluations are inaccurate. They cover two important aspects that were not discussed here, namely inaccuracy in function values and inaccuracy with randomness. However, these works all measure inaccuracy as *absolute* errors that can be controlled actively and diminished to zero at least with a certain probability, which is similar to what we cover in Appendix G.2 but in probabilistic fashions. None of them measures inaccuracy in terms of *relative* errors as we did, yet we note that function values or gradients with bounded relative errors occur naturally if they are provided by solving lower-level subproblems with finite computing resources and finite precision arithmetic [59, 78]. Thus it will be interesting to extend our work to the situation where function values and/or gradients are with *relative errors* that are only bounded in certain *probabilistic* senses.

The probabilistic situation suggests that we should extend our space transformation framework by incorporating random transformations. In this current work, all the requirements we impose on the transformations are assumed to hold at each iteration. It will be interesting to relax the requirements and investigate, following [3, 66, 122, 67], how the space transformation framework will behave if the transformations are random and the requirements hold only probabilistically. Besides being mathematically fascinating, such an investigation would have several implications in practice. For example, it could enable us to understand the *fault-tolerance* behavior of the space decomposition framework if the subproblem solver in each subspace fails with a certain probability to deliver the subspace step. It will also be highly intriguing to study stochastic gradient descent (SGD) [123, 10, 60] methods under such a random space transformation framework. Besides, introducing randomness in the transformations might provide a new way of avoiding convergence to saddle points as in [58, 84].

With all these perspectives, the paradigm of optimization by space transformation opens an exciting and broad research avenue to explore.

Appendix

A Proof of inequality (3.18)

Proof. By considering $u = -d_k/\|d_k\|$, $v = \mathbf{t}_k/\|\mathbf{t}_k\|$, and $w = \mathbf{r}_k/\|\mathbf{r}_k\|$, it suffices to prove that

$$u^\mathsf{T} v \;\geq\; (u^\mathsf{T} w)(v^\mathsf{T} w) - \sqrt{[1-(u^\mathsf{T} w)^2][1-(v^\mathsf{T} w)^2]}. \tag{A.1}$$

Since $\|w\| = 1$, it is easy to see that $I - ww^\mathsf{T}$ is idempotent. Hence

$$u^\mathsf{T} v - (u^\mathsf{T} w)(v^\mathsf{T} w) \;=\; u^\mathsf{T}(I-ww^\mathsf{T})v \;=\; u^\mathsf{T}(I-ww^\mathsf{T})^2 v \;\geq\; -\|(I-ww^\mathsf{T})u\|\|(I-ww^\mathsf{T})v\|. \tag{A.2}$$

Noting that $\|(I-ww^\mathsf{T})u\| = \sqrt{u^\mathsf{T}(I-ww^\mathsf{T})^2 u} = \sqrt{u^\mathsf{T}(I-ww^\mathsf{T})u} = \sqrt{1-(u^\mathsf{T} w)^2}$, and similarly, $\|(I-ww^\mathsf{T})v\| = \sqrt{1-(v^\mathsf{T} w)^2}$, we obtain (A.1) from (A.2). □

B Proof of Proposition 3.1

Proof. without loss of generality that $\Delta_k \leq \|\nabla h_k(0)\|/L_h$ (otherwise, the desired conclusion is trivial). Then $\bar{d}_k = -\Delta_k \nabla h_k(0)/\|\nabla h_k(0)\|$ due to (3.7). By Taylor expansion,

$$h_k(0) - h_k(\bar{d}_k) \;\geq\; -(\bar{d}_k)^\mathsf{T}\nabla h_k(0) - \frac{L_h}{2}\|\bar{d}_k\|^2 \;=\; \Delta_k\|\nabla h_k(0)\| - \frac{L_h}{2}[\Delta_k]^2, \tag{B.1}$$

$$h_k(0) - h_k(d_k) \;\leq\; -d_k^\mathsf{T}\nabla h_k(0) + \frac{L_h}{2}\|d_k\|^2 \;\leq\; -d_k^\mathsf{T}\nabla h_k(0) + \frac{L_h}{2}[\Delta_k]^2. \tag{B.2}$$

From (B.1)–(B.2) and $h_k(d_k) \leq h_k(\bar{d}_k)$, we obtain $-d_k^\mathsf{T}\nabla h_k(0) \geq \Delta_k\|\nabla h_k(0)\| - L_h[\Delta_k]^2 \geq 0$. Hence

$$\cos\phi_k \;=\; \frac{-d_k^\mathsf{T}\nabla h_k(0)}{\|d_k\|\|\nabla h_k(0)\|} \;\geq\; \frac{\Delta_k\|\nabla h_k(0)\| - L_h[\Delta_k]^2}{\Delta_k\|\nabla h_k(0)\|} \;=\; 1 - \frac{L_h\Delta_k}{\|\nabla h_k(0)\|} \;\geq\; \sqrt{1 - \frac{2L_h\Delta_k}{\|\nabla h_k(0)\|}},$$

and consequently $\sin\phi_k = \sqrt{1-\cos^2\phi_k} \leq \sqrt{2L_h\Delta_k/\|\nabla h_k(0)\|}$. □

C Proof of Proposition 3.4

Proof. Let $a = \|\nabla h_k(0)\|$ and $b = \min\{\Delta_k, \|\nabla h_k(0)\|/L_h\}$. By Assumption 3.3 and Taylor expansion,

$$\frac{\alpha}{2}ab \;\leq\; -d_k^\mathsf{T}\nabla h_k(0) + \frac{L_h}{2}\|d_k\|^2 \;\leq\; \|d_k\|a + \frac{L_h}{2}\|d_k\|^2.$$

Hence

$$\frac{L_h}{2}\|d_k\|^2 + \|d_k\|a - \frac{\alpha}{2}ab \;\geq\; 0.$$

Regarding this as a quadratic inequality about $\|d_k\|$, and noting that $L_h ab \leq a^2$, we obtain

$$\|d_k\| \;\geq\; \frac{-a + \sqrt{a^2 + \alpha L_h ab}}{L_h} \;=\; \frac{\alpha ab}{a + \sqrt{a^2 + \alpha L_h ab}} \;\geq\; \frac{\alpha ab}{a + \sqrt{a^2 + \alpha a^2}} \;=\; \frac{\alpha b}{1 + \sqrt{1+\alpha}}.$$

We complete the proof by noting that $1 + \sqrt{1+\alpha} \leq 3$, as $\alpha \in (0, 1]$. □

D A lemma on the convergence rate of real sequences

The following lemma facilitates our complexity analysis under convexity. It is independent of our algorithmic discussion. Its proof technique has been widely used in the complexity analysis of optimization algorithms under convexity (for instance, see the proof of Theorem 2.1.14 in [102]).

Lemma D.1. *Let k be a positive integer, and $\{a_i\}$ be a real sequence such that*

$$a_i \;\geq\; r\sum_{j=i}^{k} a_j^2 \quad \textit{for each} \quad i \in \{1,2,\dots,k\}, \tag{D.1}$$

where r is a positive constant. Then

$$\min_{1\leq i\leq k} a_i \;\leq\; \frac{2}{rk}. \tag{D.2}$$

Proof. If $k=1$, then (D.1) reduces to $a_1 \geq r a_1^2$, implying (D.2). If $a_k = 0$, then (D.2) is trivial. So we assume $k \geq 2$ and $a_k > 0$. Let $A_i = \sum_{j=i}^{k} a_j^2$ for $i \in \{1,2,\dots,k\}$. Then $A_1 \geq \cdots \geq A_k > 0$. By (D.1),

$$A_i - A_{i+1} \;=\; a_i^2 \;\geq\; r^2 A_i^2 \quad \text{for each} \quad i \in \{1,2,\dots,k-1\}.$$

Therefore,

$$\frac{1}{A_{\lceil k/2\rceil}} \;=\; \frac{1}{A_1} + \sum_{i=1}^{\lceil k/2\rceil-1} \frac{A_i - A_{i+1}}{A_i A_{i+1}} \;\geq\; \frac{A_1 - A_2}{A_1^2} + \sum_{i=1}^{\lceil k/2\rceil-1} \frac{A_i - A_{i+1}}{A_i^2} \;\geq\; r^2\lceil k/2\rceil \;\geq\; \frac{r^2 k}{2}.$$

This leads to the following estimation and hence justifies (D.2):

$$\frac{2}{r^2 k} \;\geq\; A_{\lceil k/2\rceil} \;=\; \sum_{i=\lceil k/2\rceil}^{k} a_i^2 \;\geq\; (k - \lceil k/2\rceil + 1) \min_{\lceil k/2\rceil \leq i\leq k} a_i^2 \;\geq\; \frac{k}{2} \min_{1\leq i\leq k} a_i^2.$$

□

E Proofs of Propositions 5.1–5.3

Propositions 5.1 and 5.2 connect $\{h_k^i\}$ with $\{h_k\}$ and $\{d_k^i\}$ with $\{d_k\}$. Here are their proofs.

Proof of Proposition 5.1. The differentiability of h_k is obvious. We will verify the Lipschitz continuity of ∇h_k. To this end, let x and y be vectors in $\mathbb{R}^N$ and decompose them as

$$x \;=\; \left[x^1\,;\, x^2\,;\,\cdots\,;\, x^m\right] \quad \text{and} \quad y \;=\; \left[y^1\,;\, y^2\,;\,\cdots\,;\, y^m\right] \quad \text{with} \quad x^i,\, y^i \in \mathbb{R}^{N^i}.$$

Then due to the separability of h_k and the Lipschitz continuity of ∇h_k^i $(i = 1,2,\dots,m)$,

$$\|\nabla h_k(x) - \nabla h_k(y)\|^2 \;=\; \sum_{i=1}^{m} \left\|\nabla h_k^i(x^i) - \nabla h_k^i(y^i)\right\|^2 \;\leq\; \sum_{i=1}^{m} L_h^2 \|x^i - y^i\|^2 \;=\; L_h^2 \|x - y\|^2.$$

Therefore, ∇h_k is L_h-Lipschitz continuous. □

Proof of Proposition 5.2. By Assumption 5.2 and the definition (5.26) of h_k, we have

$$h_k(0) - h_k(d_k) \;=\; \sum_{i=1}^{m} \left[h_k^i(0) - h_k^i(d_k^i)\right] \;\geq\; \sum_{i=1}^{m} \left[h_k^i(0) - h_k^i(\bar{d}_k^i)\right] \;=\; h_k(0) - h_k(\hat{d}_k),$$

where

$$\hat{d}_k \;=\; \left[\bar{d}_k^1\,;\, \bar{d}_k^2\,;\,\cdots\,;\, \bar{d}_k^m\right]. \tag{E.1}$$

We will show that $\hat{d}_k$ indeed equals $\bar{d}_k$ defined in (3.7). Consequently, inequality (5.45) is true according to the reduction estimation (3.8), and (5.46) holds according to Proposition 3.1. With (5.26), we have

$$\|\nabla h_k(0)\| \;=\; \Big[\sum_{i=1}^{m} \big\|\nabla h_k^i(0)\big\|^2\Big]^{\frac{1}{2}}.$$

Hence, according to (5.41), we can rewrite (5.44) as

$$\bar{d}_k^i \;=\; -\min\left\{\frac{\Delta_k}{\|\nabla h_k(0)\|},\;\frac{1}{L_h}\right\}\nabla h_k^i(0). \tag{E.2}$$

Combining (5.27) and (E.2), we can formulate $\hat{d}_k$ defined in (E.1) as

$$\hat{d}_k \;=\; -\min\left\{\frac{\Delta_k}{\|\nabla h_k(0)\|},\;\frac{1}{L_h}\right\}\nabla h_k(0) \;=\; \bar{d}_k,$$

with $\bar{d}_k$ being defined by (3.7). This completes the proof. □

Now we consider Proposition 5.3, which specifies some bounds for the quantities $\boldsymbol{\tau}$, $\boldsymbol{\kappa}$, and $\boldsymbol{\lambda}$ defined in (5.47). For convenience, we first prove the following lemma on the matrices $\{U^i\}_{i=1}^m$, $\{\tilde{U}^i\}_{i=1}^m$, and $\{W^i\}_{i=1}^m$ defined in Subsection 5.5.1. We will use

$$\underset{1\le j\le n}{\mathrm{Diag}}\,\big(a_j\big)$$

to denote the $n\times n$ diagonal matrix whose diagonal entries are $a_1, a_2, \ldots, a_n$.

Lemma E.1. *For $\{U^i\}_{i=1}^m$, $\{\tilde{U}^i\}_{i=1}^m$, and $\{W^i\}_{i=1}^m$ defined in (5.31), (5.32), and (5.33), it holds that*

$$\sum_{i=1}^{m} U^i[U^i]^{\mathsf{T}} \;=\; \underset{1\le j\le n}{\mathrm{Diag}}\,\big(\theta_j\big), \tag{E.3}$$

$$\sum_{i=1}^{m} \tilde{U}^i[\tilde{U}^i]^{\mathsf{T}} \;=\; \sum_{i=1}^{m} W^i[U^i]^{\mathsf{T}} \;=\; I, \tag{E.4}$$

$$\underset{1\le j\le n}{\mathrm{Diag}}\,\big(\theta_j^{-1}\big) \;\le\; \sum_{i=1}^{m} W^i[W^i]^{\mathsf{T}} \;\le\; I. \tag{E.5}$$

Note that inequality (E.5) is entry-wise. It implies that $\sum_{i=1}^m W^i[W^i]^{\mathsf{T}}$ is diagonal and

$$\underset{1\le j\le n}{\mathrm{Diag}}\,\big(\theta_j^{-1}\big) \;\preceq\; \sum_{i=1}^{m} W^i[W^i]^{\mathsf{T}} \;\preceq\; I,$$

where $\preceq$ signifies the Löwner partial order: $A\preceq B$ if and only if $B-A$ is positive semidefinite.

Proof. Only the facts concerning $\{W^i\}$ need proofs. The others can be verified by direct calculations. We first prove $\sum_{i=1}^m W^i[U^i]^{\mathsf{T}} = I$. According to (5.31) and (5.33),

$$W^i[U^i]^{\mathsf{T}} \;=\; \sum_{j\in X^i} w_j^i e_j e_j^{\mathsf{T}} \;=\; \sum_{j=1}^{n} w_j^i\,\mathbb{1}\big(j\in X^i\big)e_j e_j^{\mathsf{T}}.$$

Thus

$$\sum_{i=1}^{m} W^i[U^i]^{\mathsf{T}} \;=\; \sum_{i=1}^{m}\sum_{j=1}^{n} w_j^i\,\mathbb{1}\big(j\in X^i\big)e_j e_j^{\mathsf{T}} \;=\; \sum_{j=1}^{n}\sum_{i=1}^{m} w_j^i\,\mathbb{1}\big(j\in X^i\big)e_j e_j^{\mathsf{T}} \;=\; \sum_{j=1}^{n} e_j e_j^{\mathsf{T}} \;=\; I,$$

where the penultimate equality is because $\sum_{i=1}^m w_j^i \mathbb{1}\big(j \in X^i\big) = 1$ as imposed by (5.34).

Now we prove (E.5). Since $0 \le W^i \le U^i$ for each $i \in \{1, 2, \dots, m\}$, we have

$$0 \;\le\; \sum_{i=1}^m W^i [W^i]^\mathsf{T} \;\le\; \sum_{i=1}^m W^i [U^i]^\mathsf{T} \;=\; I,$$

implying that $\sum_{i=1}^m W^i [W^i]^\mathsf{T}$ is a diagonal matrix. It remains to check that the (j,j) entry of this matrix is at least θ_j^{-1}. We first observe that, for any real number t,

$$0 \;\preceq\; \sum_{i=1}^m \left[tU^i - W^i\right] \left[tU^i - W^i\right]^\mathsf{T} = \sum_{i=1}^m \left[t^2 U^i [U^i]^\mathsf{T} - tU^i [W^i]^\mathsf{T} - tW^i [U^i]^\mathsf{T} + W^i [W^i]^\mathsf{T}\right].$$

According to (E.3) and the fact that both $\sum_{i=1}^m U^i [W^i]^\mathsf{T}$ and $\sum_{i=1}^m W^i [U^i]^\mathsf{T}$ equal I (they are the transposes of each other and the second one is already proved to be identity), the above inequality implies

$$0 \;\preceq\; t^2 \operatorname*{Diag}_{1 \le j \le n} \big(\theta_j\big) - 2tI + \sum_{i=1}^m W^i [W^i]^\mathsf{T}.$$

Setting $t = \theta_j^{-1}$ and checking the (j,j) entry of the right-hand side, we see that the (j,j) entry of $\sum_{i=1}^m W^i [W^i]^\mathsf{T}$ is at least θ_j^{-1}. □

Now we give the proof of Proposition 5.3.

Proof of Proposition 5.3.

1. AS. According to AS (5.35) and Lemma E.1, we have

$$T_k T_k^\mathsf{T} \;=\; \sum_{i=1}^m U^i [U^i]^\mathsf{T} \;=\; \operatorname*{Diag}_{1 \le j \le n} (\theta_j), \tag{E.6}$$

$$R_k^\mathsf{T} R_k \;=\; T_k T_k^\mathsf{T} \;=\; \operatorname*{Diag}_{1 \le j \le n} (\theta_j) \;\succeq\; I, \tag{E.7}$$

$$T_k R_k \;=\; R_k^\mathsf{T} R_k. \tag{E.8}$$

Invoking (E.6), (E.7) and (E.8) one by one, we obtain

$$\|T_k\| \;=\; \sqrt{\max_{1 \le j \le n} \theta_j} \;=\; \sqrt{\theta}, \quad \|R_k g\| \;\ge\; \|g\|, \quad \text{and} \quad g^\mathsf{T} T_k R_k g \;=\; \|R_k g\|^2,$$

implying $\boldsymbol{\tau} = \sqrt{\theta}$, $\boldsymbol{\kappa} \ge 1$, and $\boldsymbol{\lambda} = 1$.

2. WRAS. According to WRAS (5.37) and Lemma E.1, we have

$$T_k T_k^\mathsf{T} \;=\; \sum_{i=1}^m W^i [W^i]^\mathsf{T} \;\preceq\; I,$$

$$R_k^\mathsf{T} R_k \;=\; \sum_{i=1}^m U^i [U^i]^\mathsf{T} \;=\; \operatorname*{Diag}_{1 \le j \le n} (\theta_j) \;\succeq\; I,$$

$$T_k R_k \;=\; \sum_{i=1}^m W^i [U^i]^\mathsf{T} \;=\; I \;\succeq\; \theta^{-1} \operatorname*{Diag}_{1 \le j \le n} (\theta_j) \;=\; \theta^{-1} R_k^\mathsf{T} R_k.$$

Hence $\|T_k\| \le 1$, $\|R_k g\| \ge \|g\|$, and $g^\mathsf{T} T_k R_k g \ge \theta^{-1} \|R_k g\|^2$, implying $\boldsymbol{\tau} \le 1$, $\boldsymbol{\kappa} \ge 1$, and $\boldsymbol{\lambda} \ge \theta^{-1}$.

3. WASH. According to WASH (5.39) and Lemma E.1, we have

$$T_k T_k^\mathsf{T} \;=\; \sum_{i=1}^m U^i[U^i]^\mathsf{T} \;=\; \operatorname*{Diag}_{1\le j\le n}(\theta_j),$$

$$R_k^\mathsf{T} R_k \;=\; \sum_{i=1}^m W^i[W^i]^\mathsf{T} \;\succeq\; \operatorname*{Diag}_{1\le j\le n}(\theta_j^{-1}) \;\succeq\; \theta^{-1} I,$$

$$T_k R_k \;=\; \sum_{i=1}^m U^i[W^i]^\mathsf{T} \;=\; \left[\sum_{i=1}^m W^i[U^i]^\mathsf{T}\right]^\mathsf{T} \;=\; I \;\succeq\; \sum_{i=1}^m W^i[W^i]^\mathsf{T} \;=\; R_k^\mathsf{T} R_k.$$

Hence $\|T_k\| = \theta$, $\|R_k g\| \ge \sqrt{\theta^{-1}}\|g\|$, and $g^\mathsf{T} T_k R_k g \ge \|R_k g\|^2$, implying $\boldsymbol{\tau} = \sqrt{\theta}$, $\boldsymbol{\kappa} \ge \sqrt{\theta^{-1}}$, and $\boldsymbol{\lambda} \ge 1$.

4. RASH. According to RASH (5.40) and Lemma E.1, we have

$$T_k T_k^\mathsf{T} \;=\; \sum_{i=1}^m \tilde{U}^i[\tilde{U}^i]^\mathsf{T} \;=\; I, \quad R_k^\mathsf{T} R_k \;=\; T_k T_k^\mathsf{T} \;=\; I, \quad \text{and} \quad T_k R_k \;=\; R_k^\mathsf{T} T_k.$$

Hence $\|T_k\| = 1$, $\|R_k g\| = \|g\|$, and $g^\mathsf{T} T_k R_k g = \|R_k g\|^2$, implying $\boldsymbol{\tau} = \boldsymbol{\kappa} = \boldsymbol{\lambda} = 1$. □

F Proofs of Theorems 6.1–6.3

Theorems 6.1–6.3 cast the theory of Algorithm 3.1 to Algorithm 6.1 under three different conditions on the inaccurate gradient $\hat{g}_k$. To prove them, we only need to verify that R_k and T_k defined by (6.1)–(6.2) fulfill Assumptions 3.5–3.7 with appropriate values of τ, κ, and λ, and then invoke Theorems 3.1–3.4 and 3.7.

Indeed, since $T_k \equiv I$, Assumption 3.5 always holds trivially with $\tau = 1$. We only need to justify Assumptions 3.6–3.7. Since $R_k g_k = \hat{g}_k$ and $T_k^\mathsf{T} g_k = g_k$ under (6.1)–(6.2), it is equivalent to show that

$$\|\hat{g}_k\| \;\ge\; \kappa\|g_k\| \quad \text{and} \quad g_k^\mathsf{T}\hat{g}_k \;\ge\; \lambda\|\hat{g}_k\|^2$$

for appropriate values of κ and λ.

Proof of Theorem 6.1. Assumption 3.5 holds with $\tau = 1$. According to (6.4),

$$\|\hat{g}_k\| \;\ge\; \|g_k\| - \|\hat{g}_k - g_k\| \;\ge\; (1-\zeta)\|g_k\|,$$

which validates Assumption 3.6 with $\kappa = 1 - \zeta > 0$. Condition (6.4) also implies

$$\|\hat{g}_k\|^2 - 2g_k^\mathsf{T}\hat{g}_k + \|g_k\|^2 \;\le\; \zeta^2\|g_k\|^2 \quad \text{and} \quad \|\hat{g}_k\| \;\le\; \|g_k\| + \|\hat{g}_k - g_k\| \;\le\; (1+\zeta)\|g_k\|,$$

which give us

$$g_k^\mathsf{T}\hat{g}_k \;\ge\; \frac{\|\hat{g}_k\|^2}{2} + \frac{1-\zeta^2}{2}\|g_k\|^2 \;\ge\; \frac{\|\hat{g}_k\|^2}{2} + \frac{1-\zeta^2}{2}\cdot\frac{\|\hat{g}_k\|^2}{(1+\zeta)^2} \;=\; (1+\zeta)^{-1}\|\hat{g}_k\|^2,$$

justifying Assumption 3.7 with $\lambda = (1+\zeta)^{-1} > \eta_1$.

Assumptions 3.5–3.7 are verified. The proof is completed by invoking Theorems 3.1–3.4 and 3.7. □

Proof of Theorem 6.2. Assumption 3.5 holds with $\tau = 1$. According to (6.11),

$$\|g_k\| \;\le\; \|\hat{g}_k\| + \|\hat{g}_k - g_k\| \;\le\; (1+\zeta)\|\hat{g}_k\|, \tag{F.1}$$

which validates Assumption 3.6 with $\kappa = (1+\zeta)^{-1} > 0$. Condition (6.11) also implies that

$$(\hat{g}_k - g_k)^\mathsf{T}\hat{g}_k \;\le\; \|\hat{g}_k - g_k\|\|\hat{g}_k\| \;\le\; \zeta\|\hat{g}_k\|^2,$$

and hence $g_k^\mathsf{T}\hat{g}_k \ge (1-\zeta)\|\hat{g}_k\|^2$, justifying Assumption 3.7 with $\lambda = 1 - \zeta > \eta_1$.

Assumptions 3.5–3.7 are verified. The proof is completed by invoking Theorems 3.1–3.4 and 3.7. □

Proof of Theorem 6.3. Assumption 3.5 holds with $\tau = 1$. According to (6.19),

$$\|R_k g_k\| \;=\; \|\hat{g}_k\| \;\geq\; \left\|\frac{g_k}{2\eta_1}\right\| - \left\|\hat{g}_k - \frac{g_k}{2\eta_1}\right\| \;\geq\; \frac{1-\zeta}{2\eta_1}\|g_k\|,$$

which validates Assumption 3.6 with $\kappa = (1-\zeta)/(2\eta_1) > 0$. Condition (6.19) also implies

$$\|\hat{g}_k\|^2 - \frac{g_k^{\mathsf{T}}\hat{g}_k}{\eta_1} + \frac{\|g_k\|^2}{4\eta_1^2} \;\leq\; \frac{\zeta^2}{4\eta_1^2}\|g_k\|^2 \quad \text{and} \quad \|\hat{g}_k\| \;\leq\; \left\|\frac{g_k}{2\eta_1}\right\| + \left\|\hat{g}_k - \frac{g_k}{2\eta_1}\right\| \;\leq\; \frac{1+\zeta}{2\eta_1}\|g_k\|,$$

which gives us

$$g_k^{\mathsf{T}}\hat{g}_k \;\geq\; \eta_1\|\hat{g}_k\|^2 + \frac{1-\zeta^2}{4\eta_1}\|g_k\|^2 \;\geq\; \eta_1\|\hat{g}_k\|^2 + \frac{1-\zeta^2}{4\eta_1}\cdot\left(\frac{2\eta_1}{1+\zeta}\right)^2\|\hat{g}_k\|^2 \;=\; \frac{2\eta_1}{1+\zeta}\|\hat{g}_k\|^2,$$

justifying Assumption 3.7 with $\lambda = 2\eta_1/(1+\zeta) > \eta_1$.

Assumptions 3.5–3.7 are verified. The proof is completed by invoking Theorems 3.1–3.4 and 3.7. □

G Alternative assumptions on $\{d_k\}$, $\{R_k\}$, and $\{T_k\}$

In the investigation of our space transformation framework (*i.e.*, Algorithm 3.1), Assumptions 3.4 and 3.7 seem more particular than the other hypotheses. The first of them requires that the angle ϕ_k between the trust-region step d_k and the negative model gradient $-\nabla h_k(0)$ becomes small when the trust-region radius Δ_k is small (compared with $\|\nabla h_k(0)\|$), and the second demands that $g_k T_k R_k g_k/\|R_k g_k\|^2$ remains uniformly larger than the algorithmic parameter η_1.

This section will elaborate more on these assumptions and present alternatives to them. The alternative assumptions can render the same global convergence and worst-case complexity theory of Algorithm 3.1 as in Section 3, except for modifications on the lower bound μ of $\{\Delta_k/\|\tilde{g}_k\|\}$ in (3.30).

As an application of the theory under the alternative assumptions, we will investigate the behavior of trust-region methods (as described in Algorithm 6.1) when the gradient accuracy is controlled dynamically according to the size of the trust region.

G.1 The alternative assumptions and the resultant theory of Algorithm 3.1

To find alternatives for Assumptions 3.4 and 3.7, let us first examine what are their consequences that are critical to the convergence of Algorithm 3.1. For convenience, we denote

$$\vartheta_k = \frac{\Delta_k}{\|\nabla h_k(0)\|}.$$

What matters is the situation when ϑ_k tends to zero. Indeed, Assumption 3.7 ensures that

$$\liminf_{\vartheta_k \to 0} \frac{g_k^{\mathsf{T}} T_k R_k g_k}{\|R_k g_k\|^2} \;>\; \eta_1, \tag{G.1}$$

With the help of Assumptions 3.5 and 3.6, Assumption 3.7 also implies that

$$\limsup_{\vartheta_k \to 0} |\tan\psi_k| \;<\; \infty, \tag{G.2}$$

as we can see from (3.23). Meanwhile, we know from Assumption 3.4 that

$$\lim_{\vartheta_k \to 0} \tan\phi_k \;=\; 0. \tag{G.3}$$

Consequently,

$$\lim_{\vartheta_k \to 0} \tan\phi_k \tan\psi_k \;=\; 0. \tag{G.4}$$

According to the bound for ρ_k in Lemma 3.1, inequality (G.1) and equation (G.4) are the essentials that ensure Proposition 3.2, which says that ρ_k surpasses η_1 when $\Delta_k/\|\nabla h_k(0)\|$ is small. Any assumptions that guarantee (G.1) and (G.4) with certain rates can serve as alternatives to Assumptions 3.4 and 3.7 to establish an analogue of Proposition 3.2 and hence secure the global convergence and worst-case complexity of Algorithm 3.1. For (G.1), the following slight generalization of Assumptions 3.7 is sufficient.

Assumption G.1 (Alternative to Assumption 3.7)**.** *There exist constants $\lambda > \eta_1$, $\upsilon > 0$, and $q > 0$ such that*

$$\frac{g_k^{\mathsf{T}} T_k R_k g_k}{\|R_k g_k\|^2} \;\geq\; \lambda - \upsilon \left[\frac{\Delta_k}{\|\nabla h_k(0)\|}\right]^q \qquad \text{for each} \qquad k \geq 0.$$

Note that $\Delta_k/\|\nabla h_k(0)\|$ is indeed $\Delta_k/\|R_k g_k\|$ as $\nabla h_k(0) = R_k g_k$. Assumption G.1 displays it as $\Delta_k/\|\nabla h_k(0)\|$ to stress that it is the ratio between the trust-region radius and the model gradient length.

Now we seek an alternative to Assumption 3.4 that can render (G.4). Symmetric to (G.2)–(G.3), one can ensure (G.4) by

$$\begin{cases} \limsup\limits_{\vartheta_k \to 0} |\tan\phi_k| \;<\; \infty, & \text{(G.5)} \\ \lim\limits_{\vartheta \to 0} \tan\psi_k \;=\; 0. & \text{(G.6)} \end{cases}$$

Note that (G.5) holds under the conventional assumptions on h_k and d_k, namely Assumptions 3.2 and 3.3. Indeed, as shown in Subsection 3.2.2, the named assumptions lead to (3.10), which implies

$$\cos\phi_k \;\geq\; \frac{1}{2}\left(\alpha - \frac{L_h \Delta_k}{\|\nabla h_k(0)\|}\right) \;=\; \frac{1}{2}(\alpha - L_h \vartheta_k). \tag{G.7}$$

Hence

$$\limsup_{\vartheta_k \to 0} |\tan\phi_k| \;\leq\; \limsup_{\vartheta_k \to 0} |\sec\phi_k| \;\leq\; 2\alpha^{-1}.$$

Therefore, to guarantee (G.4), it suffices to ensure (G.6). Thus we propose the following assumption on R_k and T_k, more precisely, on the angle between $R_k g_k$ and $T_k^{\mathsf{T}} g_k$. It is the ψ_k-counterpart of Assumption 3.4.

Assumption G.2 (Alternative to Assumption 3.4)**.** *There exist constants $\beta \geq 0$ and $p > 0$ such that*

$$\sin\psi_k \;\leq\; \beta \left[\frac{\Delta_k}{\|\nabla h_k(0)\|}\right]^p \qquad \text{for each} \qquad k \geq 0,$$

where ψ_k is the angle between $R_k g_k$ and $T_k^{\mathsf{T}} g_k$ as defined in (3.12).

Assumption G.2 requires $R_k g_k$ and $T_k^{\mathsf{T}} g_k$ to be approximately in the same direction when $\Delta_k/\|\nabla h_k(0)\|$ is small. In general, this is not necessarily the case, and that is why Assumption 3.4 was proposed. However, Assumption G.2 may hold true if one actively controls the discrepancy between $R_k g_k$ and $T_k^{\mathsf{T}} g_k$ in course of the algorithm. We will see such an example in Section G.2 when studying trust-region methods that use inaccurate gradients whose accuracy is adaptively chosen according to the size of the trust region.

If Assumption G.2 is true, then the convergence (G.4) holds with rate $\mathcal{O}([\Delta_k/\|\nabla h_k(0)\|]^p)$, and Assumption 3.4 is not needed any more. In Subsection 3.2.5, we mentioned the possibility of waiving Assumption 3.4 when R_k and T_k are always the transposes of each other. Assumption G.2 can be interpreted as a generalization of that comment.

Theorem G.1 presents the theory of Algorithm 3.1 when Assumption 3.7 is replaced by its generalized version Assumption G.1 and the place of Assumption 3.4 is taken by its ψ_k-counterpart Assumption G.2. Its proof is sketched briefly. Similar to Theorem 3.5, if Algorithm 3.1 updates Δ_k following the more general rule (3.65) instead of (3.3), then Theorem G.1 holds after modifying μ defined below in (G.8).

Theorem G.1. *Under Assumptions 3.1, 3.2, 3.3, 3.5, 3.6, G.1, and G.2, Algorithm 3.1 enjoys the global convergence elaborated in Theorems 3.1 and 3.7, and it possesses the worst-case complexity bounds quantified in Theorems 3.2–3.4, except that the constant μ defined in (3.30) should be replaced with*

$$\mu \;=\; \min\left\{ \frac{\Delta_0}{\|g_0\|},\;\; \gamma_0\kappa\left[\frac{\alpha(\lambda-\eta_1)}{5\beta\lambda}\right]^{\frac{1}{p}},\;\; \gamma_0\kappa\left(\frac{\lambda-\eta_1}{5\upsilon}\right)^{\frac{1}{q}},\;\; \frac{\alpha\gamma_0\kappa(\lambda-\eta_1)}{5(\lambda L_h+\tau^2 L_f)} \right\}. \tag{G.8}$$

Proof. We only need to prove that $\rho_k > \eta_1$ when

$$\frac{\Delta_k}{\|\nabla h_k(0)\|} \;\le\; \min\left\{ \left[\frac{\alpha(\lambda-\eta_1)}{5\beta\lambda}\right]^{\frac{1}{p}},\;\; \left(\frac{\lambda-\eta_1}{5\upsilon}\right)^{\frac{1}{q}},\;\; \frac{\alpha(\lambda-\eta_1)}{5(\lambda L_h+\tau^2 L_f)} \right\}, \tag{G.9}$$

which is an analogue of Proposition 3.2. If this is proved, then one can verify that Proposition 3.3 holds with the new μ, and Theorems 3.1–3.4 and 3.7 can be established as in Section 3.

Essentially the same arguments as in the proof of Proposition 3.2 can prove that (G.9) ensures $\rho_k > \eta_1$, except that the roles of ϕ_k and ψ_k are exchanged. We claim that

$$0 \;\le\; \tan\phi_k \tan\psi_k \;<\; \frac{5(\lambda-\eta_1)}{4\sqrt{6}\lambda}. \tag{G.10}$$

Let us first show how to derive $\rho_k > \eta_1$ from (G.9)–(G.10), and then verify (G.10) in the end.

Assumption G.1 and the second term in (G.9) imply that $g_k^{\mathsf{T}} T_k R_k g_k \;\ge\; 0$, and hence $0 \le \psi_k \le \pi/2$. Meanwhile, the third term in (G.9) ensures $\Delta_k \le \alpha\|\nabla h_k(0)\|/L_h$, which tells us $0 \le \phi_k \le \pi/2$ according to (3.9). Additionally, $\tan\phi_k \tan\psi_k < 1$ by (G.10). Therefore,

$$\phi_k + \psi_k \;<\; \frac{\pi}{2}.$$

Hence, recalling that $\Delta_k \le \alpha\|\nabla h_k(0)\|/L_h$, we can invoke Lemma 3.1 to obtain

$$\rho_k \;\ge\; \frac{g_k^{\mathsf{T}} T_k R_k g_k}{\|R_k g_k\|^2}\left(1-\tan\phi_k\tan\psi_k-\frac{L_h\Delta_k}{\alpha\|R_k g_k\|}\right)-\frac{L_f\|T_k\|^2\Delta_k}{\alpha\|R_k g_k\|}. \tag{G.11}$$

By (G.10) and the third term in (G.9), and noting that $\nabla h_k(0) = R_k g_k$, we also have

$$1-\tan\phi_k\psi_k-\frac{L_h\Delta_k}{\alpha\|R_k g_k\|} \;>\; 1-\frac{5(\lambda-\eta_1)}{4\sqrt{6}\lambda}-\frac{L_h}{\alpha}\cdot\frac{\alpha(\lambda-\eta_1)}{5(\lambda L_h+\tau^2 L_f)} \;>\; 1-\frac{5}{4\sqrt{6}}-\frac{1}{5} \;>\; 0.$$

Thus, we can push (G.11) forward by Assumption G.1 and Assumption 3.5, obtaining

$$\begin{aligned}
\rho_k \;&\ge\; \left(\lambda-\upsilon\left[\frac{\Delta_k}{\|\nabla h_k(0)\|}\right]^q\right)\left(1-\tan\phi_k\tan\psi_k-\frac{L_h\Delta_k}{\alpha\|\nabla h_k(0)\|}\right)-\frac{\tau^2 L_f\Delta_k}{\alpha\|\nabla h_k(0)\|}\\
&=\; \lambda-\upsilon\left[\frac{\Delta_k}{\|\nabla h_k(0)\|}\right]^q-\left(\lambda-\upsilon\left[\frac{\Delta_k}{\|\nabla h_k(0)\|}\right]^q\right)\left(\tan\phi_k\tan\psi_k+\frac{L_h\Delta_k}{\alpha\|\nabla h_k(0)\|}\right)-\frac{\tau^2 L_f\Delta_k}{\alpha\|\nabla h_k(0)\|}\\
&\ge\; \lambda-\upsilon\left[\frac{\Delta_k}{\|\nabla h_k(0)\|}\right]^q-\lambda\left(\tan\phi_k\tan\psi_k+\frac{L_h\Delta_k}{\alpha\|\nabla h_k(0)\|}\right)-\frac{\tau^2 L_f\Delta_k}{\alpha\|\nabla h_k(0)\|}\\
&=\; \lambda-\upsilon\left[\frac{\Delta_k}{\|\nabla h_k(0)\|}\right]^q-\lambda\tan\phi_k\tan\psi_k-\frac{(\lambda L_h+\tau^2 L_f)\Delta_k}{\alpha\|\nabla h_k(0)\|}.
\end{aligned}$$

Plugging (G.10) and the last two terms in (G.9) into the last line above, we arrive at

$$\begin{aligned}
\rho_k \;&>\; \lambda-\upsilon\cdot\frac{\lambda-\eta_1}{5\upsilon}-\lambda\cdot\frac{5(\lambda-\eta_1)}{4\sqrt{6}\lambda}-\frac{(\lambda L_h+\tau^2 L_f)}{\alpha}\cdot\frac{\alpha(\lambda-\eta_1)}{5(\lambda L_h+\tau^2 L_f)}\\
&>\; \lambda-\frac{\lambda-\eta_1}{5}-\frac{5(\lambda-\eta_1)}{4\sqrt{6}\lambda}-\frac{\lambda-\eta_1}{5} \;>\; \eta_1.
\end{aligned}$$

Now we verify (G.10). By Assumption G.2 and the first term in (G.9),

$$\sin\psi_k \;\le\; \beta\cdot\frac{\alpha(\lambda-\eta_1)}{5\beta\lambda} \;=\; \frac{\alpha(\lambda-\eta_1)}{5\lambda}.$$

Thus $\sin\psi_k < 1/5$. Recalling that $0 \le \psi_k \le \pi/2$, we have $\cos\psi_k > \sqrt{1-(1/5)^2} = 2\sqrt{6}/5$, and hence

$$0 \;\le\; \tan\psi_k \;<\; \frac{\alpha(\lambda-\eta_1)}{2\sqrt{6}\lambda}. \tag{G.12}$$

The third term in (G.9) implies that $\Delta_k/\|\nabla h_k(0)\| \le \alpha/(5L_h)$. Thus we can obtain from (G.7) that

$$0 \;\le\; \tan\phi_k \;\le\; \sec\phi_k \;\le\; \frac{5}{2\alpha}. \tag{G.13}$$

With (G.12)–(G.13), inequality (G.10) is justified. □

G.2 Inaccurate gradients with dynamic accuracy

Let us consider Algorithm 6.1 again. In practice, if one has control over the accuracy of $\hat{g}_k$, then it is natural to adjust $\|\hat{g}_k - g_k\|$ adaptively according to the progress of the computation. In particular, large values of $\|\hat{g}_k - g_k\|$ should be tolerable when algorithms takes long steps, which is the case when Δ_k is large in 6.1. Otherwise, $\|\hat{g}_k - g_k\|$ has to be small. In this spirit, [75, inequality (4.4)] proposes condition

$$\|\hat{g}_k - g_k\| \;\le\; \xi\min\{\|\hat{g}_k\|,\; \Delta_k\} \quad \text{for each} \quad k \ge 0. \tag{G.14}$$

on the inaccurate gradients $\{\hat{g}_k\}$ used in trust-region methods to guarantee the global convergence, where ξ is a positive constant. Indeed, [75] extends its investigation to the more general context of trust-region SQP algorithms, though worst-case complexity was not discussed.

Using Theorem G.2, we can obtain the global convergence and worst-case complexities of Algorithm 6.1 under a condition that generalizes (G.14) slightly.

Theorem G.2. *Consider Algorithm 6.1 with $\eta_1 < 1$. Assume there exist constants ζ and ξ such that*

$$\|\hat{g}_k - g_k\| \;\le\; \min\{\zeta\|\hat{g}_k\|,\; \xi\Delta_k\} \quad \textit{for each} \quad k \ge 0. \tag{G.15}$$

Then under Assumptions 3.1–3.3, Algorithm 6.1 enjoys the global convergence elaborated in Theorems 3.1 and 3.7, and it possesses the worst-case complexity bounds quantified in Theorems 3.2–3.4, with κ and μ taking the particular values

$$\kappa(\zeta) \;=\; (1+\zeta)^{-1} \quad \textit{and} \quad \mu(\zeta,\xi) \;=\; \min\left\{\frac{\Delta_0}{\|g_0\|},\; \frac{\alpha\gamma_0\kappa(\zeta)\,(1-\eta_1)}{5\xi},\; \frac{\alpha\gamma_0\kappa(\zeta)\,(1-\eta_1)}{5(L_h+L_f)}\right\}.$$

Proof. According to Theorem G.1, we only need to verify that the transformations R_k and T_k defined in (6.1) and (6.2) fulfill Assumptions 3.5, 3.6, G.1 and G.2 with

$$\tau \;=\; 1, \quad \kappa \;=\; (1+\zeta)^{-1}, \quad \lambda \;=\; 1, \quad \upsilon \;=\; \beta \;=\; \xi, \quad \text{and} \quad p \;=\; q \;=\; 1.$$

After the verification, the value of μ specified by (G.8) will turn out to be

$$\begin{aligned}\mu(\zeta,\xi) \;&=\; \min\left\{\frac{\Delta_0}{\|g_0\|},\; \frac{\alpha\gamma_0\kappa(\zeta)\,(1-\eta_1)}{5\xi},\; \frac{\alpha\gamma_0(1-\eta_1)}{5\xi},\; \frac{\alpha\gamma_0\kappa(\zeta)\,(1-\eta_1)}{5(L_h+L_f)}\right\}\\ &=\; \min\left\{\frac{\Delta_0}{\|g_0\|},\; \frac{\alpha\gamma_0\kappa(\zeta)\,(1-\eta_1)}{5\xi},\; \frac{\alpha\gamma_0\kappa(\zeta)\,(1-\eta_1)}{5(L_h+L_f)}\right\}.\end{aligned}$$

Assumption 3.5 holds trivially with $\tau = 1$ as $T_k = I$.
Condition (G.15) implies (F.1), meaning that Assumption 3.6 holds with $\kappa = (1+\zeta)^{-1}$.
Recalling that $R_k g_k = \hat{g}_k$ and $T_k^{\mathsf{T}} g_k = g_k$, condition (G.15) implies that

$$1 - \frac{g_k^{\mathsf{T}} T_k R_k g_k}{\|R_k g_k\|^2} \;=\; 1 - \frac{g_k^{\mathsf{T}} \hat{g}_k}{\|\hat{g}_k\|^2} \;\leq\; \frac{\|\hat{g}_k - g_k\|\|\hat{g}_k\|}{\|\hat{g}_k\|^2} \;\leq\; \frac{\xi\Delta_k}{\|\hat{g}_k\|},$$

justifying Assumption G.1 with $\lambda = 1$, $\upsilon = \xi$, and $q = 1$.

Considering the orthogonal projection $\bar{g}_k$ of $\hat{g}_k$ to the direction of g_k, we have

$$\sin\psi_k \;=\; \frac{\|\hat{g}_k - \bar{g}_k\|}{\|\hat{g}_k\|} \;\leq\; \frac{\|\hat{g}_k - g_k\|}{\|\hat{g}_k\|} \;\leq\; \frac{\xi\Delta_k}{\|\hat{g}_k\|},$$

which validates Assumption G.2 with $\beta = \xi$ and $p = 1$. The proof is then complete. □

Via the complexities revealed by Theorem G.2, we can again observe the impact of gradient inaccuracy on the worst-case iteration complexity of trust-region methods. Following the argument in Subsection 6.3.2, we can see according to Theorem G.2 that the gradient inaccuracy subject to (G.15) enlarges the iteration complexity bounds of Algorithm 6.1 at most by a factor of

$$\frac{\kappa(0)\,\mu(0,0)}{\kappa(\zeta)\,\mu(\zeta,\xi)} \;\leq\; (1+\zeta)^2 \max\left\{1,\; \frac{\xi}{L_h + L_f}\right\},$$

provided that the threshold ϵ for stationarity is not too large.

H Stabilizing the trust-region algorithms in GNU Octave

As an interesting application of our theoretical investigation in Section 6, we will examine the trust-region algorithms `fminunc` and `fsolve` built in GNU Octave (`www.gnu.org/software/octave/`) and show that their trust-region radius updating schemes are problematic, leading to severe instability when applied to problems with inaccurate gradient (for `fminunc`) or function (for `fsolve`) evaluation. The problem can be solved by replacing their trust-region updating schemes with any classical one.

We focus on `fminunc`, a trust-region algorithm for (1.1) with quadratic models based on Powell's damped BFGS update [117]. In version 5.1.0 (the latest one up to November 2019), it maintains the trust-region radius as follows: with $\Omega(t) = \min\{\max\{0.1,\ 0.8\,t\},\ 0.9\}$, $\tilde{\rho}_0 = 0$, and $\Gamma_0 = 1/2$, define

$$\begin{cases} \tilde{\rho}_{k+1} \;=\; \tilde{\rho}_k, \quad \Gamma_{k+1} \;=\; \Gamma_k^{\sqrt{2}}, \quad \Delta_{k+1} \;=\; \Gamma_k \Delta_k & \text{if } \rho_k < \Omega(\tilde{\rho}_k), \\ \tilde{\rho}_{k+1} \;=\; \rho_k, \quad \Gamma_{k+1} \;=\; 1/2, \quad \Delta_{k+1} \;\geq\; \min\{\Delta_k,\ \sqrt{2}\|D_k d_k\|\} & \text{else,} \end{cases} \tag{H.1}$$

where D_k is a scaling matrix. The definition of Δ_{k+1} when $\rho_k \geq \Omega(\tilde{\rho}_k)$ is not detailed as it is irrelevant to our discussion here. Note that $\Gamma_k \leq 1/2$ for each $k \geq 0$.

Unfortunately, (H.1) will not work when gradients are inaccurate, *i.e.*, when $\hat{g}_k = \nabla h_k(0)$ is not necessarily $g_k = \nabla f(x_k)$, following the notation of Section 6. To see why, consider a scenario where the trust-region model at iteration 0 works so well that $\rho_0 = \rho \geq 0.9$, while, additionally and deliberately, assuming that Δ_1 is small. Let us see what will happen by quick estimations without rigorous details. Since $\rho_0 = \rho \geq 0.9 \geq \Omega(\tilde{\rho}_0)$, (H.1) sets $\tilde{\rho}_1 = \rho_0 = \rho$. As Δ_1 is small, we have

$$\rho_1 \;=\; \frac{f(x_1) - f(x_1 + d_1)}{h_1(0) - h_1(d_1)} \;\approx\; \frac{-d_1^{\mathsf{T}} g_1}{-d_1^{\mathsf{T}} \hat{g}_1} \;\approx\; \frac{\hat{g}_1^{\mathsf{T}} g_1}{\|\hat{g}_1\|^2}, \tag{H.2}$$

where the first approximation replaces the differences by first-order terms, while the second is because problem $\min_{\|d\| \leq \Delta_k} h_k(d)$, solved by Powell's dogleg method [115] in `fminunc`, leads to $d_1 \approx -\Delta_1 \hat{g}_1 / \|\hat{g}_1\|$ when Δ_1 is small. Since $\tilde{\rho}_1 = \rho$, it is sensible to infer from (H.2) that $\rho_1 < \Omega(\tilde{\rho}_1)$ if $\hat{g}_1^{\mathsf{T}} g_1 < \Omega(\rho)\|\hat{g}_1\|^2$.

If $\rho_1 < \Omega(\tilde{\rho}_1)$ does occur, then (H.1) will set $\tilde{\rho}_2 = \tilde{\rho}_1 = \rho$, and $\Delta_2 \leq \Delta_1/2$, which is again small. We can reuse the inference above and speculate $\rho_2 < \Omega(\tilde{\rho}_2)$ provided that $\hat{g}_2^{\mathsf{T}} g_2 < \Omega(\rho)\|\hat{g}_2\|^2$. Indeed, if

$$\hat{g}_k^{\mathsf{T}} g_k \;<\; \Omega(\rho)\|\hat{g}_k\|^2 \quad \text{for each} \quad k \geq 1, \tag{H.3}$$

then it is likely that $\rho_k < \Omega(\tilde{\rho}_k)$ will take place consecutively, and (H.1) will keep setting $\tilde{\rho}_{k+1} = \rho$ and $\Delta_{k+1} \leq \Delta_k/2$, forcing $\{x_k\}$ to converge to a point within a neighborhood of x_0 as in Example 6.1, which necessarily implies failure of `fminunc` unless x_0 is sufficiently close to a stationarity point. This process is even expedited by the fact that $\{\Gamma_k\}$ decays according to (H.1). Indeed, (H.3) means that $\hat{g}_k$ always falls outside the admissible region (6.18) of inaccurate gradients found in Subsection 6.5 for $\eta_1 = \Omega(\rho)$, because, similar to (6.22), one can show that (H.3) is equivalent to

$$\left\| \hat{g}_k - \frac{g_k}{2\Omega(\rho)} \right\| \;>\; \frac{\|g_k\|}{2\Omega(\rho)} \quad \text{for each} \quad k \geq 1.$$

Although the argument made above contains stringent conditions and unjustified estimations, we will see that it reflects very well the difficulty encountered by `fminunc` in practice. The reason behind the difficulty is that (H.1) does not contain a small η_1 that ensures $\Delta_{k+1} \geq \Delta_k$ whenever $\rho_k > \eta_1$. Indeed, in the scenario described above, `fminunc` demands $\rho_k \geq \Omega(\rho)$ before stopping the contraction of Δ_k. The value of $\Omega(\rho)$ is at least $\Omega(0.9) = 0.72$ due to the aforementioned condition that $\rho \geq 0.9$.

This situation can be rectified by switching (H.1) to any classical scheme with a small η_1. As an illustration (not necessarily the optimal choice), we correct the `fminunc` code using (6.31) with $\eta_1 = 1/10$, $\eta_2 = 3/4$, and then test the corrected version together with the GNU Octave version on problems with relative gradient error $\zeta \in \{0, 1/50, 2/50, \ldots, 49/50, 1\}$ as described in Subsection 6.7.2. Figure 9 plots their effectiveness defined in (6.29) against relative gradient error. Obviously, the GNU Octave version suffered from gradient inaccuracy, while the corrected version behaved stably as ensured by our theory.

`fsolve` built in GNU Octave is a trust-region algorithm for nonlinear least-squares. Its trust-region radius updating scheme is essentially the same as (H.1) and can be corrected likewise. We also mention that the `fminunc` and `fsolve` in MATLAB (version R2019b) update trust-region radius using particular instances of (3.65) with $\eta_1 = 1/4$ and $\eta_1 = 1/20$ respectively, and hence they are stable with respect to gradient inaccuracy according to our theory in Section 6.

We end this whole paper by an interesting fact hidden in Figure 9. Note that (H.1) cannot be included by (3.65) as a special case and hence stays outside the coverage of our theory in Section 6. However, in the aforementioned scenario where (H.1) does not work well, `fminunc` behaves as if it took a classical scheme to update the trust-region radius, but chose $\eta_1 = \Omega(0.9) = 0.72$, a value too large. By such an analogy, Theorem 6.1 suggests that the algorithm will suffer when relative gradient error has magnitude

$$\zeta \;\geq\; [\Omega(0.9)]^{-1} - 1 \;\approx\; 0.39.$$

Figure 9 shows a vertical line at $\zeta = 0.39$. We can see a rapid drop in the effectiveness of GNU Octave `fminunc` once this line is passed. It is entertaining to realize that such a behavior is predicted so precisely by the theory derived from Algorithm 2.1, a framework that looks so unrelated to `fminunc`.

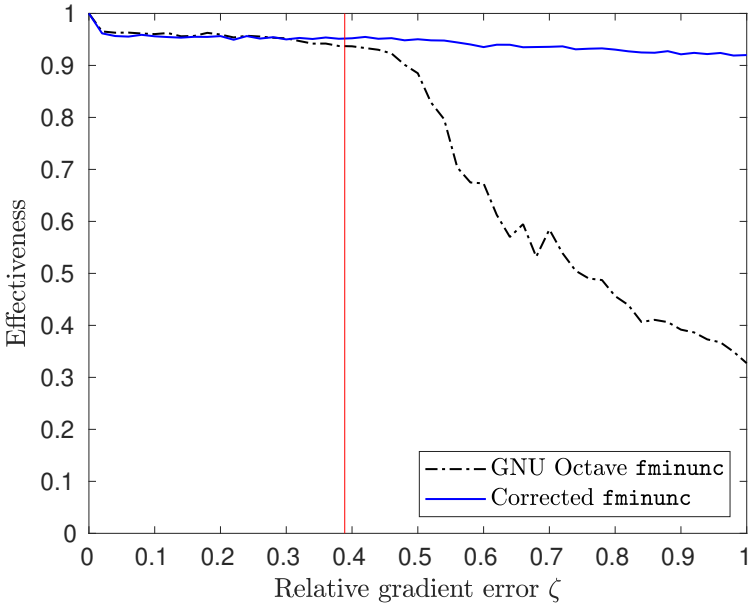


Figure 9: Effectiveness of two versions of `fminunc` plotted against relative gradient error.

References

[1] C. Audet, J. E. Dennis, Jr., and S. Le Digabel. Parallel space decomposition of the mesh adaptive direct search algorithm. *SIAM J. Optim.*, 19:1150–1170, 2008.

[2] M. Baboulin, A. Buttari, J. Dongarra, J. Kurzak, J. Langou, J. Langou, P. Luszczek, and S. Tomov. Accelerating scientific computations with mixed precision algorithms. *Comput. Phys. Commun.*, 180:2526–2533, 2009.

[3] A. S. Bandeira, K. Scheinberg, and L. N. Vicente. Convergence of trust-region methods based on probabilistic models. *SIAM J. Optim.*, 24:1238–1264, 2014.

[4] J. W. Bandler, R. M. Biernacki, S. H. Chen, P. A. Grobelny, and R. H. Hemmers. Space mapping technique for electromagnetic optimization. *IEEE Trans. Microw. Theory Tech.*, 42:2536–2544, 1994.

[5] J. W. Bandler, R. M. Biernacki, S. H. Chen, R. H. Hemmers, and K. Madsen. Electromagnetic optimization exploiting aggressive space mapping. *IEEE Trans. Microw. Theory Tech.*, 43:2874–2882, 1995.

[6] A. S. Berahas, R. H. Byrd, and J. Nocedal. Derivative-free optimization of noisy functions via quasi-Newton methods. *SIAM J. Optim.*, 29:965–993, 2019.

[7] E. G. Birgin, J. L. Gardenghi, J. M. Martínez, S. A. Santos, and Ph. L. Toint. Worst-case evaluation complexity for unconstrained nonlinear optimization using high-order regularized models. *Math. Program.*, 163:359–368, 2017.

[8] A. Borzì and V. Schulz. Multigrid methods for PDE optimization. *SIAM Rev.*, 51:361–395, 2009.

[9] A. Borzì and V. Schulz. *Computational Optimization of Systems Governed by Partial Differential Equations*. Comput. Sci. Eng. SIAM, Philadelphia, 2011.

[10] L. Bottou. Large-scale machine learning with stochastic gradient descent. In Y. Lechevallier and G. Saporta, editors, *Proceedings of COMPSTAT'2010*, pages 177–186, Heidelberg, 2010. Physica-Verlag.

[11] J. Bramble, J. Pasciak, and A. Schatz. The construction of preconditioners for elliptic problems by substructuring, I. *Math. Comp.*, 47:103–134, 1986.

[12] R. H. Byrd, R. Schnabel, and G. A. Shultz. Approximate solution of the trust region problem by minimization over two-dimensional subspaces. *Math. Program.*, 40:247–263, 1988.

[13] X.-C. Cai and D. E. Keyes. Nonlinearly preconditioned inexact Newton algorithms. *SIAM J. Sci. Comput.*, 24:183–200, 2002.

[14] X.-C. Cai and M. Sarkis. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM J. Sci. Comput.*, 21:792–797, 1999.

[15] R. Carter. On the global convergence of trust region algorithms using inexact gradient information. *SIAM J. Numer. Anal.*, 28:251–265, 1991.

[16] R. Carter. A worst-case example using linesearch methods for numerical optimization with inexact gradient evaluations. Technical Report MCS-P283-1291, Argonne National Laboratory, 1991.

[17] R. Carter. Numerical experience with a class of algorithms for nonlinear optimization using inexact function and gradient information. *SIAM J. Sci. Comput.*, 14, 1993.

[18] C. Cartis, N. I. M. Gould, and Ph. L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. Part I: motivation, convergence and numerical results. *Math. Program.*, 127:245–295, 2011.

[19] C. Cartis, N. I. M. Gould, and Ph. L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. Part II: worst-case function-and derivative-evaluation complexity. *Math. Program.*, 130:295–319, 2011.

[20] C. Cartis, N. I. M. Gould, and Ph. L. Toint. On the oracle complexity of first-order and derivative-free algorithms for smooth nonconvex minimization. *SIAM J. Optim.*, 22:66–86, 2012.

[21] C. Cartis, N. I. M. Gould, and Ph. L. Toint. Evaluation complexity bounds for smooth constrained nonlinear optimization using scaled KKT conditions and high-order models. In I. C. Demetriou and P. M. Pardalos, editors, *Approximation and Optimization*. Springer, Berlin, 2019.

[22] C. Cartis and Ph. L. Gould, N. I. M. Toint. Complexity bounds for second-order optimality in unconstrained optimization. *J. Complexity*, 28:93 – 108, 2012.

[23] C. Cartis and K. Scheinberg. Global convergence rate analysis of unconstrained optimization methods based on probabilistic models. *Math. Program.*, 169:337–375, 2018.

[24] T. F. Chan and T. P. Mathew. Domain decomposition algorithms. *Acta Numer.*, 3:61–143, 1994.

[25] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6571–6583. Curran Associates, Inc., 2018.

[26] X. Chen and Ph. L. Toint. High-order evaluation complexity for convexly-constrained optimization with non-Lipschitzian group sparsity terms. *arXiv:1902.10767*, 2019.

[27] X. Chen, Ph. L. Toint, and H. Wang. Partially separable convexly-constrained optimization with non-Lipschitzian singularities and its complexity. *SIAM J. Optim.*, 29:874–903, 2019.

[28] A. R. Conn, N. I. M. Gould, A. Sartenaer, and Ph. L. Toint. Convergence properties of minimization algorithms for convex constraints using a structured trust region. *SIAM J. Optim.*, 6:1059–1086, 1996.

[29] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*. Springer-Verlag, Berlin, 1992.

[30] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-Region Methods*. SIAM, Philadelphia, 2000.

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

[32] A. R. Conn, K. Scheinberg, and L. N. Vicente. Global convergence of general derivative-free trust-region algorithms to first- and second-order critical points. *SIAM J. Optim.*, 20:387–415, 2009.

[33] F. E. Curtis, Z. Lubberts, and D. P. Robinson. Concise complexity analyses for trust region methods. *Optim. Lett.*, 12:1713–1724, 2018.

[34] F. E. Curtis and D. P. Robinson. Regional complexity analysis of algorithms for nonconvex smooth optimization. Technical Report COR@L Technical Report 18T-003-R1, Department of Industrial and Systems Engineering, Lehigh University, Lehigh, 2018.

[35] F. E. Curtis, D. P. Robinson, and M. Samadi. A trust region algorithm with a worst-case iteration complexity of $\mathcal{O}(\epsilon^{-3/2})$ for nonconvex optimization. *Math. Program.*, 162:1–32, 2017.

[36] F. E. Curtis, D. P. Robinson, and M. Samadi. An inexact regularized newton framework with a worst-case iteration complexity of $\mathcal{O}(\epsilon^{-3/2})$ for nonconvex optimization. *IMA J. Numer. Anal.*, 39:1296–1327, 2018.

[37] A. d'Aspremont. Smooth optimization with approximate gradient. *SIAM J. Optim.*, 19:1171–1183, 2008.

[38] J. C. De los Reyes. *Numerical PDE-Constrained Optimization*. Springer, Berlin, 2015.

[39] O. Devolder, F. Glineur, and Y. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Math. Program.*, 146:37–75, 2014.

[40] V. Dolean, M. J. Gander, W. Kheriji, F. Kwok, and R. Masson. Nonlinear preconditioning: How to use a nonlinear Schwarz method to precondition Newton's method. *SIAM J. Sci. Comput.*, 38:A3357–A3380, 2016.

[41] V. Dolean, P. Jolivet, and F. Nataf. *An Introduction to Domain Decomposition Methods: Algorithms, Theory, and Parallel Implementation*. SIAM, Philadelphia, 2015.

[42] Q. Dong, X. Liu, Z. Wen, and Y. Yuan. A parallel line search subspace correction method for composite convex optimization. *J. Oper. Res. Soc. China*, 3:163–187, 2015.

[43] J. Douglas. A method of numerical solution of the problem of Plateau. *Ann. of Math.*, 29:180–188, 1927.

[44] J. Douglas. Solution of the problem of Plateau. *Trans. Amer. Math. Soc.*, 33:263–321, 1931.

[45] J. W. Eaton. *GNU Octave 4.2 Reference Manual*. Samurai Media Limited, United Kingdom, 2017.

[46] E. Efstathiou and M. J. Gander. Why restricted additive Schwarz converges faster than additive Schwarz. *BIT*, 43:945–959, 2003.

[47] J. Fan. Convergence rate of the trust region method for nonlinear equations under local error bound condition. *Comput. Optim. Appl.*, 34:215–227, 2006.

[48] J. Fan and Y. Yuan. A new trust region algorithm with trust region radius converging to zero. In D. Li and X. Q. Cai, editors, *Proceedings of the 5th International Conference on Optimization: Techniques and Applications*, pages 786–794. World Scientific Publishing, Singapore, 2001.

[49] M. C. Ferris and O. L. Mangasarian. Parallel variable distribution. *SIAM J. Optim.*, 4:815–832, 1994.

[50] A. Frommer and H. Schwandt. A unified representation and theory of algebraic additive Schwarz and multisplitting methods. *SIAM J. Matrix Anal. Appl.*, 18:893–912, 1997.

[51] A. Frommer and D. B. Szyld. An algebraic convergence theory for restricted additive Schwarz methods using weighted max norms. *SIAM J. Numer. Anal.*, 39:463–479, 2002.

[52] M. Fukushima. Parallel variable transformation in unconstrained optimization. *SIAM J. Optim.*, 8:658–672, 1998.

[53] M. J. Gander. Schwarz methods over the course of time. *Electron. Trans. Numer. Anal.*, 31:228–255, 2008.

[54] M. J. Gander. On the origins of linear and non-linear preconditioning. In C.-O. Lee, X.-C. Cai, D. E. Keyes, H. Kim, A. Klawonn, E.-J. Park, and O. B. Widlund, editors, *Domain Decomposition Methods in Science and Engineering XXIII*, pages 153–161. Springer, Berlin, 2017.

[55] M. J. Gander and A. Loneland. SHEM: An optimal coarse space for RAS and its multiscale approximation. In C.-O. Lee, X.-C. Cai, D. E. Keyes, H. Kim, A. Klawonn, E.-J. Park, and O. B. Widlund, editors, *Domain Decomposition Methods in Science and Engineering XXIII*, pages 313–321. Springer, Berlin, 2017.

[56] M. J. Gander, A. Loneland, and T. Rahman. Analysis of a new harmonically enriched multiscale coarse space for domain decomposition methods. *arXiv:1512.05285*, 2015.

[57] M. J. Gander and G. Wanner. The origins of the alternating Schwarz method. In J. Erhel, M. J. Gander, L. Halpern, G. Pichot, T. Sassi, and O. B. Widlund, editors, *Domain Decomposition Methods in Science and Engineering XXI*, pages 487–495. Springer, Berlin, 2014.

[58] R. Ge, F. Huang, C. Jin, and Y. Yuan. Escaping from saddle points — online stochastic gradient for tensor decomposition. In P. Grünwald, E. Hazan, and S. Kale, editors, *Proceedings of The 28th Conference on Learning Theory*, pages 797–842. PMLR, 2015.

[59] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Comput. Surv.*, 23:5–48, 1991.

[60] E. Gorbunov, F. Hanzely, and P. Richtárik. A unified theory of SGD: Variance reduction, sampling, quantization and coordinate descent. *arXiv:1905.11261*, 2019.

[61] N. I. M. Gould, D. Orban, A. Sartenaer, and Ph. L. Toint. Sensitivity of trust-region algorithms to their parameters. *4OR*, 3:227–241, 2005.

[62] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Comput. Optim. Appl.*, 60:545–557, 2015.

[63] G. N. Grapiglia, J. Yuan, and Y. Yuan. On the convergence and worst-case complexity of trust-region and regularization methods for unconstrained optimization. *Math. Program.*, 152:491–520, 2015.

[64] G. N. Grapiglia, J. Yuan, and Y. Yuan. Nonlinear stepsize control algorithms: Complexity bounds for first-and second-order optimality. *J. Optim. Theory Appl.*, 171:980–997, 2016.

[65] S. Gratton, C. Royer, and L. N. Vicente. A decoupled first/second-order steps technique for nonconvex nonlinear unconstrained optimization with improved complexity bounds. to appear in *Math. Program.*, 2019.

[66] S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang. Direct search based on probabilistic descent. *SIAM J. Optim.*, 25:1515–1541, 2015.

[67] S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang. Complexity and global rates of trust-region methods based on probabilistic models. *IMA J. Numer. Anal.*, 38:1579–1597, 2018.

[68] S. Gratton and Ph. L. Toint. A note on solving nonlinear optimization problems in variable precision. *arXiv:1812.03467*, 2018.

[69] S. Gratton and L. N. Vicente. A surrogate management framework using rigorous trust-region steps. *Optim. Methods Softw.*, 29:10–23, 2014.

[70] A. Griewank. The modification of Newton's method for unconstrained optimization by bounding cubic terms. Technical Report Technical Report NA/12, DAMTP, University of Cambridge, 1981.

[71] A. Griewank and Ph. L. Toint. On the unconstrained optimization of partially separable functions. In M. J. D. Powell, editor, *Nonlinear Optimization 1981*, pages 119–137. Academic Press, London, 1982.

[72] C. Groß. *A Unifying Theory for Nonlinear Additively and Multiplicatively Preconditioned Globalization Strategies: Convergence Results and Examples From the Field of Nonlinear Elastostatics and Elastodynamics*. PhD thesis, University of Bonn, Bonn, 2009.

[73] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan. Deep learning with limited numerical precision. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, pages 1737–1746. PMLR, 2015.

[74] G. Heidel and V. Schulz. A Riemannian trust-region method for low-rank tensor completion. *Numer. Linear Algebra Appl.*, 25:e2175, 2018.

[75] M. Heinkenschloss and L. N. Vicente. Analysis of inexact trust-region SQP algorithms. *SIAM J. Optim.*, 12:283–302, 2002.

[76] C. F. Higham and D. J. Higham. Deep learning: An introduction for applied mathematicians. *SIAM Rev.*, to appear.

[77] D. J. Higham. Trust region algorithms and timestep selection. *SIAM J. Numer. Anal.*, 37:194–210, 1999.

[78] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, second edition, 2002.

[79] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich. *Optimization with PDE Constraints*. Springer, Berline, 2009.

[80] M. Holst. Algebraic Schwarz theory. Technical Report CRPC-994-10, Department of Applied Mathematics and CRPC, California Institute of Technology, California, 1994.

[81] P. D. Hough and J. C. Meza. A class of trust-region methods for parallel optimization. *SIAM J. Optim.*, 13:264–282, 2002.

[82] L. Jacob, G. Obozinski, and J.-P. Vert. Group lasso with overlap and graph lasso. In L. Bottou and M. Littman, editors, *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 433–440. ACM, New York, 2009.

[83] C. G. J. Jacobi. Über eine neue Auflösungsart der bei der Methode der kleinsten Quadrate vorkommenden linearen Gleichungen. *Astronomische Nachrichten*, 22:297–306, 1845.

[84] C. Jin, R. Ge, P. Netrapalli, Sham M. Kakade, and M. I. Jordan. How to escape saddle points efficiently. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, pages 1724–1732. JMLR, 2017.

[85] D. Kalamkar et al. A study of BFLOAT16 for deep learning training. *arXiv:1905.12322*, 2019.

[86] C. T. Kelley. *Iterative Methods for Optimization*. SIAM, Philadelphia, 1999.

[87] L. Kugler. Is "good enough" computing good enough? *Commun. ACM*, 58:12–14, 2015.

[88] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quart. Appl. Math.*, 2:164–168, 1944.

[89] S. Leyffer, S. M. Wild, M. Fagan, M. Snir, K. Palem, H. Finkel, and K. Yoshii. Doing Moore with less – leapfrogging Moore's law with inexactness for supercomputing. Technical Report ANL/MCS-P6077-1016, Argonne National Laboratory MCS, 2016.

[90] K.-A. Lie. *An introduction to reservoir simulation using MATLAB/GNU Octave*. Cambridge University Press, Cambridge, 2019.

[91] P. L. Lions. On the Schwarz alternating method. I. In R. Glowinski, G. H. Golub, G. A. Meurant, and J. Périaux, editors, *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 1–42. SIAM, Philadelphia, 1988.

[92] J. Mandel and B. Sousedík. Coarse spaces over the ages. In Y. Huang, R. Kornhuber, O. B. Widlund, and J. Xu, editors, *Domain decomposition Methods in Science and Engineering XIX*, pages 213–220. Springer, 2011.

[93] O. L. Mangasarian. Parallel gradient distribution in unconstrained optimization. *SIAM J. Control Optim.*, 33:1916–1925, 1995.

[94] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Appl. Math.*, 11:431–441, 1963.

[95] J. R. R. A. Martins, J. J. Alonso, and J. J. Reuther. A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design. *Optim. Eng.*, 6:33–62, 2005.

[96] L. Meier, S. Van De Geer, and P. Bühlmann. The group lasso for logistic regression. *J. R. Stat. Soc. Ser. B. Stat. Methodol.*, 70:53–71, 2008.

[97] J. J. Moré. Recent developments in algorithms and software for trust region methods. In A. Bachem, M. Grötschel, and B. Korte, editors, *Mathematical Programming: The State of Art*. Springer-Verlag, Berlin, 1983.

[98] J. J. Moré and D. Sorensen. Computing a trust region step. *SIAM J. Sci. Stat. Comp.*, 4:553–572, 1983.

[99] J. J. Moré and S. M. Wild. Estimating computational noise. *SIAM J. Sci. Comput.*, 33:1292–1314, 2011.

[100] J. J. Moré and S. M. Wild. Estimating derivatives of noisy simulations. *ACM Trans. Math. Software*, 38:19:1–19:21, 2012.

[101] J. J. Moré and S. M. Wild. Do you trust derivatives or differences? *J. Comput. Phys.*, 273:268–277, 2014.

[102] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, London, 2004.

[103] Yu. Nesterov. A method of solving a convex programming problem with $\mathcal{O}(1/k^2)$ convergence rate. *Soviet Mathematics Doklady*, 27:372–376, 1983.

[104] Yu. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J. Optim.*, 22:341–362, 20112.

[105] Yu. Nesterov and B. T. Polyak. Cubic regularization of Newton method and its global performance. *Math. Program.*, 108:177–205, 2006.

[106] Q. Ni and Y. Yuan. A subspace limited memory quasi-Newton algorithm for large-scale nonlinear bound constrained optimization. *Math. Comp.*, 66:1509–1520, 1997.

[107] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Math. Comp.*, 35:773–782, 1980.

[108] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Ser. Oper. Res. Financ. Eng. Springer, New York, second edition, 2006.

[109] M. R. Osborne. Nonlinear least squares — the Levenberg algorithm revisited. *J. Austral. Math. Soc.*, 19:343–357, 1976.

[110] K. V. Palem. Inexactness and a future of computing. *Phil. Trans. R. Soc. A*, 372:20130281, 2014.

[111] B. Peherstorfer, K. Willcox, and M. Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *SIAM Rev.*, 60:550–591, 2018.

[112] Z. Peng, Y. Xu, M. Yan, and W. Yin. ARock: An algorithmic framework for asynchronous parallel coordinate updates. *SIAM J. Sci. Comput.*, 38:A2851–A2879, 2016.

[113] M. Pilanci and M. J. Wainwright. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM J. Optim.*, 27:205–245, 2017.

[114] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4:1–17, 1964.

[115] M. J. D. Powell. A hybrid method for nonlinear equations. In P. Rabinowitz, editor, *Numerical Methods for Nonlinear Algebraic Equations*, pages 87–114. Gordon & Breach Science, London, 1970.

[116] M. J. D. Powell. Convergence properties of a class of minimization algorithms. In O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, editors, *Nonlinear Programming 2: Proceedings of the Special Interest Group on Mathematical Programming Symposium Conducted by the Computer Sciences Department at the University of Wisconsin-Madison, April 15–17, 1974*, pages 1–27. Academic Press, 1975.

[117] M. J. D. Powell. Algorithms for nonlinear constraints that use Lagrangian functions. *Math. Program.*, 14:224–248, 1978.

[118] M. J. D. Powell. On the global convergence of trust region algorithms for unconstrained minimization. *Math. Program.*, 29:297–303, 1984.

[119] M. J. D. Powell. The NEWUOA software for unconstrained optimization without derivatives. In G. Di Pillo and M. Roma, editors, *Large-scale nonlinear optimization*, pages 255–297. Springer, Boston, 2006.

[120] M. J. D. Powell. On the convergence of a wide range of trust region methods for unconstrained optimization. *IMA J. Numer. Anal.*, 30:289–301, 2010.

[121] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Oxford Science Publications, Oxford, 1999.

[122] P. Richtárik and M. Takáč. Parallel coordinate descent methods for big data optimization. *Math. Program.*, 156:433–484, 2016.

[123] H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22:400–407, 1951.

[124] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, New Jersey, 1970.

[125] R. T. Rockafellar and R. J.-B. Wets. *Variational Analysis*. Springer, Berlin, 1998.

[126] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, second edition, 2003.

[127] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and Ph. Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, pages 1889–1897. PMLR, 2015.

[128] H. A. Schwarz. Über einen Grenzübergang durch alternierendes Verfahren. *Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich*, 15:272–286, 1870.

[129] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNS. In H. Li, H. Meng, B. Ma, E. Chng, and L. Xie, editors, *Fifteenth Annual Conference of the International Speech Communication Association*. ISCA, 2014.

[130] B. F. Smith, P. E. Bjørstad, and W. D. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, Cambridge, 1996.

[131] M. V. Solodov. New inexact parallel variable distribution algorithms. *Comput. Optim. Appl.*, 7:165–182, 1997.

[132] M. V. Solodov and S. K. Zavriev. Error stability properties of generalized gradient-type algorithms. *J. Optim. Theory Appl.*, 98:663–680, 1998.

[133] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM J. Numer. Anal.*, 20:626–637, 1983.

[134] X.-C. Tai and M. Espedal. Rate of convergence of some space decomposition methods for linear and nonlinear problems. *SIAM J. Numer. Anal.*, 35:1558–1570, 1998.

[135] X.-C. Tai and J. Xu. Global and uniform convergence of subspace correction methods for some convex optimization problems. *Math. Comp.*, 71:105–124, 2002.

[136] S. W. Thomas. *Sequential Estimation Techniques for Quasi-Newton Algorithms*. PhD thesis, Cornell University, New York, 1975.

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

[138] Ph. L. Toint. Global convergence of a class of trust-region methods for nonconvex minimization in Hilbert space. *IMA J. Numer. Anal.*, 8:231–252, 1988.

[139] Ph. L. Toint. Nonlinear stepsize control, trust regions and regularizations for unconstrained optimization. *Optim. Methods Softw.*, 28:82–95, 2013.

[140] Ph. L. Toint and D. Tuyttens. On large scale nonlinear network optimization. *Math. Program.*, 48:125–159, 1990.

[141] A. Toselli and O. B. Widlund. *Domain Decomposition Methods — Algorithms and Theory*, volume 34 of *Springer Ser. Comput. Math.* Springer, Berlin, 2006.

[142] Ø. Tråsdahl and E. M. Rønquist. High order numerical approximation of minimal surfaces. *J. Comput. Phys.*, 230:4795–4810, 2011.

[143] M. Ulbrich. Optimization methods in Banach spaces. In *Optimization with PDE Constraints*, pages 97–156. Springer, Dordrecht, 2009.

[144] L. N. Vicente. Space mapping: Models, sensitivities, and trust-regions methods. *Optim. Eng.*, 4:159–175, 2003.

[145] X. Wang and Y. Yuan. Stochastic trust region methods with trust region radius depending on probabilistic models. *arXiv:1904.03342*, 2019.

[146] Z. Wang and Y. Yuan. A subspace implementation of quasi-Newton trust region methods for unconstrained optimization. *Numer. Math.*, 104:241–269, 2006.

[147] A. J. Wathen. Preconditioning. *Acta Numer.*, 24:329–376, 2015.

[148] O. B. Widlund. The development of coarse spaces for domain decomposition algorithms. In M. Bercovier, M. J. Gander, R. Kornhuber, and O. B. Widlund, editors, *Domain Decomposition Methods in Science and Engineering XVIII*, pages 241–248. Springer, 2009.

[149] S. Wright. Coordinate descent algorithms. *Math. Program.*, 151:3–34, 2015.

[150] Y. Wu, E. Mansimov, R. Grosse, S. Liao, and J. Ba. Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. In I. Guyon, U. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 5279–5288. Curran Associates, Inc., 2017.

[151] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM Rev.*, 34:581–613, 1992.

[152] P. Xu, F. Roosta-Khorasani, and M. W. Mahoney. Second-order optimization for non-convex machine learning: An empirical study. *arXiv:1708.07827*, 2018.

[153] Z. Yao, A. Gholami, P. Xu, K. Keutzer, and M. W. Mahoney. Trust region based adversarial attack on neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[154] Y. Yuan. An example of non-convergence of trust region algorithms. In Y. Yuan, editor, *Advances in Nonlinear Programming*, pages 205–215. Kluwer Academic Publishers, Dordrecht, 1998.

[155] Y. Yuan. On the truncated conjugate gradient method. *Math. Program.*, 87:561–573, 2000.

[156] Y. Yuan. A review of trust region algorithms for optimization. In J. M. Ball and J. C. R. Hunt, editors, *Proceedings of the Fourth International Congress on Industrial and Applied Mathematics*, pages 271–282. Oxford University Press, Oxford, 2000.

[157] Y. Yuan. Subspace techniques for nonlinear optimization. In R. Jeltsch, T.-T. Li, and I. H. Sloan, editors, *Some topics in industrial and applied mathematics*, pages 206–218. World Scientific Publishing, Singapore, 2007.

[158] Y. Yuan. A review on subspace methods for nonlinear optimization. In S. Y. Jang, Y. R. Kim, Lee D.-W., and I. Yie, editors, *Proceedings of the International Congress of Mathematicians (Seoul 2014)*, pages 807–827. Kyung Moon Sa Co. Ltd., Seoul, 2014.

[159] Y. Yuan. Recent advances in trust region algorithms. *Math. Program.*, 151:249–281, 2015.

[160] Y. Yuan and J. Stoer. A subspace study on conjugate gradient algorithms. *ZAMM Z. Angew. Math. Mech.*, 75:69–77, 1995.