FACULTE DES SCIENCES
DEPARTEMENT DE MATHEMATIQUE

Multilevel Optimization: Convergence Theory, Algorithms and Application to Derivative-Free Optimization

Dissertation présentée par
Melissa Weber Mendonça
pour l'obtention du grade
de Docteur en Sciences

Composition du Jury:

Serge Gratton
Annick Sartenaer
Philippe Toint (Promoteur)
Michael Ulbrich
Jean-Jacques Strodiot

2009

Rempart de la Vierge, 13
B-5000 Namur (Belgique)

Imprimé en Belgique

ISBN : 978-2-87037-647-8
Dépôt légal: D / 2009 / 1881 / 35

Facultés Universitaires Notre-Dame de la Paix
Faculté des Sciences
rue de Bruxelles, 61, B-5000 Namur, Belgium

Facultés Universitaires Notre-Dame de la Paix
Faculté des Sciences
Rue de Bruxelles, 61, B-5000 Namur, Belgium

Optimisation Multiniveaux: Théorie de Convergence, Algorithmes et Application en Optimisation Sans Dérivées

par Melissa Weber Mendonça

Résumé: Nous présentons des nouveaux développements dans le cadre des méthodes de région de confiance multi-niveaux pour l'optimisation non-linéaire. Motivés par les résultats obtenus pour l'optimisation sans contraintes, nous avons développé une théorie de convergence pour les problèmes aux contraintes de borne, et pour les régions de confiance définies par la norme infinie. Comme alternative à ce genre de méthodes, nous avons développé un algorithme qui utilise des techniques multi-niveaux pour la résolution exacte du sous-problème de la région de confiance. Cette nouvelle méthode garantit la convergence de l'algorithme à un point critique du deuxième ordre, avec un coût associé nettement inférieur comparé aux méthodes classiques. Malheureusement, il y a des problèmes où les dérivées de la fonction objectif ne peuvent pas être calculées. Les méthodes utilisées pour résoudre ce genre de problèmes sont limitées par le coût du calcul de la fonction objective. Nous présentons donc une version multi-niveaux de cette méthode qui permet de traiter des problèmes de taille plus conséquente, ainsi que des résultats numériques obtenus avec ce nouvel algorithme.

Multilevel Optimization: Convergence Theory, Algorithms and Application to Derivative-Free Optimization

by Melissa Weber Mendonça

Abstract: We present new developments in the context of multilevel trust-region methods for nonlinear optimization. Motivated by the results obtained for unconstrained problems, we have extended the convergence theory for bound-constrained problems and for the use of infinity-norm trust regions. As an alternative for these methods, we have developed an algorithm that uses multilevel techniques for the exact resolution of the trust-region subproblem. This new method guarantees the convergence of the trust-region algorithm to a second-order critical point, with a much reduced associated cost when compared to classical methods. Unfortunately, there are problems for which we cannot compute the derivatives of the objective function. The methods used today to solve this kind of problems are limited by the cost of the objective function computation. We present a multilevel version of one of these methods, that allows for the treatment of larger instances of the problem, as well as numerical results obtained with this new algorithm.

Dissertation doctorale en Sciences mathématiques (Ph.D. thesis in Mathematics)
Date: 13-05-2009
Département de Mathématique
Promoteur (Advisor): Prof. Ph. L. Toint

Acknowledgements

This thesis could not have been done if it were not for the people who supported me and helped me in my way here. First and foremost, I would like to thank my advisor, Philippe Toint, for giving me the opportunity to do this project and for all the invaluable help and support while I was in Belgium. Your kind words and insightful comments were immensely appreciated. I could not think of a better advisor to work with. Thank you for accepting a student that you barely knew and for believing that I could do this. I would also like to thank Annick Sartenaer, Serge Gratton, Jean-Jacques Strodiot and Michael Ulbrich, for having accepted to be in the jury for this thesis and for their kind remarks and interesting questions.

I would like to express my gratitude to all my friends and colleagues, both in Brazil and Belgium, for all the laughs, the support, the long conversations and their friendship. Thank you Dimitri Tomanos, for all the fun and the work we did together, and also for all the game cheats. I'll miss the "third-world" jokes and the Frank Sinatra songs. I would like to thank my friends Katia Demaseure, Caroline Sainvitu, Charlotte Beauthier, Anne-Sophie Libert, Emilie Wanufelle, Sebastian Xhonneux, Julien Dufey, Simone Righi, Vincent Malmedy, Laetitia Legrain, Stéphane Valk, Nicolas Delsate, Patrick Laloyaux, Benoît Noyelles, Joffray Baune, Audrey Compère, Sandrine D'Hoedt, Martine de Vleeschouwer, André Füzfa, Bertrand Chenal and all the members of the mathematics department at Namur. I would also like to thank Alexander Thekale, Margherita Porcelli and Selime Gürol for their friendship, even at a distance. Thank you also to Luc Goffinet and all the members of the Amnesty International group at the University of Namur. These four years in Belgium have been wonderful and I am genuinely lucky to have been a part of the mathematics department and of the community of the Facultés Universitaires Notre-Dame de la Paix. In particular, a big thank you to Martine Van Caenegem and Pascale Hermans for their help and good humour. Thank you, Eric Cornélis, for the constant smile and the willingness to help. I would also like to thank Murielle Haguinet, for all the invaluable help and advice, and for always being ready to solve any problem as quickly as possible. I am deeply grateful to you all.

I want to especially thank my parents, Luli and Claudio, because without their support and their sacrifices I would never be here today. They have inspired and supported me through the difficult choices and celebrated the victories with me. Thank you, dad, for teaching me that science was cool; thank you, mom, for teaching me to be proud of being a woman. Without your guidance and love, none of this would have been possible.

Last, but not least, I would like to thank Rodrigo Vieira. Thank you for being there, for supporting me every step of the way, and for helping me through the hard times. I cannot thank you enough. I hope someday I can do for you what you did for me.

Contents

Introduction

In the history of mathematics, optimization is a fairly recent field, even if the problem of minimizing or maximizing some quantity is a natural one that arises frequently in life. One might wish to minimize one's route to work, or maximize production in a factory, for example. More importantly, many modelling problems in applied sciences, engineering and economics can also be formulated as optimization problems. Applications as diverse as weather forecasting, aircraft design, scheduling, medical imaging and mobile network communications can all be formulated as optimization problems.

Throughout history, however, these kinds of problems were treated by heuristics, either because of lack of computing power necessary to treat large problems, or because of a lack of suitable methods that exploit each problem's characteristics in order to effectively *solve* it, in a rigorous mathematical sense. The first optimization technique developed for this specific end, known as the *steepest descent* method (or gradient method), is attributed to Gauss (1777 – 1855). Mathematics has come a long way from that time, and so has optimization, which today allows us to solve many extremely complicated and costly problems in the applied sciences and in life in general. The advances in computing we have experienced in the last 50 years have been key to this development, with mathematical programming becoming one of the most important areas of mathematics today.

Along with the optimization approach, linear systems are fundamental in modelling these types of problems. In particular, large sparse systems are very common, especially in mathematical formulations resulting from the discretization of elliptic Partial Differential Equations. Several methods have been proposed for the solution of large linear systems of equations, and here we will focus on the *Multigrid* methodology which has been established as one of the most efficient techniques for this end.

In this thesis, we aim to present three different techniques, developed in the course of four years, and which are somewhat different from each other. However, these three techniques are profoundly related, in that they are all a mix of trust-region and multigrid approaches, and as such we will refer to them as *Multilevel* strategies. These multilevel strategies are a part of a growing number of optimization methods, and are thus in the cutting edge of mathematical programming. We will present a brief introduction to the elements that make up these methods, and detail our own contributions to this rapidly growing field of mathematics.

First, we will present a new multilevel method designed to solve the trust-region subproblem exactly. It is capable of treating problems much larger than the ones solvable by the *Moré-Sorensen* method, and it consists in interpreting the trust-region subproblem as a *multigrid* problem with one parameter. In this sense, it is a more direct application of multigrid techniques to a

fundamental part of the trust-region algorithm, without altering the basic trust-region iteration.

In contrast with this methodology, we will present the Recursive Multilevel Trust-Region class of methods, abbreviated here by RMTR, developed by Gratton et al. in 2008, which consists of a different formulation of the trust-region iteration that includes different possibilities for the computation of the model to be minimized inside the trust region, and the computation of the trial point. In a way, we apply the trust-region method *at each level*. This means that the basic trust-region iteration has to be modified and thus that the convergence theory for this method differs from the convergence theory of a basic trust-region algorithm. We will present here the convergence theory developed for the ℓ_∞-norm version of RMTR.

Finally, we will present a third application of multilevel techniques, this time for derivative-free optimization problems. In this class of problems, the derivatives of the objective function are not available, either because they are too expensive to compute or because they are simply not known analytically. There are several methods capable of solving these problems, and we will focus once more on a trust-region approach, where the quadratic model to be minimized at each trust-region iteration is computed using an interpolation technique. We will present a brief introduction to this method, and a new methodology for problems in which a multilevel strategy is possible.

This work is organized as follows. In Chapter 1, we will present a brief introduction to optimization, with some basic definitions and results necessary for the rest of this work. In Chapter 2, we will present the basic ideas behind multigrid methods. In Chapter 3, we will present the Multilevel Moré-Sorensen method, a multilevel strategy for the solution of the trust-region subproblem, along with some computational results. In Chapter 4, we will describe the RMTR family of methods and the convergence theory of the ℓ_∞-norm version of the method. Finally, in Chapter 5, we will describe in details a new multilevel method for derivative-free optimization.

Chapter 1

Optimization

Optimization or *mathematical programming* is concerned with minimizing or maximizing some quantity, represented by an *objective function*. Often, the desired result must lie in a certain subset of the domain of this function. The particular characteristics of the function and the subset that must contain the solution to the problem define several different types of optimization problems, each requiring a different method to be solved.

In this chapter, we will present the formal definitions of optimization and mathematical programming problems and some of the methods, as well as basic definitions from Analysis and Linear Algebra that will be needed throughout this thesis.

1.1 The problem

Optimization problems can be divided into two large classes, namely *Constrained* and *Unconstrained* problems. The basic unconstrained optimization problem can be stated in its standard form as

$$\text{minimize } f(x), \text{ subject to } x \in \mathbb{R}^n, \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the *objective function*. On the other hand, constrained optimization problems can be written as

$$\begin{aligned} \text{minimize} \quad & f(x), && \text{(1.2a)}\\ \text{subject to} \quad & x \in X \subseteq \mathbb{R}^n, && \text{(1.2b)}\\ & g_i(x) \leq 0, i \in \mathcal{I} && \text{(1.2c)}\\ & g_i(x) = 0, i \in \mathcal{E}. && \text{(1.2d)} \end{aligned}$$

Equations (1.2b-1.2d) indicate the *constraints*. The disjoint index sets $\mathcal{I}$ and $\mathcal{E}$ correspond to the inequality and equality constraints, respectively, defined by the functions $g_i : \mathbb{R}^n \to \mathbb{R}, i \in \mathcal{I} \cup \mathcal{E}$. The set X is contained in $\mathbb{R}^n$ and is also contained in the domain of f and $g_i, i \in \mathcal{I} \cup \mathcal{E}$. A point $x \in X$ is said to be *feasible* if it satisfies all the constraints, and the set of all feasible points is called the *feasible set*, and denoted by $\mathcal{F}$.

The formulations (1.1) and (1.2) are called standard formulations due to the observation that

$$\max f(x) = -\min(-f(x)).$$

Thus, any maximization problem can be rewritten as a minimization problem, falling into one of these two formulations. Because of this, all problems in this thesis will be described as minimization problems.

When the constraints can be written as

$$l_i \leq x_i \leq u_i,\ i = 1, \ldots, n,$$

where x_i denotes the i-th component of the vector $x \in \mathbb{R}^n$ and l_i and u_i, $i = 1, \ldots, n$, are real values, then we refer to them as *bound constraints*. The values l_i and u_i, for $i = 1, \ldots, n$ are called *lower* and *upper* bounds, respectively. Since this type of constraints are very specific and can be treated differently from the general inequality constraints, they are usually formulated separately or incorporated into the definition of X.

Some problems differ from others in such a significant way that the methods to solve them have to be fundamentally different, and thus require a new classification. This is the case, for instance, with problems where f is a linear function. In this case, we refer to the problem as a *Linear Programming Problem*. If the function is not necessarily linear, then we refer to the problem as a general *Nonlinear Programming Problem*.

In this thesis, we will pursue methods to solve general unconstrained nonlinear programming problems.

1.2 The solution

The solution of an optimization problem can be characterized by certain properties. In a minimization problem, if we are looking for a point x^* in the domain $\mathcal{D}$ of f such that

$$f(x^*) \leq f(x), \text{ for all } x \in \mathcal{D},$$

then x^* is called the *global minimizer* and $f(x^*)$ the *global minimum* of f. Similarly, in a constrained problem, the solution must lie inside the feasible set $\mathcal{F}$, and thus a global constrained minimizer satisfies

$$f(x^*) \leq f(x), \text{ for all } x \in \mathcal{F}.$$

However, in both cases, finding a global minimizer of a function f can prove to be very difficult in practice. It might be interesting, thus, to look for a solution x^* in a neighborhood $\mathcal{N}$ of x^*, $\mathcal{N} \subseteq \mathcal{D}$ such that

$$f(x^*) \leq f(x), \text{ for all (feasible) } x \in \mathcal{N}. \tag{1.3}$$

The point x^* is then called a *local minimizer* and $f(x^*)$ a *local minimum* of f in $\mathcal{N}$. If x^* is such that

$$f(x^*) < f(x), \text{ for all (feasible) } x \in \mathcal{N}. \tag{1.4}$$

then x^* is said to be a *strict local minimizer* and $f(x^*)$ a *strict local minimum* of f in $\mathcal{N}$. Figure 1.1 shows an example of a function with two distinct local minimizers, and Figure 1.2 shows examples of functions with non-strict and strict local minimizers.

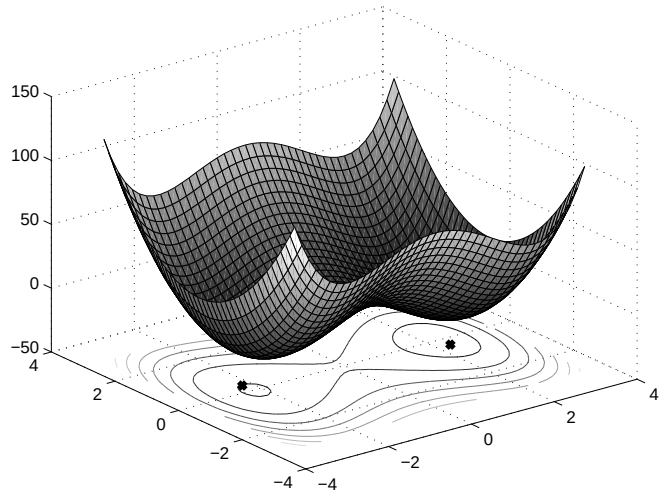
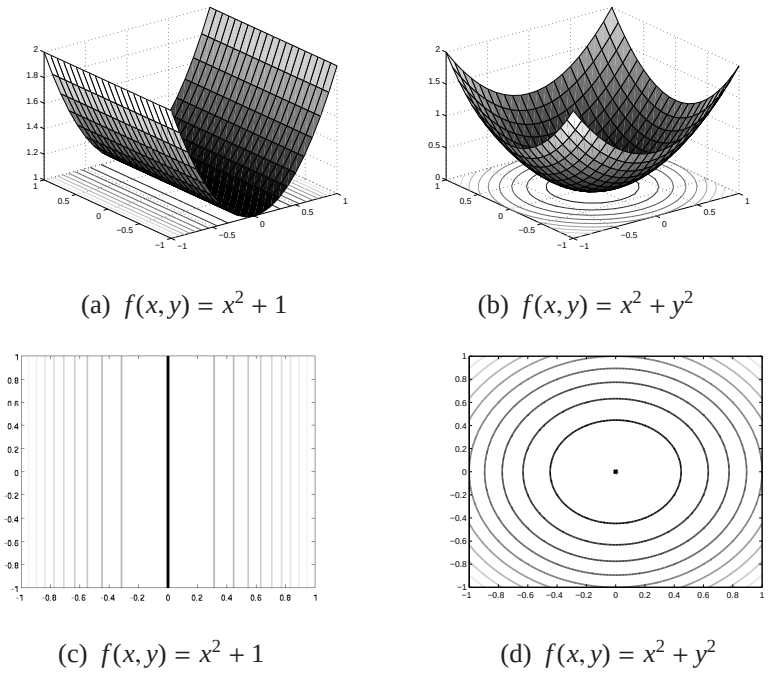


Figure 1.1: Global and local minima of the function $f(x,y) = -10x^2 + 10y^2 + 4\sin xy - 2x + x^4$. This function admits two local minima, at points $(x_1, y_1) = (-2.21, 0.32)$ and $(x_2, y_2) = (2.30, -0.33)$, as indicated in the picture. The global minimum, however, is found at point (x_2, y_2), since $f(x_1, y_1) \approx -22.14$ and $f(x_2, y_2) \approx -31.18$.



(a) $f(x,y) = x^2 + 1$ (b) $f(x,y) = x^2 + y^2$

(c) $f(x,y) = x^2 + 1$ (d) $f(x,y) = x^2 + y^2$

Figure 1.2: In (a) and (b), we see the surface plots of the functions $f(x,y) = x^2 + 1$ and $f(x,y) = x^2 + y^2$, respectively, and in (c) and (d) we see the level curves for the same functions. On the left, we see an example of a local minimizer which is not strict. In this case, any point in the line $(0, y)$ (for all $y \in \mathbb{R}$) is a local minimizer for $f(x,y) = x^2 + 1$. On the right, the point $(x,y) = (0,0)$ is a strict local minimizer for $f(x,y) = x^2 + y^2$.

In general, we would like to be able to check if a point is a solution to the problem being solved. In order to do this, there are some *optimality conditions*, which allow us to determine if we are at the solution or not.

First, let us state a few definitions which will be useful for our analysis throughout this work.

1.2.1 Basic Theoretical Concepts

Since we will be dealing mostly with problems defined in a multivariate space $\mathbb{R}^n$, some definitions are useful here.

As already mentioned, if v is a vector in $\mathbb{R}^n$, we use v_i to denote the i-th component of v, $i = 1, \ldots, n$ (except when indicated otherwise). We denote by $e \in \mathbb{R}^n$ the vector composed of ones in every component, while $e^{[i]} \in \mathbb{R}^n$ is the i-th *coordinate vector* i n $\mathbb{R}^n$ with zeros in every component but 1 in the i-th one.

1.2.1.1 Eigenvalues and Eigenvectors

Given a matrix $A \in \mathbb{R}^{n \times n}$, the nonlinear equation

$$Au = \lambda u$$

defines solution pairs (u, λ) which are called *eigenpairs*. The scalar value λ is called an *eigenvalue* and the vector $u \in \mathbb{R}^n$ is called an *eigenvector*. The matrix A can have up to n eigenpairs, and the set of all eigenvalues is called the *spectrum* of A. The *spectral radius* of A is denoted by $\rho(A)$ and is defined as the largest eigenvalue of A in absolute value, that is,

$$\rho(A) \stackrel{\text{def}}{=} \max\{|\lambda| \mid \lambda \text{ is an eigenvalue of } A\}.$$

Symmetric matrices are a special class of matrices, since all eigenvalues of a symmetric matrix are real, and it is possible to find a complete set of n orthonormal eigenvectors associated with such a matrix. For any symmetric matrix S, we can write

$$S = U \Lambda U^T,$$

where U^T denotes the transpose of matrix U, and where the entries of the diagonal matrix Λ are the eigenvalues of S, and the columns of the orthonormal matrix U are the associated eigenvectors.

For any matrix $A \in \mathbb{R}^{m \times n}$, the *singular value decomposition* is a factorization of the form

$$A = U \Sigma V^T,$$

where U and V are, respectively, m by m and n by n orthogonal matrices, and $\Sigma \in \mathbb{R}^{m \times n}$ is a diagonal matrix with nonnegative entries which are called the *singular values* of A. The columns of U and V are known as the left and right *singular vectors* of A. The squares of the nonzero singular values of A are the nonzero eigenvalues of AA^T (which are equal to the eigenvalues of $A^T A$), while the left and right singular vectors are eigenvectors of AA^T and $A^T A$, respectively.

From the singular value decomposition of a matrix A, we can obtain its *rank*, which is the number of its nonzero singular values. The rank of A, denoted by rank(A) is also the number of linearly independent rows (and columns) of A. The matrix is said to be of *full rank* if rank(A) = min(m, n).

For any vectors $u, v \in \mathbb{R}^n$, we define the *inner product* of u and v as

$$\langle u, v \rangle = \sum_{i=1}^{n} u_i v_i. \tag{1.5}$$

Then, a symmetric matrix $S \in \mathbb{R}^{n \times n}$ is *positive semidefinite* if and only if

$$\langle x, Sx \rangle \geq 0, \qquad \text{for all } x \in \mathbb{R}^n.$$

S is said to be *positive definite* if and only if

$$\langle x, Sx \rangle > 0, \qquad \text{for all } x \in \mathbb{R}^n, x \neq 0. \tag{1.6}$$

If S is positive semidefinite, we can define

$$\sqrt{S} \stackrel{\text{def}}{=} U \sqrt{\Lambda} U^T,$$

where $\sqrt{\Lambda}$ is a diagonal matrix with $\Lambda_{ii} = \sqrt{\lambda_i(S)}$, and where $\lambda_i(S)$ denotes the i-th eigenvalue of S.

Now, if S is symmetric and $x \neq 0$, the scalar

$$\frac{\langle x, Sx \rangle}{\langle x, x \rangle}$$

is known as the *Rayleigh quotient* of x. This is an important quantity, as we can estimate the largest and smallest eigenvalues of S from it:

$$\lambda_{\min}(S) \leq \frac{\langle x, Sx \rangle}{\langle x, x \rangle} \leq \lambda_{\max}(S) \tag{1.7}$$

for any $x \neq 0$, where $\lambda_{\min}(S)$ denotes the smallest eigenvalue of S, and $\lambda_{\max}(S)$ denotes the largest eigenvalue of S. This inequality is known as the *Rayleigh quotient inequality*, and the quotient attains its maximum and minimum values when x is the eigenvector associated with the largest or smallest eigenvalue, respectively.

Since it is usually very expensive to compute the eigenvalues of a general matrix A, it is useful to have bounds on these values. The *Gershgorin bounds* are very well known. For real symmetric matrices, they state that

$$\min_i \left(s_{i,i} - \sum_{j \neq i} |s_{i,j}| \right) \leq \lambda_{\min}(S) \leq \lambda_{\max}(S) \leq \max_i \left(s_{i,i} + \sum_{j \neq i} |s_{i,j}| \right), \tag{1.8}$$

where $s_{i,j}$, $i, j = 1, \ldots, n$ is the element located in the i-th row and j-th column of S.

1.2.1.2 Vector Norms and Properties

Given a vector

$$v = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} \in \mathbb{R}^{n\times 1},$$

we denote by $v^T \in \mathbb{R}^{1\times n}$ its transpose

$$v^T = (v_1\, v_2\, \cdots\, v_n)\,.$$

Note that we can also write the inner product (1.5) as $\langle u, v\rangle = u^T v$.

A *vector norm* on $\mathbb{R}^n$ is a function $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ that satisfies the following properties:

(i) $\|v\| \geq 0, \qquad v \in \mathbb{R}^n$;

(ii) $\|v\| = 0$ if and only if $v = 0$;

(iii) $\|v + u\| \leq \|v\| + \|u\|$ for all $v, u \in \mathbb{R}^n$;

(iv) $\|\alpha v\| = |\alpha|\|v\|$ for all $\alpha \in \mathbb{R}$ and for all $v \in \mathbb{R}^n$.

The most important class of norms for our work will be the class of *p-norms* (or ℓ_p*-norms*), defined by

$$\|v\|_p = (|v_1|^p + |v_2|^p + \ldots + |v_n|^p)^{\frac{1}{p}}\,, p \geq 1.$$

In this class, three norms are very important:

$$\|v\|_1 = |v_1| + |v_2| + \ldots + |v_n| \tag{1.9}$$

$$\|v\|_2 = \left(|v_1|^2 + |v_2|^2 + \ldots + |v_n|^2\right)^{\frac{1}{2}} = \sqrt{v^T v} \tag{1.10}$$

$$\|v\|_\infty = \max_{1\leq i\leq n} |v_i|. \tag{1.11}$$

The ℓ_2 and ℓ_∞ norms are also called *Euclidean norm* and *Maximum norm*, respectively.

The *Holder inequality* is a classic result for p norms that states that

$$|\langle x, y\rangle| \leq \|x\|_p \|y\|_q\,, \qquad \frac{1}{p} + \frac{1}{q} = 1.$$

A special case of this inequality is the *Cauchy-Schwartz inequality*:

$$|\langle x, y\rangle| \leq \|x\|_2 \|y\|_2.$$

In the space $\mathbb{R}^n$, all norms are *equivalent*, which means that if $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$ are norms in $\mathbb{R}^n$, then there exist constants $c_1, c_2 > 0$ such that

$$c_1\|x\|_\alpha \leq \|x\|_\beta \leq c_2\|x\|_\alpha, \text{ for all } x \in \mathbb{R}^n.$$

For example, for all $x \in \mathbb{R}^n$,

$$\|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\|x\|_2 \tag{1.12a}$$

$$\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty \tag{1.12b}$$

$$\|x\|_\infty \leq \|x\|_1 \leq n\|x\|_\infty. \tag{1.12c}$$

1.2.1.3 Matrix Norms

Given a vector norm $\|\cdot\|$, for any matrix $A \in \mathbb{R}^{m\times n}$, we define a matrix norm *induced* by its vector counterpart by

$$\|A\| = \max_{x\neq 0} \frac{\|Ax\|}{\|x\|}.$$

The most frequently used matrix norms in this category are the matrix ℓ_1, ℓ_2 and ℓ_∞ norms. They can also be written as

$$\|A\|_1 = \max_{1\leq i\leq n} \|Ae^{[i]}\|_1 \tag{1.13}$$

$$\|A\|_2 = \sigma_{\max}(A) \tag{1.14}$$

$$\|A\|_\infty = \max_{1\leq i\leq n} \|A^T e^{[i]}\|_1 \tag{1.15}$$

for $A \in \mathbb{R}^{m\times n}$, where $\sigma_{\max}(A)$ denotes the largest singular value of A.

There is another norm that will be useful for what follows. It is called the *Frobenius* or Euclidean matrix norm, and it is defined by

$$\|A\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{i,j}^2}.$$

As in the case of vector norms, we can show the equivalence of these norms in the following inequalities.

$$\|A\|_2 \leq \|A\|_{\mathrm{F}} \leq \sqrt{n}\|A\|_2. \tag{1.16a}$$

$$\frac{1}{\sqrt{n}}\|A\|_\infty \leq \|A\|_2 \leq \sqrt{m}\|A\|_\infty \tag{1.16b}$$

$$\frac{1}{\sqrt{m}}\|A\|_1 \leq \|A\|_2 \leq \sqrt{n}\|A\|_1. \tag{1.16c}$$

Another useful property is that

$$\|A\|_2 \leq \sqrt{\|A\|_1\|A\|_\infty},$$

which has as a consequence the fact that, if A is symmetric, then bounds (1.16) become

$$\|A\|_2 \leq \|A\|_1 \equiv \|A\|_\infty. \tag{1.17}$$

We say that a matrix norm $\|\cdot\|_{uv}$ is *consistent* with a vector norm $\|\cdot\|_u$ and a vector norm $\|\cdot\|_v$ if

$$\|Ax\|_u \leq \|A\|_{uv}\|x\|_v$$

for all $A \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^n$.

For any matrix $A \in \mathbb{R}^{n\times n}$, we can define the A-norm of $x \in \mathbb{R}^n$ as

$$\|x\|_A = \sqrt{\langle x, Ax\rangle}.$$

The *condition number* $\kappa(A)$ of a matrix A is defined as

$$\kappa(A) = \|A\| \, \|A^{-1}\|.$$

We say that the matrix A is *ill-conditioned* if this number is large, otherwise we say that A is *well-conditioned*. This number depends on the norm chosen; for example, if we choose the ℓ_2 norm, then

$$\kappa_2(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)},$$

where $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ are the largest and smallest singular value of A, respectively.

1.2.1.4 Matrix Factorizations

When trying to solve a linear system

$$Ax = b,$$

where $A \in \mathbb{R}^{n \times n}$ and $x, b \in \mathbb{R}^n$, it is possible to *factorize* A in order to obtain simpler systems to solve. One example of such a factorization is the *LU factorization*,

$$A = LU,$$

where L is a lower triangular matrix with 1s in the diagonal, and U is an upper triangular matrix. This factorization exists if the determinants of all submatrices $A_{(k)}$, obtained by deleting all rows i and columns j of A where $i, j > k$, are nonzero. Furthermore, if A is nonsingular and its LU factorization exists, then it is unique and $\det(A) = u_{11} \cdots u_{nn}$, where $u_{ii}, i = 1, \ldots, n$, are the diagonal elements of U.

This factorization is very important since it allows us to solve a dense linear system by solving two triangular systems, which are much simpler. The idea is to solve

$$\begin{cases} Ly &= b \\ Ux &= y \end{cases}$$

in sequence in order to obtain x.

When the matrix in question is symmetric and positive definite, a special factorization called *Cholesky factorization* can be computed. In this case, we can find a unique lower triangular $L \in \mathbb{R}^{n \times n}$ with positive diagonal entries such that

$$A = LL^T. \tag{1.18}$$

In this case, the system can also be solved by successively solving

$$\begin{cases} Ly &= b \\ L^T x &= y \end{cases}$$

The Cholesky factorization has another advantage, in that it allows us to check for positive definite matrices, since the usual definition (1.6) is very hard to test for in practice. If the matrix is not positive definite, the Cholesky factorization will fail, as a L with positive diagonal entries cannot be computed.

1.2.1.5 Derivatives and Taylor's Theorem

All methods considered here are based on the fact that the function to be minimized has derivatives. In fact, even if the derivatives are not known, the mere fact that these derivatives could be computed changes dramatically the approach used in solving the problem.

First, consider a real function $f : \mathbb{R} \to \mathbb{R}$. Then, f is *differentiable* at x if the limit

$$\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

exists. If this is the case, then

$$f'(x) = \frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

is called the *derivative* of f at x.

Now, let $f : \mathbb{R}^n \to \mathbb{R}$ be a multivariate function. Then, it is said to be *differentiable* at $x \in \mathbb{R}^n$ if all its *partial derivatives*

$$\frac{\partial f(x)}{\partial x_i} = \lim_{h \to 0} \frac{f(x + he^{[i]}) - f(x)}{h}, \; i = 1, \ldots, n$$

exist, where $e^{[i]}$ is the i-th coordinate vector in $\mathbb{R}^n$. If this is the case, then we define the *gradient* of f as the vector that groups all its partial derivatives, and we denote it by

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{pmatrix}$$

If f is differentiable, and all derivatives of f are continuous with respect to x, then we say that f is *continuously differentiable*, and this is denoted by $f \in C^1$.

The second partial derivatives of f are defined by[(1)]

$$\frac{\partial^2 f(x)}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_i} \left(\frac{\partial f(x)}{\partial x_j} \right), \; 1 \leq i, j \leq n.$$

If all second partial derivatives of f exist, then f is said to be *twice differentiable*; if, furthermore, all second partial derivatives of f are continuous, we say that f is *twice continuously differentiable*, and denote this by $f \in C^2$.

The $n \times n$ matrix defined as

$$\nabla^2 f(x) = \begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{pmatrix}$$

is called the *Hessian matrix* of f. The *curvature* of f at $x \in \mathbb{R}^n$ along a direction $d \in \mathbb{R}^n$ is given by

$$\frac{\langle d, \nabla^2 f(x) d \rangle}{\|d\|^2}.$$

[(1)] We will use the notation $\frac{\partial^2 f(x)}{\partial x_i \partial x_i} = \frac{\partial^2 f(x)}{\partial x_i^2}$, $1 \leq i \leq n$.

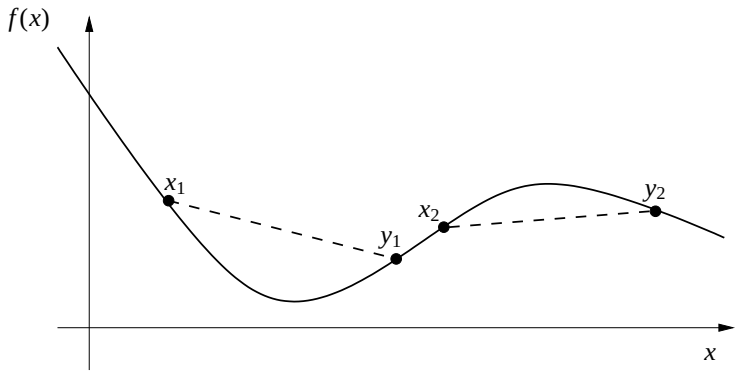


Figure 1.3: Here, the function is convex when restricted to the interval $[x_1, y_1]$, but it is not convex when restricted to the interval $[x_2, y_2]$.

If the domain $\mathcal{D}$ of the function f is a convex set[(2)], then f is said to be *convex* if

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y) \tag{1.19}$$

for all $x, y \in \mathcal{D}$ and $\theta \in [0,1]$. The function f is *strictly convex* if the inequality in (1.19) is strict for all $x, y \in \mathcal{D}$ such that $x \neq y$ and for $\theta \in (0,1)$. In other words, a function is convex if it always lies below its linear interpolant. Figure 1.3 shows an example of this.

Another definition is that if $\nabla^2 f(x)$ is positive semidefinite for every x in the domain of f, we say that f is convex. If $\nabla^2 f(x)$ is positive definite in its domain, we say that it is strictly convex. Figure 1.4(a) shows the surface plot of the function $f(x,y) = x^2 + 1$, which is a convex function, but not strictly convex. Its Hessian

$$\nabla^2 f(x,y) = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}$$

is positive semidefinite. Figure 1.4(b) shows the surface plot of the function $f(x,y) = x^2 + y^2$, which is strictly convex, since its Hessian

$$\nabla^2 f(x,y) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

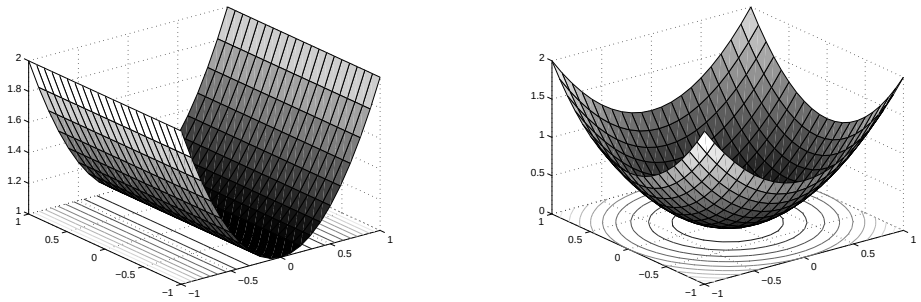
is positive definite for all $(x,y) \in \mathbb{R}^2$.

If the function we are interested in depends on several vectors, for instance if we consider a function

$$\begin{aligned} h : \mathbb{R}^n \times \mathbb{R}^q &\to \mathbb{R} \\ (x,y) &\mapsto h(x,y), \end{aligned}$$

then we will use the following notation: $\nabla_x h(x,y) \in \mathbb{R}^n$ and $\nabla^2_{xx} h(x,y) \in \mathbb{R}^{n \times n}$ will denote the gradient and the Hessian matrix of h with respect to x; $\nabla_y h(x,y) \in \mathbb{R}^q$ and $\nabla^2_{yy} h(x,y) \in \mathbb{R}^{q \times q}$ will denote the gradient and the Hessian matrix of h with respect to y; and $\nabla h(x,y) \in \mathbb{R}^{n+q}$ and

[(2)] A subset $\mathcal{D} \subseteq \mathbb{R}^n$ is convex if for any points $x, y \in \mathcal{D}$ and $\theta \in [0,1]$, the point $x + \theta(y - x) \in \mathcal{D}$.



(a) $f(x,y) = x^2 + 1$ (b) $f(x,y) = x^2 + y^2$

Figure 1.4: Examples of convex and strictly convex functions.

$\nabla^2 h(x,y) \in \mathbb{R}^{(n+q)\times(n+q)}$ will denote the complete first- and second-order derivatives of h with respect to both x and y. This notation will be used only when necessary.

With these definitions, we can state one of the most important results in the theory of optimization:

Theorem 1.2.1 (Taylor's Theorem.) *Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable and $s \in \mathbb{R}^n$. Then, there exists some $t \in (0,1)$ such that*

$$f(x+s) = f(x) + \langle \nabla f(x+ts), s \rangle.$$

Moreover, if f is twice continuously differentiable, then

$$\nabla f(x+s) = \nabla f(x) + \int_0^1 \nabla^2 f(x+ts)s\, dt.$$

Furthermore, we have that, for some $t \in (0,1)$,

$$f(x+s) = f(x) + \langle \nabla f(x), s \rangle + \frac{1}{2}\langle s, \nabla^2 f(x+ts)s \rangle. \tag{1.20}$$

Taylor's theorem is the main result we will use throughout this thesis to derive the *model* used in trust-region methods, and is the justification for the theorems in the next section, which characterize a local minimizer of a function. The proof can be found in Conn et al. [12], for example.

A function $F : \mathbb{R}^n \to \mathbb{R}^m$ is said to be differentiable if all its partial derivatives

$$\frac{\partial F_j(x)}{\partial x_i} = \lim_{h\to 0} \frac{F_j(x + he^{[i]}) - F_j(x)}{h}, \quad i = 1,\ldots,n, \quad j = 1,\ldots,m$$

exist. In this case, we can write all partial derivatives of F in the $m \times n$ matrix

$$JF(x) = \begin{pmatrix} \nabla F_1(x) \\ \vdots \\ \nabla F_m(x) \end{pmatrix} = \begin{pmatrix} \frac{\partial F_1(x)}{\partial x_1} & \frac{\partial F_1(x)}{\partial x_2} & \cdots & \frac{\partial F_1(x)}{\partial x_n} \\ \frac{\partial F_2(x)}{\partial x_1} & \frac{\partial F_2(x)}{\partial x_2} & \cdots & \frac{\partial F_2(x)}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_m(x)}{\partial x_1} & \frac{\partial F_m(x)}{\partial x_2} & \cdots & \frac{\partial F_m(x)}{\partial x_n} \end{pmatrix}.$$

This matrix is called the *Jacobian matrix* of F.

Another definition will be useful in the discussion that follows. Let $F : \mathbb{R}^m \to \mathbb{R}^n$, and suppose that $\|\cdot\|_{[m]}$ and $\|\cdot\|_{[n]}$ are norms defined in $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively. Then we say that F is *Lipschitz continuous* at $x \in \mathbb{R}^m$ if there is a $\gamma(x) \geq 0$ such that

$$\|F(y) - F(x)\|_{[n]} \leq \gamma(x)\|y - x\|_{[m]}, \quad \text{for all } y \in \mathbb{R}^m.$$

1.2.2 Solutions of Unconstrained Problems

Now, we can present the main concepts and results that allow us to identify the solution to an optimization problem. First, we look at unconstrained strict local minimizers as defined in (1.4).

Theorem 1.2.2 (First-Order Necessary Conditions) *If x^* is a local minimizer of $f : \mathbb{R}^n \to \mathbb{R}$, where f is continuously differentiable in an open neighborhood $\mathcal{N}$ of x^*, then*

$$\nabla f(x^*) = 0. \tag{1.21}$$

If $\nabla^2 f$ exists and is continuous in a neighborhood of x^*, we can state another necessary condition satisfied by a local minimizer.

Theorem 1.2.3 (Second-Order Necessary Conditions) *If x^* is a local minimizer of f, and f is twice continuously differentiable in an open neighborhood $\mathcal{N}$ of x^*, then*

$$\nabla f(x^*) = 0 \ \text{ and } \ \nabla^2 f(x^*) \text{ is positive semidefinite.} \tag{1.22}$$

Any x^* that satisfies (1.21) is called a *stationary point* of f. Thus, Theorem 1.2.2 states that any local minimizer must be a stationary point; it is important to note, however, that the opposite is not necessarily true. Fortunately, if the next conditions, called *sufficient conditions*, are satisfied by a stationary point x^*, they guarantee that it is a local minimizer.

Theorem 1.2.4 (Second-Order Sufficient Conditions) *Let f be twice continuously differentiable on an open neighborhood $\mathcal{N}$ of x^*. If x^* satisfies*

$$\nabla f(x^*) = 0 \ \text{ and } \ \nabla^2 f(x^*) \text{ is positive definite,}$$

then x^* is a strict local minimizer of f.

It is important to note here that the second-order sufficient conditions are not necessary: a point can be a strict local minimizer and fail to satisfy the sufficient conditions.

1.2.3 Constrained Problems

Finally, let us state the main results that allow us to characterize a constrained strict local minimizer of a function f. To do this, we must define some additional concepts related to constrained problems.

We say that an inequality constraint g_j, for some index $j \in \mathcal{I}$ is *active* at a feasible point x if $g_j(x) = 0$. At any feasible point x, the *active set* $\mathcal{A}(x)$ is the union of the set $\mathcal{E}$ with the index set of the active inequality constraints at x, that is

$$\mathcal{A}(x) = \mathcal{E} \cup \{ j \in \mathcal{I} \text{ such that } g_j(x) = 0\}.$$

The *Lagrangian* (or *Lagrange function*) for problem (1.2) is defined by

$$\mathcal{L}(x, \lambda) = f(x) + \sum_{i \in \mathcal{I} \cup \mathcal{E}} \lambda_i g_i(x),$$

and the scalars λ_i ($i \in \mathcal{I} \cup \mathcal{E}$) are called the *Lagrange multipliers*.

In constrained optimization, a solution is not only characterized by conditions on the objective function at the solution, but also by conditions on the constraints. These conditions are called *constraint qualifications* and they ensure that we exclude pathological cases in the geometry of the constraints at the solution. Here we present only one of them, but many others have been proposed. For a more complete investigation, see Nocedal and Wright [50], for example.

Condition 1.1 Linear Independence Constraint Qualification (LICQ): *Given a point x^* and the corresponding active set $\mathcal{A}(x^*)$, the* linear independence constraint qualification *is said to hold for problem (1.2) at x^* if the active gradients*

$$\{\nabla g_i(x^*) \mid i \in \mathcal{A}(x^*)\}$$

are linearly independent.

This is equivalent to the requirement that the Jacobian of the active constraints at x^* has full row rank.

Now we can state the optimality conditions for constrained problems, starting by the necessary optimality conditions.

Theorem 1.2.5 (First Order Necessary Conditions) *Assume that x^* is a local solution of the constrained optimization problem (1.2) and that the LICQ Condition 1.1 holds at x^*. Then there exists a Lagrange multiplier vector λ^*, with components $\lambda_i^* (i \in \mathcal{E} \cup \mathcal{I})$ such that the following conditions are satisfied at (x^*, λ^*):*

$$\nabla_x \mathcal{L}(x^*, \lambda^*) = 0, \tag{1.23a}$$
$$g_i(x^*) = 0 \quad \text{for all} \quad i \in \mathcal{E}, \tag{1.23b}$$
$$g_i(x^*) \leq 0 \quad \text{for all} \quad i \in \mathcal{I}, \tag{1.23c}$$
$$\lambda_i^* \geq 0 \quad \text{for all} \quad i \in \mathcal{I}, \tag{1.23d}$$
$$\lambda_i^* g_i(x^*) = 0 \quad \text{for all} \quad i \in \mathcal{I}. \tag{1.23e}$$

Conditions (1.23) are known as the *Karush-Kuhn-Tucker (KKT) conditions*. They were introduced first by Karush [34] and rediscovered later by Kuhn and Tucker [38]. Equation (1.23a) is called the *stationarity condition*, (1.23b) and (1.23c) are called the *feasibility conditions*, (1.23d) states the *non-negativity of the multipliers* and (1.23e) is the *complementarity condition*. A point x^* satisfying (1.23) is called a *first-order critical point* or *KKT point* for problem (1.2). Since the *complementarity condition* (1.23e) implies that the Lagrange multipliers associated with inactive inequality constraints are zero, we can rewrite condition (1.23a) as

$$\nabla f(x^*) + \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* g_i(x^*) = 0. \tag{1.24}$$

In order to state the second-order conditions for constrained problems, we define the *critical cone* $\mathcal{N}_+(x^*, \lambda^*)$ as

$$\mathcal{N}_+(x^*, \lambda^*) \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} w \in \mathbb{R}^n \mid & \langle \nabla g_i(x^*), w \rangle = 0, \text{ for all } i \in \mathcal{E} \cup \left(\mathcal{A}(x^*) \cap \mathcal{I} \text{ with } \lambda_i^* > 0 \right), \\ \text{and} & \langle \nabla g_i(x^*), w \rangle \geq 0, \text{ for all } i \in \mathcal{A}(x^*) \cap \mathcal{I} \text{ with } \lambda_i^* = 0. \end{array} \right\}$$

First, let us state the necessary conditions.

Theorem 1.2.6 (Second-Order Necessary Conditions) *Suppose that x^* is a local solution of problem (1.2) and that the LICQ Condition 1.1 is satisfied. Let λ^* be the Lagrange multiplier vector for which the KKT conditions (1.23) are satisfied. Then,*

$$\langle w, \nabla^2_{xx} \mathcal{L}(x^*, \lambda^*) w \rangle \geq 0, \quad \text{for all } w \in \mathcal{N}_+(x^*, \lambda^*). \tag{1.25}$$

A point x^* that satisfies (1.25) is called a *strong second-order critical point* for problem (1.2). A *sufficient* condition, like the one derived for unconstrained problems, can be stated as follows.

Theorem 1.2.7 (Second-Order Sufficient Conditions) *Assume that for some feasible point $x^* \in \mathbb{R}^n$ there exists a Lagrange multiplier vector λ^* such that the KKT conditions (1.23) are satisfied. Assume further that*

$$\langle w, \nabla^2_{xx} \mathcal{L}(x^*, \lambda^*) w \rangle > 0, \quad \text{for all } w \in \mathcal{N}_+(x^*, \lambda^*) \text{ with } w \neq 0.$$

Then, x^* is a strict local solution for problem (1.2).

1.3 Solving the problem

Mathematical programming is concerned with the methods that can be used to solve optimization problems. In practice, we will be concerned with algorithms, defined so that their computational implementation finds either an approximate or an exact solution to the original mathematical programming problem.

These algorithms are mostly *iterative methods*, which from a starting point x_0, use some rule to compute a sequence of points $\{x_1, x_2, \ldots, x_k, \ldots\}$ such that

$$\lim_{k \to \infty} x_k = x^*,$$

where x^* is the solution to this problem.

Here, we are mostly interested in unconstrained problems. Thus, Theorems 1.2.2, 1.2.3 and 1.2.4 will be the basis of the methods to solve the minimization problem, as they will be based in this characterization of the solution whenever possible.

As already mentioned in the Introduction, the simplest method developed for the solution of a minimization problem is the steepest descent method. This method is based on the fact that, from any starting point x, the direction in which any function f decreases most rapidly, at least locally, is the direction $-\nabla f(x)$[3]. However, the steepest descent method can be extremely slow for some problems. There are, fortunately, several other methods that work very well in practice, and here we briefly present some of them.

1.3.1 Newton's Method

Consider any nonlinear system of equations of the form

$$F(x) = 0, \tag{1.26}$$

where $F : \mathbb{R}^n \to \mathbb{R}^n$. If the Jacobian of F exists, then we can write the Taylor first-order approximation to this function as

$$F(x + s) \approx F(x) + J(x)s, \tag{1.27}$$

where $J(x)$ denotes the Jacobian of F evaluated at x. From these equations, we can derive an iterative method. Given an initial point x_0, at each iteration k, we will compute a new iterate $x_{k+1} = x_k + s_k$ such that $F(x_k + s_k) = 0$, which means that s_k must satisfy the linear system

$$J(x_k)s_k = -F(x_k).$$

This is called the *Newton's method* for solving nonlinear systems of equations.

Now, returning to our optimization problem, note that when the first and second derivatives of f are available, we can use Newton's method to solve the (possibly nonlinear) system of equations defined by

$$\nabla f(x) = 0, \tag{1.28}$$

since we know from Theorem 1.2.2 that any minimizer of f must satisfy this condition. This is the basis for the Newton's method for optimization problems and, with some variations, it is the basis for many other methods in unconstrained optimization. More formally, if we want to apply this method to equation (1.28), observe that the second-order Taylor's approximation to f at x_k is

$$f(x_k + s_k) \approx f(x_k) + \langle \nabla f(x_k), s_k \rangle + \frac{1}{2}\langle s_k, \nabla^2 f(x_k)s_k \rangle. \tag{1.29}$$

In order to find a minimum of this function, we will try to find a solution to $\nabla f(x_k + s_k) = 0$, which is equivalent to

$$\nabla f(x_k) + \nabla^2 f(x_k)s_k = 0.$$

[3] This is a direct consequence of Theorem 1.2.1. For more details, see Nocedal and Wright [50].

Thus, we have that s_k must satisfy the so called *Newton equations*

$$\nabla^2 f(x_k) s_k = -\nabla f(x_k). \tag{1.30}$$

If $\nabla^2 f(x_k)$ is positive definite, then we can find its inverse, and the solution to (1.30) is

$$s_k = -\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k). \tag{1.31}$$

This direction s_k is called the *Newton direction*.

One of the problems in Newton's method is that the method is based on a necessary first-order optimality condition (namely, that the gradient of the objective function be equal to zero). In order to guarantee that we have found a minimizer of f, it suffices to guarantee that the Hessian $\nabla^2 f(x^*)$ be positive definite. Moreover, the approximations (1.27) and (1.29) are only valid in a neighborhood of the solution of (1.26) and (1.1), respectively. Thus, Newton's method is only appropriate when the starting point x_0 is *sufficiently close* to the solution x^*. However, when it works, it is very fast and most optimization methods try to mimic its behavior around the solution.

There are so called *globalization techniques* that can be used to guarantee the convergence of Newton's method to a sationary point x^* from any starting point. These techniques give rise to different methods which can be divided into two classes: *Line Search Methods* and *Trust-Region Methods*. The main difference between these two classes is that in Line Search methods, the direction in which we choose to take our next iterate is selected first, while the size of the step to be taken in this direction is computed with the direction fixed. On the other hand, in Trust-Region methods, the step size and the direction are more or less chosen simultaneously. We describe both strategies in more detail in the following sections.

1.3.2 Line Search Methods

As their name suggests, the idea behind Line Search methods is to find a step size along a certain line which gives us a good reduction on the function value, while being reasonably inexpensive to compute. More formally, they are iterative methods that, at every step, choose a certain *descent direction* and move along this direction. Thus, at every iteration k,

$$x_{k+1} = x_k + \alpha_k p_k.$$

For example, p_k can be chosen as the Newton direction s_k given by (1.31), but in practice any direction which is a descent direction, that is, one for which $\langle p_k, \nabla f(x_k)\rangle < 0$, can be chosen. The most popular methods use some (cheaper) approximation to the Newton direction, and are thus called *Quasi-Newton* methods. Several other choices are possible, but we will not discuss them here. More details can be found in Nocedal and Wright [50].

Once the direction has been chosen, the step length α_k can be then computed in various ways, trying to solve exactly or approximately the one-dimensional minimization problem

$$\min_{\alpha>0} f(x_k + \alpha p_k). \tag{1.32}$$

The solution to this problem is the step length α_k that gives the lowest function value in direction p_k.

Solving this problem exactly yields the best results, but sometimes it may be too expensive and we might get a very good approximate result using simpler techniques. These approximate solutions are usually obtained by proceeding in two phases: first, we find an interval defining minimal and maximal step lengths, and then a bisection or interpolation phase computes a good step length in this interval.

In order to ensure that even an approximate solution is enough to guarantee the convergence of the line search method to a minimizer of the objective function, some conditions are imposed on the step length at each iteration. One very important condition that must be satisfied is that the decrease obtained in the objective function is not too small. One way of measuring this is by using the following inequality:

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \langle \nabla f(x_k), p_k \rangle, \qquad \text{for some } c_1 \in (0, 1). \tag{1.33}$$

This condition is sometimes called the *Armijo condition* and it states that the reduction in f should be proportional to the step length α_k and the derivative of f.

On the other hand, we must also guarantee that the step is not too short. Indeed, condition (1.33) is satisfied for all sufficiently small values of α. One way of enforcing this is by imposing a *curvature condition*, which requires that α_k satisfy

$$\langle \nabla f(x_k + \alpha_k p_k), p_k \rangle \geq c_2 \langle \nabla f(x_k), p_k \rangle, \qquad \text{for some } c_2 \in (c_1, 1). \tag{1.34}$$

Conditions (1.33) and (1.34) are known as the *Wolfe conditions*. It is possible to prove that, for every function f that is smooth and bounded below, there exist step lengths that satisfy the Wolfe conditions. These conditions on the step length are very important in practice and are widely used in line search methods. See Dennis and Schnabel [14], or Nocedal and Wright [50] for more complete discussions on this subject.

As an alternative to condition (1.34), we can use a *backtracking* procedure in order to find an acceptable step length that satisfies only condition (1.33). This procedure works as follows. Given an initial $\alpha > 0$, and constants $\rho \in (0, 1), c_1 \in (0, 1)$, set $\alpha = \rho\alpha$ until α satisfies condition (1.33). After a finite number of trials, an acceptable step length will be found since α will eventually become small enough to satisfy the sufficient decrease condition. This is a very popular strategy that yields good results in practice.

1.3.3 Conjugate Gradient Methods

Now, we look at a particular problem in optimization, which is minimizing a (strictly) convex quadratic function, that is

$$\min_{x \in \mathbb{R}^n} q(x) = \langle c, x \rangle + \frac{1}{2} \langle x, Hx \rangle,$$

where $c \in \mathbb{R}^n$ and $H \in \mathbb{R}^{n \times n}$. From the optimality conditions stated in Theorem (1.2.4), if H is positive definite, then this is equivalent to finding the solution of $\nabla q(x_*) = 0$, and thus a

minimizer x^* of q must also be a solution to the linear system

$$Hx_* = -c. \tag{1.35}$$

This system can be solved either by a direct method, for instance via the factorization of H, or by an iterative method. While solving the problem by factorization might be a possibility, we are particularly interested in big problems, in which the factorization of H might be very hard to compute.

The *conjugate gradient method* is based on the fact that the function q can be minimized in n steps if we minimize it successively in a certain set of directions, called *conjugate directions*. A set of nonzero vectors $\{p_0, p_1, \ldots, p_m\}$ is said to be *conjugate* with respect to the symmetric positive definite matrix H, or *H-conjugate*, if

$$\langle p_i, Hp_j \rangle, \qquad \text{for all } i \neq j.$$

The conjugate gradient method is described in Algorithm 1.3.1.

Algorithm 1.3.1: Conjugate Gradient Method

Given x_0, set $g_0 = Hx_0 + c$ and let $p_0 = -g_0$. For $k = 0, 1, \ldots$ until convergence, do

- $\alpha_k = \dfrac{\|g_k\|_2^2}{\langle p_k, Hp_k \rangle}$,
- $x_{k+1} = x_k + \alpha_k p_k$,
- $g_{k+1} = g_k + \alpha_k H p_k$,
- $\beta_k = \dfrac{\|g_{k+1}\|_2^2}{\|g_k\|_2^2}$,
- $p_{k+1} = -g_{k+1} + \beta_k p_k$.

The conjugate gradient method is equivalent to minimizing q successively in the *Krylov subspace* of H, defined as

$$\mathcal{K}(H, g_0, j) \stackrel{\text{def}}{=} \text{span}\{g_0, g_1, \ldots, g_j\} = \text{span}\{g_0, Hg_0, H^2 g_0, \ldots, H^j g_0\}.$$

This method is important as a solver for linear systems of equations as well as for the minimization of quadratic functions. It was first developed in the 1950's by Hestenes and Stiefel [32]. However, its convergence depends strongly on the condition number $\kappa_2(H)$ of the matrix H. Indeed, it can be shown that

$$\|x_k - x_*\|_H \leq 2 \left(\frac{\sqrt{\kappa_2(H)} - 1}{\sqrt{\kappa_2(H)} + 1} \right)^k \|x_0 - x_*\|_H, \tag{1.36}$$

where $e_k \stackrel{\text{def}}{=} x_k - x_*$ is the *error* at iteration k (see Theorem 5.1.7 on page 85 in Conn et al. [12], for example).

From this inequality, we can deduce that the speed of convergence of the conjugate gradient method depends on the conditioning number of H: the smaller this number is, the faster the convergence of the method. Thus, one of the ways of speeding this convergence rate is by trying to improve on the condition number of H. This is the idea behind *preconditioning*.

Let $G \in \mathbb{R}^{n\times n}$ by a nonsingular matrix. By defining a new problem

$$(G^{-T}HG^{-1})Gx = -G^{-T}c, \tag{1.37}$$

it is clear that the solution of this new problem satisfies

$$x_* = -(G^{-T}HG^{-1}G)^{-1}G^{-T}c = -H^{-1}c,$$

and thus the solution to the preconditioned system (1.37) is the same as the solution to the original system (1.35).

If we can find a matrix G such that the condition number of $G^{-T}HG^{-1}$ gives a better bound on the convergence factor given by 1.36, then we can apply the conjugate gradient method to the new system (1.37) and have a faster convergence, using Algorithm 1.3.2. Conveniently, this adaptation of the algorithm does not require the use of G, but rather that of $M = G^TG$, which is symmetric and positive definite.

Algorithm 1.3.2: Preconditioned Conjugate Gradient Method

Given x_0, set $g_0 = Hx_0 + c$ and let $v_0 = M^{-1}g_0$ and $p_0 = -v_0$. For $k = 0, 1, \ldots$ until convergence, do

- $\alpha_k = \dfrac{\langle g_k, v_k\rangle}{\langle p_k, Hp_k\rangle}$,
- $x_{k+1} = x_k + \alpha_k p_k$,
- $g_{k+1} = g_k + \alpha_k Hp_k$,
- $v_{k+1} = M^{-1}g_{k+1}$,
- $\beta_k = \dfrac{\langle g_{k+1}, v_{k+1}\rangle}{\langle g_k, v_k\rangle}$,
- $p_{k+1} = -v_{k+1} + \beta_k p_k$.

There are several possible choices for M. One of the most commonly used is the diagonal of H, which works very well when H is diagonally dominant. We will not discuss this matter further as it is outside the scope of this work, but more details can be found in Conn et al. [12], Golub and Van Loan [21], or Nocedal and Wright [50].

1.3.4 Conjugate Gradients for Linearly Constrained Problems

Now, consider that we want to minimize a quadratic function with a set of linear constraints, that is, we wish to find the solution to the problem

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & q(x) = \langle c, x \rangle + \tfrac{1}{2} \langle x, Hx \rangle \\ \text{subject to} \quad & Ax = b, \end{aligned} \tag{1.38}$$

where $A \in \mathbb{R}^{m \times n}$ ($m \leq n$) is full-rank, and $b \in \mathbb{R}^m$. This problem is important here because the trust-region subproblem defined when trying to solve a bound-constrained minimization problem, such as the one we will consider in Chapter 4, has the form (1.38).

From the KKT conditions (1.23), and particularly from (1.24), we can thus deduce that a solution x^* to problem (1.38) must satisfy both

$$Hx + A^T \lambda = -c, \tag{1.39}$$

where λ is a vector that contains the m Lagrange multipliers for problem (1.38) at each component, and the constraints of the problem. Putting these two conditions together in matrix form, we have that x^* must satisfy

$$\begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} -c \\ b \end{pmatrix}.$$

Now, suppose that we can find a full-rank matrice N whose columns span the null-space of matrix A. Then, we can rewrite any vector x that satisfies $Ax = b$ as

$$x = x^{\mathrm{R}} + N x^{\mathrm{N}}, \tag{1.40}$$

where x^{R} also satisfies $Ax^{\mathrm{R}} = b$. Substituting then (1.40) into (1.39), and premultiplying by N^T, we obtain that, since $AN = 0$,

$$N^T H N x^{\mathrm{N}} = -N^T c - N^T H x^{\mathrm{R}}.$$

From this equation we can recover the null-space component x^{N} by factorizing $N^T HN$, which implies that problem (1.38) has a unique solution whenever $N^T HN$ is positive (semi)definite.

Here, since A is full-rank and $m \leq n$, the rows of A itself give a basis for its range-space, and thus a basis for its null-space can be found by computing

$$N = P \begin{pmatrix} -(A^{\mathrm{R}})^{-1} A^{\mathrm{N}} \\ I \end{pmatrix}, \text{ where } A = (A^{\mathrm{R}} A^{\mathrm{N}}) P^T. \tag{1.41}$$

The matrix P is a permutation matrix chosen so that $A^{\mathrm{R}} \in \mathbb{R}^{m \times m}$ is nonsingular. This method does not generate orthonormal bases, but it allows for an effective computation of $(A^{\mathrm{R}})^{-1} A^{\mathrm{N}}$, which is very useful in practice.

Now, since it is relatively easy to find a point that satisfies the constraints, we can restrict our analysis to the case where the constraints are linear, that is, $Ax = 0$. Thus, we can restate the problem we are interested in as

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & q(x) = \langle c, x \rangle + \tfrac{1}{2} \langle x, Hx \rangle \\ \text{subject to} \quad & Ax = 0. \end{aligned} \tag{1.42}$$

In this case, $x = Nx^{\mathrm{N}}$. Thus, solving (1.42) is equivalent to solving the *unconstrained* problem

$$\min_{x^{\mathrm{N}} \in \mathbb{R}^{n-m}} q(Nx^{\mathrm{N}}) = \langle c^{\mathrm{N}}, x^{\mathrm{N}} \rangle + \frac{1}{2} \langle x^{\mathrm{N}}, H^{\mathrm{N}} x^{\mathrm{N}} \rangle, \tag{1.43}$$

where $H^{\mathrm{N}} = N^T H N$ and $c^{\mathrm{N}} = N^T c$.

It is then possible to apply a preconditioned conjugate gradient method to this problem, under the condition that we can compute and apply N and that we can find a preconditioner to H^{N}. Since N can be found by (1.41), it remains the issue of finding a good preconditioner for H^{N}.

If we apply Algorithm 1.3.2 to problem (1.43), and if we rename the quantities such that $x = Nx^{\mathrm{n}}$, we obtain Algorithm 1.3.3.

Algorithm 1.3.3: Projected Preconditioned Conjugate Gradient Method

Given x_0 such that $Ax_0 = 0$, set $g_0 = Hx_0 + c$ and let $v_0 = N(M^{\mathrm{N}})^{-1} N^T g_0$ and $p_0 = -v_0$. For $k = 0, 1, \ldots$ until convergence, do

- $\alpha_k = \dfrac{\langle g_k, v_k \rangle}{\langle p_k, H p_k \rangle}$,
- $x_{k+1} = x_k + \alpha_k p_k$,
- $g_{k+1} = g_k + \alpha_k H p_k$,
- $v_{k+1} = N(M^{\mathrm{N}})^{-1} N^T g_{k+1}$,
- $\beta_k = \dfrac{\langle g_{k+1}, v_{k+1} \rangle}{\langle g_k, v_k \rangle}$,
- $p_{k+1} = -v_{k+1} + \beta_k p_k$.

Now, since M^{N} is supposed to approximate H^{N}, if we require that is has the form

$$M^{\mathrm{N}} = N^T M N,$$

where M approximates H, then v_{k+1} can be computed by

$$v_{k+1} = N(N^T M N)^{-1} N^T g_{k+1}.$$

This is simply the null-space method applied to the problem

$$\min_{v \in \mathbb{R}^n} \quad \tfrac{1}{2} \langle v, Mv \rangle - \langle v, g_{k+1} \rangle, \tag{1.44}$$

$$\text{subject to} \qquad Av = 0. \tag{1.45}$$

Thus, v_{k+1} must satisfy

$$\begin{pmatrix} M & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} v_{k+1} \\ w_{k+1} \end{pmatrix} = \begin{pmatrix} g_{k+1} \\ 0 \end{pmatrix}, \tag{1.46}$$

for some auxiliary variable w_{k+1}. Therefore, in practice, we replace the computation of v_0 and v_{k+1} in Algorithm 1.3.3 by the solution of (1.46), and we have a complete procedure for the solution of (1.42).

1.4 Trust-Region Methods

In this section, we will discuss trust-region methods, which are the basis for the work presented in this thesis. They were first proposed by Levenberg [39] and Marquardt [43] in the context of nonlinear least-squares problems. The idea was further enhanced by Goldfeldt et al. [20], and the first convergence theory in the context of unconstrained optimization appears in Powell [56]. Since then, the method has been improved and updated, and what we present here is the basic method used today.

As opposed to line search methods, trust-region methods work by, at iteration k, first defining a *trust region* around the current iterate x_k, and defining a (usually quadratic) model m_k around the current point x_k. The direction in which to take the step is then computed by (exactly or approximately) minimizing this quadratic model inside the trust region.

Let us focus on the details of a basic trust-region algorithm. Starting from an initial point x_0, at each iteration k, we define a trust region around x_k as

$$\mathcal{B}_k = \mathcal{B}(x_k, \Delta_k) \stackrel{\text{def}}{=} \{\|x - x_k\| \leq \Delta_k\}, \tag{1.47}$$

where $\Delta_k > 0$ is called the *trust-region radius*, and $\|\cdot\|$ is some vector norm. The most common norms used in this type of algorithms are the Euclidean norm $\|\cdot\|_2$ and the maximum norm $\|\cdot\|_\infty$.

After defining the trust region, we use Taylor's approximation (1.29) to define a quadratic model

$$m_k(x_k + s_k) = f(x_k) + \langle g_k, s_k \rangle + \frac{1}{2}\langle s_k, H_k s_k \rangle, \tag{1.48}$$

where g_k is a vector and H_k is a matrix. In general, $g_k = \nabla f(x_k)$ and H_k is the Hessian matrix of f at x_k, $\nabla^2 f(x_k)$, or some symmetric approximation to it. We then compute a step s_k as the solution to the minimization problem

$$\begin{aligned} &\min m_k(x_k + s) \\ &\text{s.t. } x_k + s \in \mathcal{B}_k \end{aligned} \tag{1.49}$$

This is called the *trust-region subproblem*. The new point $x_k + s_k$ is called the *trial point*.

Once the step has been found, we check if the model agrees with the objective function. This amounts to testing if the *actual reduction* obtained in the function and the *predicted reduction* obtained in the model are sufficiently close; that is, we check if

$$\rho_k \stackrel{\text{def}}{=} \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)} \tag{1.50}$$

is larger than some constant $\eta_1 \in (0, 1)$. If this is the case, then we consider the iteration as *successful* and set the new iterate to $x_{k+1} = x_k + s_k$. If the ratio (1.50) is smaller than η_1, we reject the trial point and reduce the trust-region radius Δ_k, in order to recompute a new minimizer for the model m_k inside the new, smaller trust region.

In Algorithm 1.4.1 we present the basic trust-region algorithm in its entirety, as stated by Conn et al. [12].

Algorithm 1.4.1: Basic Trust Region (BTR)

Step 0: Initialization. An initial point x_0 and an initial trust-region radius Δ_0 are given. Constants η_1, η_2, γ_1 and γ_2 such that

$$0 < \eta_1 \leq \eta_2 < 1 \text{ and } 0 < \gamma_1 \leq \gamma_2 < 1 \tag{1.51}$$

are also given. Compute $f(x_0)$ and set $k = 0$.

Step 1: Model definition. Define a model m_k in $\mathcal{B}_k$.

Step 2: Step Calculation. Compute a step s_k that "sufficiently reduces the model" m_k and such that $x_k + s_k \in \mathcal{B}_k$.

Step 3: Acceptance of the trial point. Compute $f(x_k + s_k)$ and define

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}.$$

If $\rho_k \geq \eta_1$, then define $x_{k+1} = x_k + s_k$; otherwise, define $x_{k+1} = x_k$.

Step Trust-region radius update. Set

$$\Delta_{k+1} \in \begin{cases} [\Delta_k, \infty) & \text{if} \quad \rho_k \geq \eta_2, \\ [\gamma_2 \Delta_k, \Delta_k] & \text{if} \quad \rho_k \in [\eta_1, \eta_2), \\ [\gamma_1 \Delta_k, \gamma_2 \Delta_k] & \text{if} \quad \rho_k < \eta_1. \end{cases} \tag{1.52}$$

Increment k by 1 and go to Step 1.

In this trust-region algorithm, some details have been left undefined. For instance, we do not specify which norm we use for the definition of the trust region. The choice of norm is very important for the resolution of the trust-region subproblem (1.49), and we will discuss these issues in Chapter 3.

Another detail that must be clarified is a stopping rule for this algorithm. In practice, we will test if the gradient is small enough, that is if

$$\|g\| \leq \epsilon^g, \quad \epsilon^g > 0.$$

In this case, we will accept the convergence of the algorithm.

We will now briefly discuss the main convergence results related to trust-region methods which will play an important role in the rest of this work.

1.4.1 Convergence Theory

In this section, we will state for future reference the main results referring to the convergence theory of trust-region methods. The discussion here follows closely that of Conn et al. [12], and we refer the reader to that work for details and the proofs of all the results stated here.

1.4.1.1 The Cauchy Point

In order to state these basic results, we must first describe what characterizes a solution to the trust-region subproblem (1.49) that *sufficiently reduces* the model value.

Since our model is a quadratic function, the simplest way to find a minimizer to this model is by using the steepest descent method, mentioned briefly in Section 1.3. In this case, we will try to find a point along the direction $-g_k$, inside the trust region, that gives us the largest reduction in model value. This means that we will be looking for a point along the *Cauchy arc* defined by

$$x_k^C(t) \stackrel{\text{def}}{=} \{x \mid x = x_k - tg_k, t \geq 0 \text{ and } x \in \mathcal{B}_k\}. \tag{1.53}$$

Since our model is quadratic, we can minimize the model exactly along the Cauchy arc. The unique solution to this problem,

$$x_k^C = x_k - t_k^C g_k = \arg \min_{\substack{t \geq 0 \\ x_k^C(t) \in \mathcal{B}_k}} m_k(x_k - tg_k), \tag{1.54}$$

is called the *Cauchy point*. In some cases, computing the exact minimizer to the Cauchy arc (1.53) is too expensive. In these cases, we may use a backtracking strategy, similar to the one described in Section 1.3.2. This amounts to finding the smallest integer $j = j_c \geq 0$ such that

$$x_k(j) \stackrel{\text{def}}{=} x_k - \kappa_{\text{bt}}^j \frac{\Delta_k}{\|g_k\|} g_k,$$

with $\kappa_{\text{bt}} \in (0, 1)$, satisfies the Armijo condition (1.33) for m_k, that is,

$$m_k(x_k(j)) \leq m_k(x_k) + c_1 \langle g_k, x_k(j) - x_k \rangle, \quad c_1 \in (0, 1).$$

We then define the *approximate Cauchy point* as $x_k(j_c)$. In general, we will require our model reduction to be defined as at least that obtained by the (exact or approximate) Cauchy point. We will then define the sufficient reduction on the model for all iterations k as

$$m_k(x_k) - m_k(x_k + s_k) \geq \kappa_{\text{red}} \|g_k\|_2 \min \left[\frac{\|g_k\|_2}{\beta_k}, \Delta_k \right], \tag{1.55}$$

where $\kappa_{\text{red}} \in (0, 1)$. This condition comes from the estimates for the model reduction obtained by either the exact or approximate Cauchy point, and it is often referred to as the *Cauchy condition*. We refer to Section 6.3 in Conn et al. [12] for a detailed discussion on the subject.

With this requirement, we can prove the following result.

Theorem 1.4.1 *Suppose that the model is given by (1.48), and that the model reduction satisfies (1.55) for all k. Suppose furthermore that* $\nabla f(x_k) \neq 0$. *Then* $m_k(x_k + s_k) < m_k(x_k)$ *and the step* $s_k \neq 0$.

The model decrease stated in (1.55) is a lower bound in the decrease we might obtain. If we accept the cost of additional computation, we might even want to solve the trust-region subproblem (1.49) exactly. However, an approximate solution to this problem is acceptable in practice, as shown by the following result.

Theorem 1.4.2 *Let* x_k^M *be the solution of (1.49). Suppose that, for all k, the step* s_k *is such that*

$$m_k(x_k) - m_k(x_k + s_k) \geq \kappa_{ared}[m_k(x_k) - m_k(x_k^M)], \tag{1.56}$$

where $\kappa_{ared} \in (0, 1]$. *Then, (1.55) holds for some* κ_{red}.

Thus, if the reduction obtained in the model is at least some fraction of the reduction we would obtain with the (exact) model minimizer, the Cauchy condition is satisfied.

1.4.1.2 Convergence to First-Order Critical Points

The goal of this section is to state the main results that prove the global convergence of Algorithm 1.4.1 to first-order critical points. We do this for completeness, and in order to show that the theory for Recursive Multilevel Trust-Region methods, detailed in Chapter 4, follows closely the one presented in this section.

To prove that Algorithm 1.4.1 is globally convergent to first-order critical points, we must prove that all limit points x_* of the sequence $\{x_k\}$ generated by the BTR Algorithm satisfy

$$\nabla f(x_*) = 0$$

independent of the position of the initial point x_0, or of the choice of the initial trust-region radius Δ_0.

Throughout this section, we will use the following assumptions.

Assumption 1 (Assumptions on the function.) *The objective function* f *is such that:*

- f *is twice continuously differentiable.*
- *The Hessian of* f *is uniformly bounded; that is, there exists a* $\kappa_{ufh} \geq 1$ *such that, for all* $x \in \mathbb{R}^n$, $\|\nabla^2 f(x)\|_2 \leq \kappa_{ufh}$.

Assumption 2 (Assumptions on the model.) *The model* m_k *satisfies the following conditions:*

- *For all k, the model* m_k *is twice differentiable on* $\mathcal{B}_k$.
- *For all k,* $m_k(x_k) = f(x_k)$.
- *For all k,* $g_k = \nabla f(x_k)$.

- *The Hessian of the model is bounded within the trust region:*

$$\|\nabla^2 m_k(x)\|_2 \leq \kappa_{umh} - 1, \text{ for all } x \in \mathcal{B}_k,$$

where $\kappa_{umh} \geq 1$ *is independent of* k.

Assumption 3 (Norm equivalence.) *There exists a constant* $\kappa_{une} \geq 1$ *such that, for all* k*, the norm* $\|\cdot\|$ *chosen to define the trust region satisfies*

$$\frac{1}{\kappa_{une}}\|x\| \leq \|x\|_2 \leq \kappa_{une}\|x\|$$

for all $x \in \mathbb{R}^n$.

First, we will look at the error between the objective function and the model at a new iterate $x_k + s_k \in \mathcal{B}_k$.

Theorem 1.4.3 *Suppose that Assumptions 1 and 2 hold. Then, for all* k*, we have that*

$$|f(x_k + s_k) - m_k(x_k + s_k)| \leq [\nu_k^s]^2 \max\left[\kappa_{ufh}, \kappa_{umh}\right] \Delta_k^2,$$

where $x_k + s_k \in \mathcal{B}_k$ *and*

$$\nu_k^s = \frac{\|s_k\|_2}{\|s_k\|}.$$

Moreover, if Assumption 3 also holds, then

$$|f(x_k + s_k) - m_k(x_k + s_k)| \leq \kappa_{ubh}\Delta_k^2,$$

where $\kappa_{ubh} = \kappa_{une}^2 \max[\kappa_{ufh}, \kappa_{umh}]$.

This theorem translates into the notion that the smaller the trust-region radius, the better the model approximates the objective function in that region. The next result shows that, indeed, if the trust-region radius is small enough, it must result in a successful iteration, so long as we are not in a first order critical point already.

Theorem 1.4.4 *Suppose that Assumptions 1, 2 and 3 hold, as well as (1.55). Suppose furthermore that* $g_k \neq 0$ *and that*

$$\Delta_k \leq \frac{\kappa_{red}\|g_k\|_2(1 - \eta_2)}{\kappa_{ubh}}.$$

Then, iteration k *is very successful and*

$$\Delta_{k+1} \geq \Delta_k.$$

This result implies that the trust-region radius cannot become too small, unless we are at a first-order critical point.

Theorem 1.4.5 *Suppose that Assumptions 1, 2 and 3 hold, and that $f(x)$ is bounded below on $\mathbb{R}^n$. Suppose furthermore that (1.55) is satisfied, and that there exists a constant $\kappa_{lbg} > 0$ such that $\|g_k\|_2 \geq \kappa_{lbg}$ for all k. Then, there is a constant $\kappa_{lbd} > 0$ such that*

$$\Delta_k \geq \kappa_{lbd}$$

for all k.

Theorem 1.4.5 is very important since it guarantees that Algorithm 1.4.1 can always proceed, unless we are already at a first-order critical point of f. With these results in hand, we can state the following theorem.

Theorem 1.4.6 *Suppose that Assumptions 1, 2 and 3 hold. Suppose furthermore that (1.55) is satisfied, and that there are only finitely many successful iterations. Then, $x_k = x_*$ for all sufficiently large k and x_* is first-order critical.*

Now, we must look at the case when there are infinitely many successful iterations. First, using the fact that if Δ_k is small enough, m_k approximates the function well, as stated by Theorem 1.4.3, and that at the same time, the trust-region radius cannot be too small because of Theorem 1.4.5, we can prove that at least one accumulation point of the infinite sequence of iterates must be first-order critical.

Theorem 1.4.7 *Suppose that Assumptions 1, 2 and 3 hold, and that $f(x)$ is bounded below on $\mathbb{R}^n$. Suppose furthermore that (1.55) is satisfied. Then,*

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

Using these results, it it possible then to extend the last Theorem to prove that not only one of the accumulation points of the sequence of iterates is first-order critical, but that, in fact, all are.

Theorem 1.4.8 *Suppose that Assumptions 1, 2 and 3 hold, and that $f(x)$ is bounded below on $\mathbb{R}^n$. Suppose furthermore that (1.55) is satisfied. Then,*

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

These theoretical results, and excellent behavior in practice, show that Trust-Region methods are a very good choice for unconstrained optimization problems, and why they are widely used in several applications in various fields of science and engineering. In the next chapters, we will try to show that they can be even further improved on by expanding on these ideas and ensuring that they are competitive in large applications.

1.4.1.3 Criticality Measure

We define $\pi(k, x_k)$ to be a *first-order criticality measure* of the iterate x_k if it is a nonnegative real function of its second argument such that

$$\|x_k - x_\ell\| \to 0 \text{ implies that } |\pi(k, x_k) - \pi(\ell, x_\ell)| \to 0,$$

and if the limit

$$\lim_{k\to\infty} \pi(k, x_k) = 0$$

corresponds to asymptotically satisfying the first-order criticality conditions of the optimization problem considered.

Of course, $\pi(k, x_k) = \|\nabla f(x_k)\|$ is one of many possible criticality measures for unconstrained problems. The definition of a general criticality measure will be of importance in Chapter 4, where one particular alternative will be considered. It is important to note that, despite the generality of this definition, all the first-order convergence results obtained for the classical trust-region method can be extended for a general criticality measure $\pi(k, x_k)$. For more details, see Section 8.1 in Conn et al. [12].

1.4.2 A Note on Bound Constraints

As we have mentioned in Section 1.1, bound-constrained problems of the form

$$\begin{array}{ll} \text{minimize} & f(x), \\ \text{subject to} & x \in C = \{x \in \mathbb{R}^n \mid \ell \leq x \leq u\}, \end{array} \tag{1.57}$$

where the inequalities are considered componentwise, that is $\ell_i \leq x_i \leq u_i$, for all $i = 1, \ldots, n$, can be considered separately due to certain properties satisfied by them. Indeed, in this case we can easily compute the projection of any vector y onto C, by defining

$$[\text{Proj}_C(y)]_i \stackrel{\text{def}}{=} \begin{cases} \ell_i & \text{if } y_i \leq \ell_i, \\ y_i & \text{if } \ell_i \leq y_i \leq u_i, \\ u_i & \text{if } u_i \leq y_i \end{cases}$$

for $i = 1, \ldots, n$. Thus, it is natural to think of projections when devising methods to solve bound-constrained problems. In particular, when we define the *projected-gradient path* as

$$p(t, x) \stackrel{\text{def}}{=} \text{Proj}_C[x - t\nabla f(x)],$$

for any $x \in C$ and for all $t > 0$, it is possible to prove the following theorem.

Theorem 1.4.9 *Suppose that the set C of constraints is nonempty, closed and convex. Suppose also that a constraint qualification (such as the LICQ Condition 1.1) holds at x^*. Then the point $x^* \in C$ is a first-order critical point for problem (1.57) if and only if*

$$p(t, x^*) = x^* \text{ for all } t \geq 0.$$

In this case, the projected-gradient path can be used to define a new criticality measure

$$\chi(x) \stackrel{\text{def}}{=} |\min_{\substack{x+d\in C \\ \|d\|\leq 1}} \langle \nabla f(x), d\rangle|$$

for all $x \in C$, since it can be shown (see, for example, Section 12.1.3 in Conn et al. [12], pp. 444-451) that $\chi(x) = 0$ if and only if $p(t, x) = x$ for all $t \geq 0$ and thus, from Theorem 1.4.9, $\chi(x^*) = 0$ if and only if x^* is first-order critical.

These properties are critical for the analysis in Chapter 4, since in that chapter we will deal with bound-constrained problems. For more details and proofs, see Conn et al. [12].

Chapter 2

Multigrid Methods for Linear Systems

2.1 Introduction

Linear systems of equations appear in virtually every model that aims to describe a real-life application in mathematical terms. These systems are usually stated as finding $x \in \mathbb{R}^n$ that satisfies

$$Ax = b, \tag{2.1}$$

where $A \in \mathbb{R}^{n \times n}$ is a matrix, and $b \in \mathbb{R}^n$ is a vector.

When the problem arises from the discretization of an underlying continuous problem, a multilevel hierarchy can be formulated, such that each discretization *level* has more variables (that is, is a finer discretization of the domain of the problem) than the previous one. Thus, if $p + 1$ levels of discretization are available, this amounts to a set of $p + 1$ nonlinear equations, each defined in a space $\mathbb{R}^{n_i}$, such that $n_0 \leq n_1 \leq \ldots \leq n_p$, which can be written as

$$A_i x_i = b_i,$$

with $A_i \in \mathbb{R}^{n_i \times n_i}$, and $x_i, b_i \in \mathbb{R}^{n_i}$, $i = 0, \ldots, p$.

The methods that exploit this multilevel hierarchy for the solution of linear systems of equations are called *Multigrid* methods. This well-researched field, pioneered by Fedorenko [16] and later by Brandt [2], is based on a double observation: on one hand there exist iterative solution methods (called *smoothers*) which are very efficient at reducing the high-frequency, oscillatory components of the error while being possibly very inefficient at reducing their low-frequency (also called *smooth*) components (the Jacobi and Gauss-Seidel methods are preeminent examples); on the other hand, the definition of a high-frequency component of the error is intrinsically tied to the discretization grid since the finer the grid, the higher the frequency representable on this grid. Multigrid methods then proceed by using methods called *smoothers* to reduce the high-frequency (also called *oscillatory*) error components on a fine grid, and then consider the remaining smooth components on this fine grid as oscillatory ones on a coarser grid. Broadly speaking, these can again be eliminated using smoothers on the coarser grid, and this technique may be applied recursively. One of the main attractions of multigrid methods for linear systems is that their workload increases only linearly with problem size, a feature crucial for the solution of very large instances.

Multigrid methods are fairly established as solvers for large linear systems derived from (mainly elliptic) Partial Differential Equations, in particular boundary value problems. Here, we will present a brief introduction on this technique, with the purpose of motivating the multilevel aspect of our optimization applications. For a more detailed introduction to the subject of multigrid methods, the reader is referred to Briggs et al. [3], Trottenberg et al. [67] and Wesseling [68].

2.2 Model Problem

One application where the resulting linear system is large and sparse is in the solution of elliptic boundary value problems. One example is the one-dimensional Poisson equation with Dirichlet boundary conditions

$$-\mathbf{u}''(x) = \mathrm{f}(x) \qquad x \in (0,1) \tag{2.2a}$$

$$\mathbf{u}(0) = \mathbf{u}(1) = 0 \tag{2.2b}$$

In order to solve this problem numerically, we will approximate this equation using finite differences. In other words, we will *discretize* $\Omega = (0,1)$ by partitioning it into n subintervals defined by *grid points* $x_j = jh$, $j = 0, \ldots, n$ where $h = \frac{1}{n}$ is the length of each subinterval. We will call the discretized domain of the new problem Ω_h. Figure 2.1 shows this discretized domain.

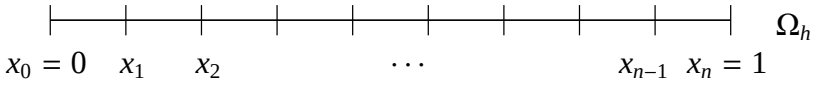


Figure 2.1: One-dimensional discretized domain $\Omega_h = [0,1]$.

At each of the $n-1$ interior grid points, we will approximate (2.2a) by finite differences. Let us define, for simplicity, u_j as an approximation to the exact solution $\mathbf{u}(x_j)$, for $j = 0, \ldots, n$. Then, grouping all u_j in a vector u, the components of this vector must satisfy the $n-1$ linear equations

$$\begin{aligned} \frac{-u_{j-1} + 2u_j - u_{j+1}}{h^2} &= f_j, \qquad 1 \le j \le n-1, \\ u_0 = u_n &= 0, \end{aligned} \tag{2.3}$$

where f is a vector such that each of its components is defined as $f_j = \mathrm{f}(x_j)$, $j = 1, \ldots, n-1$. We can thus represent this system of linear equations in matrix form as

$$\frac{1}{h^2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}\begin{bmatrix} u_1 \\ \cdot \\ \cdot \\ \cdot \\ u_{n-1} \end{bmatrix} = \begin{bmatrix} f_1 \\ \cdot \\ \cdot \\ \cdot \\ f_{n-1} \end{bmatrix} \tag{2.4}$$

or $Au = f$, where $A \in \mathbb{R}^{(n-1)\times(n-1)}$ is tridiagonal, symmetric and positive definite.

Now, let us consider the two-dimensional Poisson equation

$$\begin{aligned} -\Delta \mathbf{u}(x,y) &= \mathrm{f}(x,y) \qquad (x,y) \in \Omega \\ \mathbf{u}(x,y) &= 0 \qquad (x,y) \in \partial\Omega \end{aligned} \tag{2.5}$$

where $\Delta\mathbf{u}(x,y)$ denotes the Laplacian of $\mathbf{u}$, $\Omega = [0,1] \times [0,1]$ and $\partial\Omega$ is the boundary of the unit square. Here, we will define a two-dimensional grid by the grid points $(x_i, y_j) = (ih_x, jh_y)$, where $h_x = \frac{1}{m}$ and $h_y = \frac{1}{n}$. This grid is also denoted by Ω^h. Figure 2.2 shows this discretized domain.

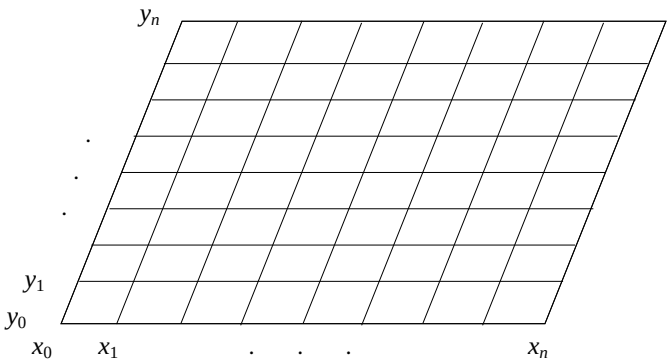


Figure 2.2: Two-dimensional discretized domain $\Omega_h = [0,1] \times [0,1]$.

Using once more a finite differences approximation to the derivatives in (2.5), we derive the system of equations

$$\begin{aligned} &\frac{-u_{i-1,j} + 2u_{ij} - u_{i+1,j}}{h_x^2} + \frac{-u_{i,j-1} + 2u_{i,j} - u_{i,j+1}}{h_y^2} = f_{i,j}, \\ &u_{i,0} = u_{i,n} = u_{0,j} = u_{m,j} = 0, \qquad 1 \le i \le m-1, 1 \le j \le n-1. \end{aligned} \tag{2.6}$$

Here, $u_{i,j}$ is an approximation to the exact solution $\mathbf{u}(x_i, y_j)$ and $f_{i,j} = \mathrm{f}(x_i, y_j)$.

Now, in order to write this equation in matrix form, we must choose the order in which we consider the grid points. Here, we will choose the *lexicographic* ordering, which takes variables from lines of constant i. This way, we can group all unknowns of the i-th row of the grid in a vector $u_i \stackrel{\text{def}}{=} (u_{i,1}, \ldots, u_{i,n-1})^T$ for $1 \le i \le m-1$. Similarly, we will define a vector $f_i \stackrel{\text{def}}{=} (f_{i,1}, \ldots, f_{i,n-1})^T$. The system (2.6) can then be rewritten in block matrix form as

$$\begin{bmatrix} B & -\frac{1}{h_x^2}I & & & \\ -\frac{1}{h_x^2}I & B & -\frac{1}{h_x^2}I & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{h_x^2}I & B & -\frac{1}{h_x^2}I \\ & & & -\frac{1}{h_x^2}I & B \end{bmatrix} \begin{bmatrix} u_1 \\ \cdot \\ \cdot \\ \cdot \\ u_{m-1} \end{bmatrix} = \begin{bmatrix} f_1 \\ \cdot \\ \cdot \\ \cdot \\ f_{m-1} \end{bmatrix}$$

which is a symmetric, block tridiagonal and sparse system. Each block is a multiple of the $(n-1) \times (n-1)$ identity matrix.

The matrices that arise from this type of discretization usually have some very important properties. For example, most are sparse, symmetric and positive definite. Another important property is that they are often *weakly diagonally dominant*. This means that, if a_{ij} is element of row i and column j of A, then

$$\sum_{j \neq i}^{n} |a_{ij}| \leq |a_{ii}| \text{ for } 1 \leq i \leq n.$$

In other words, the diagonal element is at least as large in absolute value as the sum of the off-diagonal elements in the same row.

The two discretized systems described in (2.3) and (2.6) are called *model problems*, since most of the methods described in literature have been developed with these types of problem in mind. This means that they are the starting point for the methods described here, and will provide relevant insight into the way these methods are built. The multigrid methods presented here, however, can be applied to a large class of problems, including discretized elliptic boundary value problems and others.

2.3 Basic Iterative Methods for Linear Systems

Since the problems we are interested in solving can be written as linear systems of equations, let us examine here the iterative methods available to solve

$$Au = f, \tag{2.7}$$

where $A \in \mathbb{R}^{n \times n}$ is a non-singular matrix, and $u, f \in \mathbb{R}^n$. We have already seen one of these methods (the conjugate gradient method) in Section 1.3.3. Here, we present other methods which have a special property which is particularly interesting in the context of multigrid methods. These methods are sometimes referred to as *relaxation* or *smoothing* methods; we will explain this choice of nomenclature further in this chapter.

2.3.1 Jacobi Method

First, observe that, if we write

$$A = D - L - U, \tag{2.8}$$

where D is the diagonal of A, and $-L$ and $-U$ are its strictly lower- and strictly upper-triangular parts, respectively, we can write

$$(D - L - U)u = f.$$

If we isolate the diagonal part of A, we have that

$$Du = (L + U)u + f,$$

which implies

$$u = D^{-1}(L + U)u + D^{-1} f.$$

We can thus define a fixed-point iteration such that

$$u^{\ell+1} = D^{-1}(L+U)u^{\ell} + D^{-1}f \stackrel{\text{def}}{=} M_{\text{J}}u^{\ell} + s_{\text{J}},$$

where we call M_{J} the *iteration matrix*. This is called the *Jacobi method*.

In component form, this is equivalent to doing, for each iteration ℓ, and each component j of u^{ℓ},

$$u_j^{\ell+1} = \frac{1}{2}\left(u_{j-1}^{\ell} + u_{j+1}^{\ell} + h^2 f_j\right), \quad 1 \leq j \leq n-1,$$

that is, we solve the j-th equation of the linear system (2.7) for the j-th variable using the current approximation for the $(j-1)$-st and $(j+1)$-st unknowns. This method can certainly be very effective when compared to direct methods, but a simple variation of it can yield much better results. This is what we consider next.

2.3.2 Gauss-Seidel Method

Once again considering the split (2.8), we can now isolate the diagonal and the lower-triangular parts of A, obtaining

$$(D-L)u = Uu + f,$$

and thus

$$u = (D-L)^{-1}Uu + (D-L)^{-1}f.$$

The resulting fixed point iteration

$$u^{\ell+1} = (D-L)^{-1}Uu^{\ell} + (D-L)^{-1}f \stackrel{\text{def}}{=} M_{\text{GS}}u^{\ell} + s_{\text{GS}}$$

is called the *Gauss-Seidel method*.

In component form, this amounts to computing, for each iteration ℓ and each component j of u^{ℓ},

$$u_j^{\ell+1} = \frac{1}{2}\left(u_{j-1}^{\ell+1} + u_{j+1}^{\ell} + h^2 f_j\right), \quad 1 \leq j \leq n+1.$$

In other words, in the Gauss-Seidel method we can use the already computed components of the approximation in order to compute the next component, which of course gives better results than we could obtain with the Jacobi iteration, where we use only information from the past iteration to compute the next approximation.

2.4 Error

Let us now analyze the effectiveness of the two methods described in Section 2.3. Suppose thus that this linear system (2.7) has a unique solution u_*. Then, we can define two very important quantities. The *error* is defined as the vector

$$e = u_* - u. \tag{2.9}$$

Both the Jacobi and Gauss-Seidel methods can have their iterations be written as

$$u^1 = Mu^0 + s. \tag{2.10}$$

At the same time, we can see that, as in all fixed point iterations, if u^* denotes the exact solution to the problem, then

$$u_* = Mu_* + s. \tag{2.11}$$

Subtracting equation (2.10) from equation (2.11), we have that

$$e^1 = Me^0.$$

By repeating this argument, we have that

$$e^{\ell+1} = M^\ell e^0. \tag{2.12}$$

Now, by choosing a consistent matrix norm $\|\cdot\|$, we have that

$$\|e^\ell\| \leq \|M\|^\ell \|e^0\|.$$

Thus, if $\|M\| < 1$, we can expect the error to be forced to zero after a number of iterations. It is possible to show (for instance, in Golub and Van Loan [21]) that this is only the case if

$$\rho(M) < 1, \tag{2.13}$$

which implies that the iteration associated with M converges for all initial guesses if and only if $\rho(M) < 1$.

However, in real situations where we do not know the exact solution to the problem, we cannot compute the error (2.9). Thus, we will define another quantity, the *residual*, as

$$r = f - Au, \tag{2.14}$$

This residual shows how far an approximation u to the exact solution is from satisfying the original problem. Since we suppose the exact solution u_* is unique, the residual $r = 0$ if and only if the error $e = 0$. However, it may not be true that when r is small in norm, e is also small in norm. Now, from these two quantities, we can write

$$r = f - Au = Au_* - Au = A(u_* - u) = Ae$$

and thus, we can say that the error must satisfy

$$Ae = r. \tag{2.15}$$

This equation, called *residual equation*, allows us to derive the following scheme for the improvement of an approximation u to the solution: first, we compute the residual (2.14). Then, we solve equation (2.15) for the error. Finally, we compute a new approximation to the solution by setting

$$u^{\text{new}} = u + e.$$

This scheme, although only an informal description at this point, gives us a new idea of how to proceed. Something that must be noted here is the fact that applying a relaxation method (such as Jacobi or Gauss-Seidel) to the original equation $Au = f$ with an arbitrary initial guess is equivalent to applying a relaxation method on the residual equation with the initial guess $e = 0$.

2.5 Smooth and Oscillatory Frequencies

An important concept that will be used in the analysis of iterative methods is that of *eigenfunctions*. An eigenfunction of a linear operator A, defined on some function space, is any non-zero function ω in that satisfies

$$A\omega = \lambda\omega,$$

where $\lambda \in \mathbb{R}$ is the eigenvalue associated with ω. It can be shown (see, for example, Briggs et al. [3], Exercises 8 and 9, page 28) that the eigenfunctions associated with the discrete Laplacian operator described by the matrix in (2.4) have the form

$$\omega_{k,j} = \sin\left(\frac{jk\pi}{n}\right), \quad 1 \leq k \leq n-1, 0 \leq j \leq n,$$

where each j denotes the components of these vectors ω_k, also called *wave functions*, for each k, called the *frequency* of these wave functions. The eigenvalues of A are

$$\lambda_k(A) = 4\sin^2\left(\frac{k\pi}{2n}\right), \quad 1 \leq k \leq n-1.$$

If $k \leq \frac{n}{2}$, we say that the wave ω_k is *smooth*, otherwise it is called *oscillatory*. We call each ω_k a *Fourier mode*. Figure 2.3 shows examples of such modes on a one-dimensional grid with 16 points.

These Fourier modes are very important as a tool for evaluating the progress of an iterative method. For example, since for the Jacobi method applied to problem (2.3) we have that

$$M_J = I - \frac{1}{2}A,$$

then

$$\lambda_k(M_J) = 1 - \frac{1}{2}\lambda_k(A) = 1 - 2\sin^2\left(\frac{k\pi}{2n}\right), \quad 1 \leq k \leq n-1.$$

Now, since A is a symmetric positive definite matrix, its eigenfunctions must generate the entire space. Consider now the error e in problem (2.3). It can be written as a linear combination of the wave functions ω_k, that is

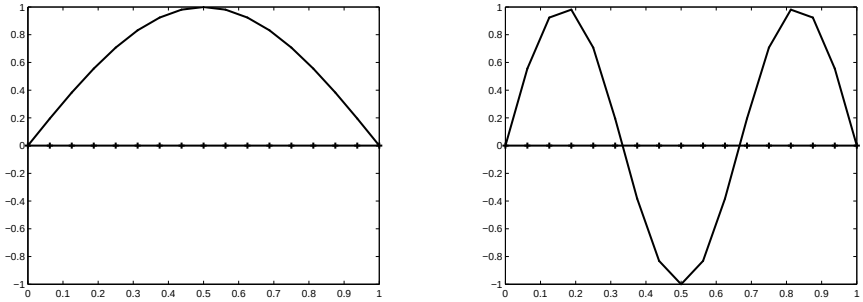
$$e = \sum_{k=1}^{n-1} \alpha_k \omega_k, \quad \alpha_k \in \mathbb{R}.$$

Therefore, (2.12) can be written as

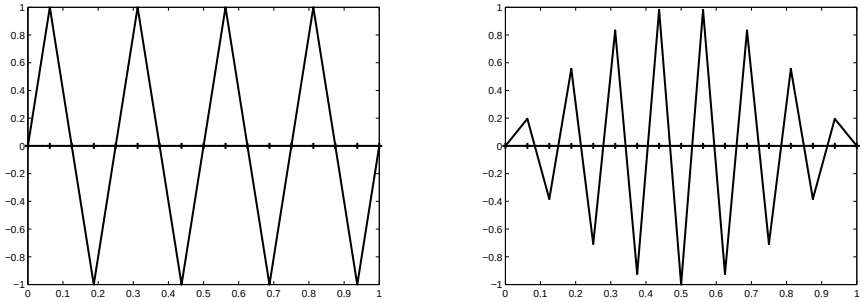
$$e^\ell = M_J^\ell e^0 = \sum_{k=1}^{n-1} \alpha_k M_J^\ell \omega_k = \sum_{k=1}^{n-1} \alpha_k \lambda_k^\ell(M_J)\omega_k, \tag{2.16}$$

since A and M_J have the same eigenvectors ω_k. Thus, after ℓ iterations, the k-th mode of the error will have been reduced by a factor of $\lambda_k^\ell(M_J)$. However, $\lambda_1(M_J)$ can be approximated by

$$\lambda_1(M_J) = 1 - 2\sin^2\left(\frac{\pi}{2n}\right) \approx 1 - \frac{\pi^2 h^2}{2}.$$



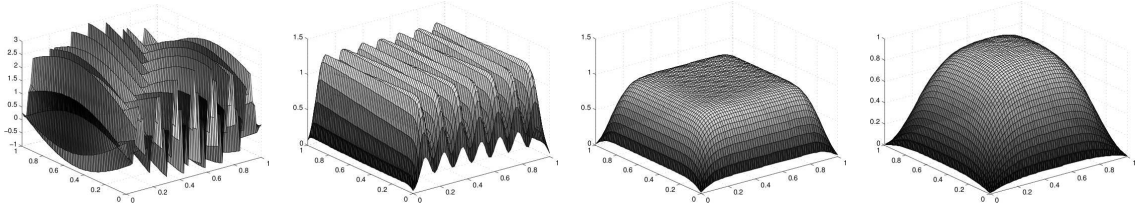
(a) $k = 1$ (b) $k = 3$



(c) $k = 8$ (d) $k = 15$

Figure 2.3: Examples of Fourier modes of A on a one-dimensional grid with 16 points. We show the modes $\omega_{1,j}$, $\omega_{3,j}$, $\omega_{8,j}$ and $\omega_{15,j}$.

This, along with (2.13), implies that the modes associated with this eigenvalue, which are smooth modes, will never be reduced effectively. This is called the *smoothing property* of this family of methods. The same kind of property is true for Gauss-Seidel methods, but we will refer the reader to other texts (for example, Briggs et al. [3] or Trottenberg et al. [67]) for this analysis. Figure 2.4 shows the evolution of the error for problem (2.6) when we apply the Gauss-Seidel method. We can clearly see that oscillatory (that is, high frequency) components of the error disappear rather quickly, while smooth (low-frequency) components remain even after 100 iterations of the method.



(a) Initial error (b) After 5 iterations (c) After 30 iterations (d) After 100 iterations

Figure 2.4: Evolution of the error for problem (2.6) after 5, 30, and 100 iterations of the Gauss-Seidel method.

2.5.1 Coarse and Fine Grids

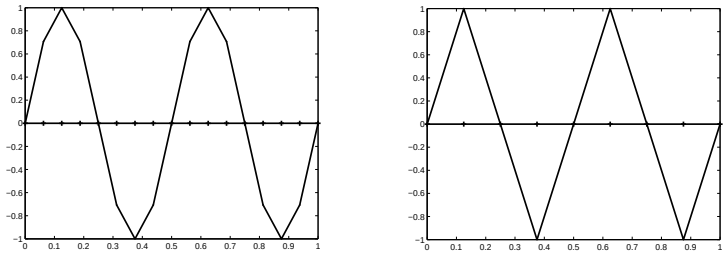
Having seen in (2.16) that the error of the linear system (2.7) can be decomposed into a sum of Fourier modes, it may be useful to look at the properties of these modes. The first thing to note is that these modes are dependent on the grid they are described in. Indeed, let us consider a grid Ω^h with $n^h = \frac{1}{h}$ points, which we will call the *fine grid*, and another grid Ω^{2h} with $n^{2h} = \frac{1}{2h} = \frac{n^h}{2}$ points, which we will call the *coarse grid*, since it has less points than the fine grid[(1)]. Then, the smooth modes evaluated at even numbered points of the grid Ω^h are

$$\omega^h_{k,2j} = \sin\left(\frac{2jk\pi}{n^h}\right) = \sin\left(\frac{jk\pi}{\frac{n^h}{2}}\right), \quad 1 \leq k < \frac{n^h}{2}, 0 \leq j \leq n^h.$$

Now, if we define another grid Ω^{2h} such that $n^{2h} = \frac{n^h}{2}$, then

$$\omega^h_{k,2j} = \omega^{2h}_{k,j}, \quad 1 \leq k < \frac{n^h}{2}.$$

Thus, the smooth modes on the grid Ω^h appear oscillatory on the grid Ω^{2h}. This can be seen in Figure 2.5, where we show how one of these modes is seen in two different grids.



(a) $k = 6, n = 16$ (b) $k = 6, n = 8$

Figure 2.5: The mode $\omega_{6,j}$ represented in two grids with $n = 16$ and $n = 8$ points, respectively. The coarser grid ($n = 8$) *sees* a more oscillatory mode.

Another very special property is what is called *aliasing*. It can be described as the fact that the oscillatory modes on the grid Ω^h are misrepresented as smooth modes on the grid Ω^{2h}. This happens because the k-th mode on Ω^h becomes the $(n-k)$-th mode on Ω^{2h}, for $k > \frac{n}{2}$.

With these properties in mind, we can see that there is some sort of transformation of the frequencies of the error of the fine grid, when they are seen on a coarser grid. This is the key to the development of multigrid methods.

Putting all of these facts together, we can devise a strategy for improving on the convergence of relaxation methods.

[(1)] Sometimes, we will refer to coarse grids as *lower* grids, and finer grids as *upper* grids. This comes from the formulation of these grids, as the finer grid is seen to be closer to the infinite-dimensional description of the problem by having more variables, and thus being higher in the problem hierarchy. It is, however, pure convention.

2.6 Coarse Grid Correction

From the analysis of the previous section, we can thus state the two principles that inspire multigrid methods.

Smoothing principle. Iterative methods such as the Gauss-Seidel and Jacobi iterations eliminate oscillatory components of the error effectively, but leave smooth frequencies of the error unchanged.

Coarse grid principle. Smooth components of the error appear oscillatory in coarser grids.

These two principles suggest a scheme that may be used in order to improve convergence of basic iterative methods for linear systems. Suppose that we may discretize our problem in two grids Ω^h and Ω^{2h}, with $n = \frac{1}{h}$ and $\frac{n}{2}$ points each, respectively. Suppose furthermore that we have two linear *transfer operators*

$$R_h : \mathbb{R}^n \rightarrow \mathbb{R}^{\frac{n}{2}} \text{ and } \quad P_h : \mathbb{R}^{\frac{n}{2}} \rightarrow \mathbb{R}^n,$$

where we call R_h the *restriction* operator from grid Ω^h to grid Ω^{2h}, and P_h is called the *prolongation* operator from grid Ω^{2h} to grid Ω^h. Then, a *coarse grid correction* step can be described as in Algorithm 2.6.1 on the facing page.

The number of smoothing iterations ν_1 and ν_2 are defined by the user, and are usually not very large. Figure 2.6 shows a representation of the coarse-grid correction scheme.

In order to define this scheme more formally, we must define the prolongation and restriction operators. Here, we will only discuss the choices we used in the course of our work, but several other choices are possible. The most common choice for the prolongation operator is the linear interpolation operator, defined in one dimension by

$$\begin{array}{rcl} v_h^{[2j]} & = & v_{2h}^{[j]}, \\ v_h^{[2j+1]} & = & \frac{1}{2}(v_{2h}^{[j]} + v_{2h}^{[j+1]}), \end{array} \quad 0 \leq j \leq \frac{n}{2} - 1,$$

where $v^{[j]}$ denotes the jth component of the vector v. This linear interpolation operator has full rank, and it will be used extensively in our implementations.

For the restriction operator, our choice is the *full weighting* operator, defined by

$$v_{2h}^{[j]} = \frac{1}{4}(v_h^{[2j-1]} + 2v_h^{[2j]} + v_h^{[2j+1]}), \quad \text{for } 1 \leq j \leq \frac{n}{2} - 1.$$

Figure 2.7 shows the action of these operators in one- and two-dimensional grids.

This restriction operator is important because it satisfies the property

$$P_h = \sigma R_h^T, \; \sigma \in \mathbb{R}. \tag{2.18}$$

This is called a *variational property* and will be very important in the following discussion. Of course, these operators can be defined also in three-dimensions. For this, it suffices to define both of the one-dimensional operators for each dimension.

Algorithm 2.6.1: Coarse Grid Correction

- Apply ν_1 iterations of an iterative method to the problem $A_h u_h = f_h$ at the fine grid Ω^h, with initial guess $u_h = 0$.
- Compute the residual $r_h = f_h - A_h u_h$ at the fine grid.
- Restrict the residual to the coarse grid Ω^{2h} by defining

$$r_{2h} = R_h(r_h).$$

- Solve the problem

$$A_{2h} e_{2h} = r_{2h} \tag{2.17}$$

  at the coarse grid.
- Prolongate the correction e_{2h} into the fine grid by defining

$$e_h = P_h(e_{2h}).$$

- Compute the new approximation u_h^{new} in the fine grid by

$$u_h^{\text{new}} = u_h + e_h.$$

- Apply ν_2 iterations of an iterative method to the problem $A_h u_h = f_h$ at the fine grid, with initial guess u_h^{new}.

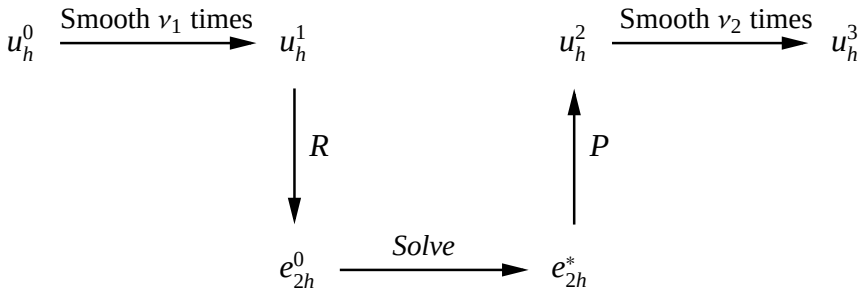
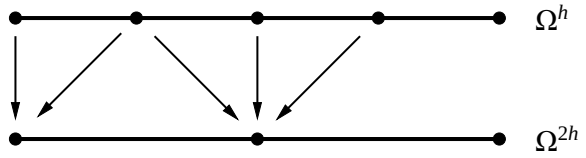


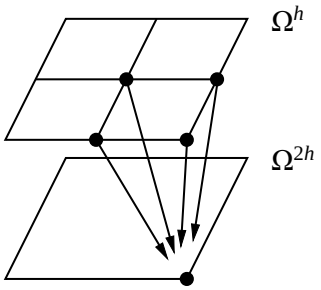
Figure 2.6: The coarse-grid correction scheme.

2.7 Multigrid

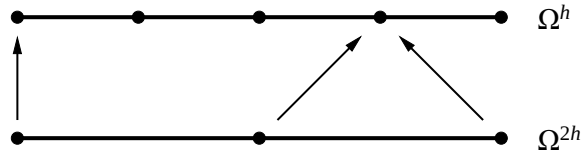
Now, we must decide how to solve the problem (2.17) at the coarse grid. If this problem is small enough, we can solve it exactly by using some factorization of A_{2h}, for example. However, if this is not the case, it can be interesting to use this coarse-grid correction procedure recursively, by correcting equation (2.17) in a coarser grid Ω^{4h}. In order to do this, Suppose that we may discretize our problem in $\ell + 1$ consecutive grids Ω^h, Ω^{2h}, Ω^{4h}, $\ldots$, $\Omega^{2^{\ell-i}h}$, with $n_i = \frac{1}{2^{\ell-i}h}$



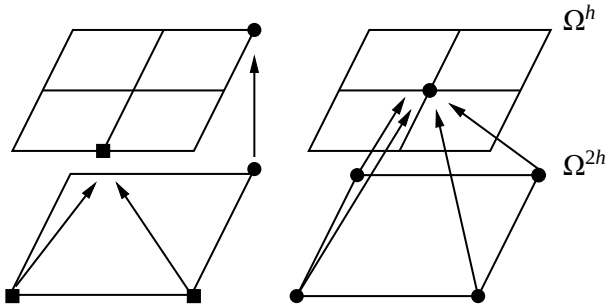
(a) 1D restriction operator.



(b) 2D restriction operator.



(c) 1D prolongation operator.



(d) 2D prolongation operator.

Figure 2.7: Prolongation and restriction operator acting between two grids.

points each, for $i = \ell, \ldots, 0$. Suppose furthermore that we have a collection of operators

$$R_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i-1}}, \quad P_i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i},$$

for $i = 0, \ldots, \ell$. In order to simplify the notation, we will denote a quantity v defined at grid $\Omega^{2^{\ell-i}h}$ as v_i. We will also denote the correction e_{i-1} by u_{i-1}, and the right-hand side r_{i-1} will be called f_{i-1}. This is done is order to facilitate the definition of the recursive procedure, but the meaning of the variables remains the same. Thus, we can state a *multigrid* procedure, described in Algorithm 2.7.1 on page 43.

This scheme can thus descend until the coarsest grid available (if it is so desired), or until we reach a level where the solution can be computed easily to obtain a desired accuracy. A compact representation of this scheme is in Figure 2.8 on page 44, and for obvious reasons this strategy is called a *V-cycle*.

Similarly, we can choose to take further advantage of coarser grids before bringing the correction back to the finest level, by repeating the coarse grid correction procedure *twice* at each grid, as represented in Figure 2.9 on page 44. In this case, this scheme is referred to as the *W-cycle*. These are the most common ways of exploiting levels since repeating the coarse grid correction procedure more than twice generally does not improve the convergence of the method.

Now, in order to exploit even more the different grids and the small cost of computing approximations at coarser grids, another idea is to use coarse grid approximations as starting points to a fine grid problem. This is the principle of the so-called *nested iteration* or *mesh refinement* scheme, and it can be outlined as Algorithm 2.7.2, on page 44.

Algorithm 2.7.1: V-cycle Scheme

$$[u_\ell] = \mathrm{V}(u_\ell, f_\ell).$$

- Apply ν_1 iterations of an iterative method to the problem $A_\ell u_\ell = f_\ell$ with initial guess $u_\ell = 0$.
- Compute the residual $r_\ell = f_\ell - A_\ell u_\ell$.
- Restrict the residual by defining

  $$f_{\ell-1} = R_\ell(r_\ell)$$

  and apply a coarse grid correction for this grid:
  - Apply ν_1 iterations of an iterative method to the problem $A_{\ell-1}u_{\ell-1} = f_{\ell-1}$ with initial guess $u_{\ell-1} = 0$.
  - Compute the residual $r_{\ell-1} = f_{\ell-1} - A_{\ell-1}u_{\ell-1}$.
  - Restrict the residual by defining $f_{\ell-2} = R_{\ell-1}(r_{\ell-1})$ and apply a coarse grid correction for this grid.

    $\vdots$

    - Solve $A_0 u_0 = f_0$.

    $\vdots$
  - Prolongate the correction $e_{\ell-2}$ by $e_{\ell-1} = P_{\ell-1}(e_{\ell-2})$.
  - Compute the new approximation $u_{\ell-1}^{\text{new}} = u_{\ell-1} + e_{\ell-1}$.
  - Apply ν_2 iterations of an iterative method to the problem $A_{\ell-1}u_{\ell-1} = f_{\ell-1}$, with initial guess $u_{\ell-1}^{\text{new}}$.
- Prolongate the correction $e_{\ell-1}$ by defining $e_\ell = P_\ell(e_{\ell-1})$.
- Compute the new approximation $u_\ell^{\text{new}} = u_\ell + e_\ell$.
- Apply ν_2 iterations of an iterative method to the problem $A_\ell u_\ell = f_\ell$ with initial guess u_ℓ^{new}.

By joining the mesh refinement and V-cycle ideas, we obtain the *Full Multigrid* scheme, describe in recursive form in Algorithm 2.7.3, on page 45.

The Full Multigrid (FMG) scheme is outlined in Figure 2.10.

One more element must be defined in what follows. We have assumed until now that the coarse grid operator $A_{\ell-1}$ is just the discretization of the problem in this coarser grid. In practice,

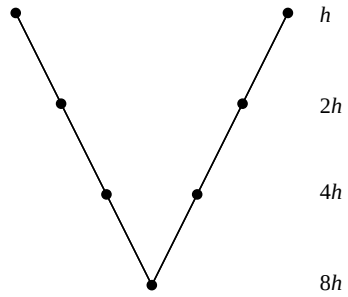


Figure 2.8: The V-cycle scheme on four levels.

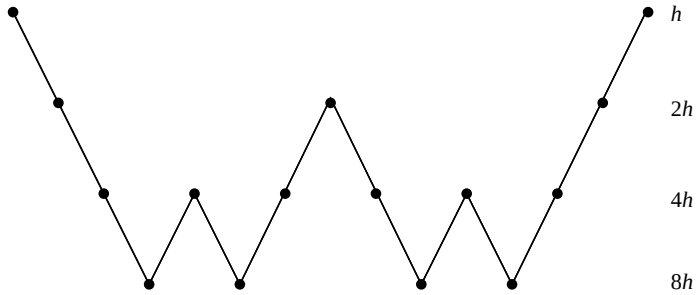


Figure 2.9: The W-cycle scheme on four levels.

Algorithm 2.7.2: Mesh Refinement Scheme

- Apply an iterative method ν_0 times to the coarse grid problem $A_0 u_0 = f_0$ with starting point $u_0 = 0$.
- Prolongate the approximation u_0 to obtain $u_1 = P_1 u_0$.
- Apply an iterative method ν_0 times to the problem $A_1 u_1 = f_1$ with initial guess u_1.

  ⋮

- Prolongate the approximation $u_{\ell-1}$ to obtain $u_\ell = P_\ell u_{\ell-1}$.
- Apply an iterative method ν_0 times to the problem $A_\ell u_\ell = f_\ell$ with initial guess u_ℓ.

it will be useful to assume that

$$A_{\ell-1} = R_\ell A_\ell P_\ell = \frac{1}{\sigma} P_\ell^T A_\ell P_\ell, \tag{2.19}$$

and this is called the *Galerkin* operator for the coarse grid. Together with (2.18), this property is important since it guarantees that, if A_ℓ is symmetric positive definite, so is $A_{\ell-1}$. Furthermore, it is vital in the convergence analysis of multigrid methods. We will not pursue this convergence theory in this thesis, but in Chapter 4 we will see that these properties are very useful.

Algorithm 2.7.3: Full Multigrid V-cycle

$$u_\ell = \text{FMG}(f_\ell)$$

Step 1. If we are at the coarsest grid, that is $\ell = 0$, go to Step 3 with initial guess $u_\ell = 0$. Else, restrict the right-hand side f_ℓ and apply the full multigrid procedure recursively:

$$[u_{\ell-1}] = \text{FMG}(f_{\ell-1}).$$

Step 2. Apply the correction to u_ℓ by prolongating $u_{\ell-1}$:

$$u_\ell = u_\ell + P_\ell u_{\ell-1}.$$

Step 3. Apply ν_0 iterations of an iterative method to $A_\ell u_\ell = f_\ell$.

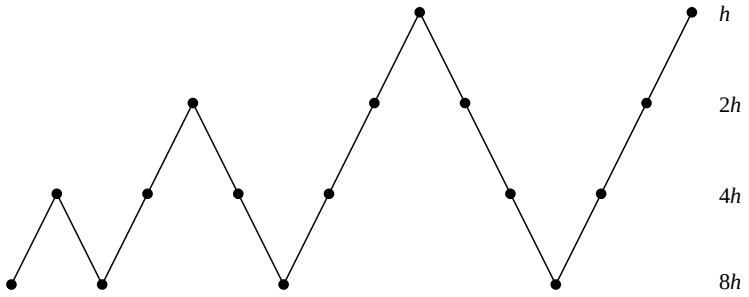


Figure 2.10: The Full Multigrid scheme on four levels.

2.8 What next?

There are two fundamental properties which are satisfied by the full multigrid procedure described in the previous section.

- The full multigrid procedure described in Algorithm 2.7.3 can be used to obtain an approximation u_r at the finest grid such that its error $\|u_* - u_r\|$ is of the same order as the discretization error, that is $O(h)$ for a one-dimensional problem, $O(h^2)$ for a two-dimensional problem.

- The full multigrid procedure is *asymptotically optimal*, which means that the computational cost needed to compute an approximate solution is proportional to the number of grid points at the finest grid.

These are extremely encouraging properties, and serve as a motivation for our work in this thesis. In a sense, for linear systems, in particular those associated with elliptic partial differential equations, we cannot do better than the full multigrid procedure. What we will see in the next chapters is that these ideas can be applied to optimization problems as well, and that

although the results are not quite as remarkable as they were for linear systems, it is worthwhile to explore the different discretizations of the problem when they are available.

Chapter 3

A Multilevel Algorithm for the Solution of the Trust-Region Subproblem

One of the most crucial points in the definition of a trust-region algorithm is the resolution of the trust-region subproblem, defined as (1.49). Indeed, the step computed by the solution of this minimization problem must satisfy certain conditions in order to guarantee the convergence of the algorithm, as seen in the theoretical results in Chapter 1.

More specifically, two choices define the step that will be computed as the solution of the trust-region subproblem. First, we must choose a norm to define the trust region. Then, we must choose a method that solves the minimization problem efficiently, such that the solution satisfies the Cauchy condition (1.56).

In this chapter, we will describe the most common solvers for the trust-region subproblem, and discuss the new method presented by Toint et al. [64]. For the most part of this chapter, we restrict our analysis to unconstrained problems and to methods where the trust region is defined using the ℓ_2 norm, that is,

$$\mathcal{B}_k = \{x_k + s \in \mathbb{R}^n \,|\, \|s\|_2 \leq \Delta_k\}.$$

For simplicity, we restate the ℓ_2-norm trust-region subproblem here without the iteration indices as

$$\min_{\|s\|_2 \leq \Delta} m(s) = \min_{\|s\|_2 \leq \Delta} \langle g, s\rangle + \frac{1}{2}\langle s, Hs\rangle \tag{3.1}$$

where $g = \nabla f(x_k)$ and H is the Hessian of f computed at x_k, or a symmetric approximation to this matrix.

In Section 1.4.1, we have seen that as long as the step computed by the solution of this subproblem satisfies the sufficient decrease condition (1.56), then by Theorem (1.4.2) it does not have to be the exact solution to the subproblem (3.1). Thus, it is interesting to analyze both exact and approximate methods, depending on the cost we are prepared to accept for the solution of this problem.

This Chapter is organized as follows. In Sections 3.1.1 and 3.1.2, we will take a look at the most common exact and approximate methods for solving the trust-region subproblem, respectively. Then, we will discuss the solution of the ℓ_∞-norm trust-region subproblem in Section 3.1.3. Finally, in Section 3.2 we will present our new multilevel method developed for the exact solution to the ℓ_2-norm trust-region subproblem.

3.1 The trust-region subproblem

The trust-region subproblem (3.1) consists of the minimization of a quadratic function with one constraint, the trust-region constraint. Therefore, from the KKT optimality conditions (1.23), stated in Chapter 1, we can obtain the following result.

Theorem 3.1.1 *A point s^M such that $\|s^M\|_2 \leq \Delta$ is a global minimizer for problem (3.1) if and only if*

$$H(\lambda^M)s^M = -g, \tag{3.2}$$

where

$$H(\lambda^M) \stackrel{\text{def}}{=} H + \lambda^M I$$

is positive semidefinite, $\lambda^M \geq 0$ and $\lambda^M(\|s^M\|_2 - \Delta) = 0$. If $H(\lambda^M)$ is positive definite, then s^M is unique.

This result is very important as it characterizes in an unusually simple way the exact solution of the trust-region subproblem, and it is the basis for the Moré-Sorensen method, which we present in the following section.

3.1.1 Finding the exact solution: the Moré-Sorensen method

Theorem 3.1.1 states that we have two possibilities for the solution of the trust-region subproblem. Either the unconstrained minimizer of m is in the interior of the trust region, and in this case $\lambda^{\mathrm{M}} = 0$, or the step s^{M} is in the boundary of the trust region, and $\lambda^{\mathrm{M}} > 0$.

Now, from Theorem 3.1.1, we can deduce that we want λ^{M} to be such that $\lambda^{\mathrm{M}} \geq -\lambda_{\min}$, where $\lambda_{\min} = \lambda_{\min}(H)$, the smallest eigenvalue of H. Furthermore, if $\lambda^{\mathrm{M}} > -\lambda_{\min}$, the minimizer of the model is unique, since in this case $H(\lambda^{\mathrm{M}})$ is positive definite. If we consider that s^{M} depends on λ^{M}, that is, λ is a parameter in this problem, we can rewrite this problem as finding λ such that

$$\|s(\lambda)\|_2 = \Delta, \tag{3.3}$$

where $s(\lambda)$ is the solution of the linear system (3.2). However, this equation has a pole in $-\lambda_{\min}$, and thus might prove to be difficult to solve in this region. Fortunately, there is an alternative formulation of this equation called the *secular equation*, defined as

$$\phi(\lambda) \stackrel{\text{def}}{=} \frac{1}{\|s(\lambda)\|_2} - \frac{1}{\Delta}. \tag{3.4}$$

This equation has a zero in $-\lambda_{\min}$, and is thus much better suited if we want to apply a root finding method to compute λ^{M}.

Since it is easy to compute the derivatives of this function, we can apply Newton's method, described in Chapter 1, Section 1.3.1, to find the solution of this problem. Now, for each iteration of this method, we must compute $\phi(\lambda)$ and its derivative, $\phi'(\lambda)$, and replace the old estimate λ^{old} with

$$\lambda^{\mathrm{new}} = \lambda^{\mathrm{old}} - \frac{\phi(\lambda^{\mathrm{old}})}{\phi'(\lambda^{\mathrm{old}})}.$$

Fortunately, both $\phi(\lambda)$ and its derivative can be found from the solution of linear systems of equations involving $H(\lambda)$. Since in the region of interest $H(\lambda)$ is positive definite, we can apply the Cholesky factorization, as described in Chapter 1, Section 1.2.1.4 to obtain

$$H(\lambda) = L(\lambda)L^T(\lambda).$$

Now, to compute $\phi(\lambda)$ it suffices to solve the linear system (3.2) to obtain $s(\lambda)$, and it can be shown that the derivative $\phi'(\lambda)$ can be written as

$$\phi'(\lambda) = \frac{\|w\|_2^2}{\|s(\lambda)\|_2^3}, \tag{3.5}$$

where w is the solution of the linear system

$$L(\lambda)w = s(\lambda).$$

The Cholesky factorization is a very important part of this method, because it also checks if $\lambda > -\lambda_{\min}$. Indeed, if this is not the case, the Cholesky factorization will fail, since we will encounter negative diagonal entries in L. Furthermore, we can easily detect if the solution to (3.1) is interior, as in this case the factorization of $H(0)$ succeeds, and the resulting step $s(0)$ is in the interior of the trust-region.

However, as mentioned in Chapter 1, Section 1.3.1, Newton's method does not always converge from any starting point. Thus, it is necessary to safeguard it somehow, so that it does not diverge. It is possible to show (see Theorem 7.3.4 in Conn et al. [12]) that if we can find an iterate between $-\lambda_{\min}$ and λ^{M}, convergence is guaranteed. Thus, we will estimate this interval by choosing a so-called *interval of uncertainty* $[\lambda^L, \lambda^U]$, where the solution is known to lie, and proceed by shrinking this interval at each iteration until we find a solution.

In order to define an initial interval of uncertainty, λ^L and λ^U must be chosen such that λ^{M} is guaranteed to be inside this interval. One way of doing this is using the Rayleigh quotient inequality (1.7) for $H(\lambda)^T H(\lambda)$ which , since $H(\lambda)$ is symmetric, gives us the following inequality:

$$(\lambda_{\min} + \lambda)^2 \leq \frac{\langle H(\lambda)s(\lambda), H(\lambda)s(\lambda)\rangle}{\langle s(\lambda), s(\lambda)\rangle} \leq (\lambda_{\max} + \lambda)^2.$$

Now, since we require that $\|s(\lambda)\|_2 \leq \Delta$ and since $H(\lambda)s(\lambda) = -g$, we can write

$$\frac{\|g\|_2}{\Delta} - \lambda_{\max} \leq \lambda \leq \frac{\|g\|_2}{\Delta} - \lambda_{\min}.$$

Now, we can replace $\lambda_{\min}$ and $\lambda_{\max}$ by any easily computable estimate, for example the Gershgorin bounds (1.8), or the norm estimates (1.16) and (1.17), obtaining initial bounds such as

$$\lambda^L = \max\left[0, -\min_i [H]_{i,i}, \frac{\|g\|_2}{\Delta} - \min\left[\max_i \left[[H]_{i,i} + \sum_{j\neq i} |[H]_{i,j}|\right], \|H\|_{\mathrm{F}}, \|H\|_\infty\right]\right].$$

and

$$\lambda^U = \max\left[0, \frac{\|g\|_2}{\Delta} + \min\left[\max_i \left[-[H]_{i,i} + \sum_{j\neq i} |[H]_{i,j}|\right], \|H\|_{\mathrm{F}}, \|H\|_\infty\right]\right].$$

Now, we can interpret λ as the convexity that we must add to the Hessian in order to bring the solution to the inside of the trust region. In practice, if we increase λ, $\|s(\lambda)\|_2$ decreases, and if we decrease λ, $\|s(\lambda)\|_2$ increases. This fact and the properties of the Newton iteration help us devise the following update for the interval $[\lambda^L, \lambda^U]$:

- If $\|s(\lambda)\|_2 < \Delta$, then we must decrease λ. Take $\lambda^U = \lambda$.
- If $\|s(\lambda)\|_2 > \Delta$, then we must increase λ. Take $\lambda^L = \lambda$.

We also know that, if $\lambda = 0$ is not the solution, then s^{M} must lie in the boundary of the trust region. However, as with any numerical method, testing if $\|s(\lambda)\|_2 = \Delta$ might be difficult. Thus, we define some $\epsilon^\Delta > 0$ and test if $\|s(\lambda)\|_2 \in [(1 - \epsilon^\Delta)\Delta, (1 + \epsilon^\Delta)\Delta]$. In this case, we have found the solution.

The complete procedure is stated in Algorithm 3.1.1.

Algorithm 3.1.1: The Moré-Sorensen Method

Step 1. If $H(0)$ is positive definite,i.e. if the factorization of $H(0)$ succeeds, and $\|s(0)\| \leq \Delta(1 + \epsilon^\Delta)$, terminate with $s = s(0)$.

Step 2. Determine an interval $[\lambda^L, \lambda^U]$ and an initial λ in this interval.

Step 3. Attempt a Cholesky factorization of $H(\lambda) = LL^T$. If this succeeds, solve

$$LL^T s = -g.$$

If $\Delta(1 - \epsilon^\Delta) \leq \|s\| \leq \Delta(1 + \epsilon^\Delta)$, i.e. if s is near the boundary of the trust region, terminate. If not s is not near the boundary, compute λ^{new} by

$$\lambda^{\mathrm{new}} = \lambda + \left(\frac{\|s\| - \Delta}{\Delta}\right)\left(\frac{\|s\|^2}{\|w\|^2}\right) \quad \text{where } w \text{ solves } \; Lw = s \tag{3.6}$$

Step 4. Update the interval $[\lambda^L, \lambda^U]$:

- if $\|s\| > \Delta(1 + \epsilon^\Delta)$, or if the factorization has failed, redefine $\lambda^L = \lambda$;
- if $\|s\| < \Delta(1 - \epsilon^\Delta)$, redefine $\lambda^U = \lambda$.

Step 5. Choose λ sufficiently inside $[\lambda^L, \lambda^U]$ and as close as possible to λ^{new}, if it has been computed. Go to Step 3.

There are many sophistications to this algorithm, in particular regarding the choice of the initial λ, that of a new λ in the interval when λ^{new} falls outside and suitable termination rules. We refer the interested reader to Sections 7.3.4 to 7.3.11 of Conn et al. [12] and to the more recent work of Dollar et al. [15] for details.

3.1.2 The approximate solution

While an exact solution is certainly the best solution, it might not be necessary when the cost of factorizing $H(\lambda)$ is too high. Since we have seen in Theorem 1.4.2 that we do not necessarily need the exact solution to the trust-region subproblem (3.1), we will describe here one of the most popular methods for the approximate solution of this problem.

In Section 1.3.3 in Chapter 1, we discussed the conjugate gradient method for the solution of strictly convex quadratic functions. Here, our problem is quadratic, but not necessarily convex. Fortunately, in the specific case of the ℓ_2-norm trust-region subproblem, a generalization of the conjugate gradient method that can be applied to non-convex problems is possible. Since the conjugate gradient method can be seen as a particular case of the preconditioned conjugate gradient method described in Algorithm 1.3.2, we will state here the more general preconditioned version of this method as well.

We will assume that we have a symmetric positive definite preconditioner M for this problem, as if this is not the case, we can just substitute M for the identity. In the general case, however, the preconditioned problem is equivalent to the trust-region subproblem defined in a $M-$norm, that is,

$$\min_{\|s\|_M \leq \Delta} m(s) = \langle g, s\rangle + \frac{1}{2}\langle s, Hs\rangle.$$

If we just apply the preconditioned conjugate gradient method to this problem, three possibilities can occur. First, if m is convex, that is, $\langle p_k, Hp_k\rangle > 0$ for every iteration k in the preconditioned conjugate gradient method[(1)], and all s_k remain inside the trust region, then it suffices to find the unconstrained minimizer of $m(s)$, without modifying the method. If $\langle p_k, Hp_k\rangle \leq 0$ for some k, in which case the function m is unbounded from below along the direction αp_k, we will find the smallest function value in the intersection of the line defined by $s_k + \alpha p_k$ and the trust-region boundary. Finally, if one of the iterates of the preconditioned conjugate gradient method lies outside the trust region, it is not clear what to do. One might think that leaving the PCG method run its course might result in a solution that is in the interior of the trust region, even if one of the iterates was found to be on the outside. However, this is not true, as the following result shows.

Theorem 3.1.2 *Suppose that the preconditioned conjugate gradient method described by Algorithm 1.3.2 is applied to the minimization of $m(s)$, with starting point $s_0 = 0$. Suppose furthermore that $\langle p_i, Hp_i\rangle > 0$ for $0 \leq i \leq k$. Then, the iterates s_j satisfy*

$$\|s_j\|_M < \|s_{j+1}\|_M,$$

for $0 \leq j \leq k-1$.

This result is due to Steihaug [62] and it is the key to our discussion. Since there is no use in pursuing the conjugate gradient method further after one of its iterates falls outside the trust region, we can find a local constrained minimizer of $m(s)$ by finding the intersection of the line defined by $s_k + \alpha p_k$ and the trust-region boundary, exactly as we did in the case of negative curvature.

[(1)] Here, s_k denotes the kth iteration of the preconditioned conjugate gradient method.

The complete *Truncated Conjugate Gradient* algorithm is described in Algorithm 3.1.2 on page 52. This method is also known as the Steihaug-Toint method, as Toint [63] was the first to suggest that the conjugate gradient method could be used in the trust-region subproblem context.

Algorithm 3.1.2: The Steihaug-Toint truncated conjugate gradient method

Let $s_0 = 0$, $g_0 = g$, $v_0 = M^{-1}g_0$ and $p_0 = -v_0$. For $k = 0, 1, \ldots$ until convergence, do:

- Set $\kappa_k = \langle p_k, Hp_k \rangle$.
- If $\kappa_k \leq 0$, compute σ_k as the positive root of $\|s_k + \sigma p_k\|_M = \Delta$, set $s_{k+1} = s_k + \sigma_k p_k$, and stop.
- Set $\alpha_k = \dfrac{\langle g_k, v_k \rangle}{\kappa_k}$.
- If $\|s_k + \alpha_k p_k\|_M \geq \Delta$, compute σ_k as the positive root of $\|s_k + \sigma p_k\|_M = \Delta$, set $s_{k+1} = s_k + \sigma_k p_k$ and stop.
- Set

$$\left\{ \begin{array}{rcl} s_{k+1} & = & s_k + \alpha_k p_k, \\ g_{k+1} & = & g_k + \alpha_k H p_k, \\ v_{k+1} & = & M^{-1} g_{k+1}, \\ \beta_k & = & \dfrac{\langle g_{k+1}, v_{k+1} \rangle}{\langle g_k, v_k \rangle}, \text{ and} \\ p_{k+1} & = & -v_{k+1} + \beta_k p_k. \end{array} \right.$$

One remarkable property of this algorithm is that if the condition number of M remains bounded over the sequence of subproblems approximately solved in the underlying trust-region algorithm, then the first iterate s_1 generated by Algorithm 3.1.2 is exactly the Cauchy point for the model, and thus already sufficient to guarantee convergence of the trust-region method. Of course, further iterations only improve on the decrease of the function and thus it is interesting to continue the iterations until some termination test is satisfied.

In the case when negative curvature is detected, or when an iterate is found to lie outside the trust region, we stop the algorithm immediately and the solution is the current iterate. When the solution of the trust-region subproblem lies inside the trust-region, though, we must decide on a stopping rule for the conjugate gradient method. In practice, it is usual to define a maximum number of iterates so that we do not spend too much time and computational cost in finding what will be an approximate solution anyway. This can be improved if we decide, for example, to stop as soon as we reach an iteration k for which the norm of the gradient has been reduced to a small fraction of its initial value. Even further improvement can be obtained by requiring

that the trust-region subproblem be solved with higher accuracy as the trust-region algorithm approaches a first-order critical point. Putting these conditions together, we may choose to stop as soon as

$$\|g_k\|_2 \leq \|g_0\|_2 \min[\kappa_{\text{fgr}}, \|g_0\|_2^{\theta}] \text{ or } k > k_{\max},$$

where $\kappa_{\text{fgr}} < 1$, $\theta > 0$ and $k_{\max} \geq 0$ is satisfied.

It is important to note here that, if we require the solution of (3.1) to also satisfy a set of affine constraints of the form $As = 0$, we can apply Algorithm 1.3.3 to this problem, so long as the preconditioner M is also used to define the trust-region norm. In this case, the iterates of the projected preconditioned conjugate gradient method all satisfy $As = 0$, and they also have the property of increasing in M norm. Thus, we can apply Algorithm 3.1.2 directly to

$$\begin{aligned} \min_{s \in \mathbb{R}^n} \quad & \langle g, s \rangle + \tfrac{1}{2}\langle s, Hs \rangle \\ \text{subject to} \quad & As = 0, \\ & \|s\|_M \leq \Delta, \end{aligned}$$

replacing, of course, the computation of v_{k+1} by the solution of the linear system (1.46).

3.1.3 The ℓ_∞-norm trust-region subproblem

As discussed earlier, the ℓ_2 norm is not the only norm possible in the definition of the trust region. Indeed, the ℓ_∞ norm can be very advantageous, and we will show in Chapter 4 that it has an important role to play in multilevel methods as well.

The first observation we can make about the ℓ_∞-norm trust-region subproblem is that it can easily be redefined as a bound-constrained problem, such that the condition that $\|s\|_\infty \leq \Delta$ is equivalent to

$$-\Delta \leq s_i \leq \Delta,$$

where s_i denotes the ith component of the vector $s \in \mathbb{R}^n$. Furthermore, this is especially interesting if the problem we are trying to solve is a bound-constrained problem, where the solution u must satisfy

$$l \leq x \leq u.$$

Indeed, the ℓ_∞-norm trust-region subproblem can then be written as

$$\max[l_i - x_i, -\Delta] \leq s_i \leq \min[u_i - x_i, \Delta].$$

Unfortunately, there are also a few disadvantages to this definition. One is that while the ℓ_2-norm trust-region subproblem can be solved rather easily, there are cases (namely, when H is indefinite) where the ℓ_∞-norm subproblem cannot be solved in polynomial time, that is, it is a NP-hard problem, meaning that there are no polynomial time algorithms known to solve it.

Another problem is that if a ℓ_∞-norm version of the truncated conjugate gradient method is used, Theorem 3.1.2 is not valid; it could be that an estimate of the solution computed by the conjugate gradient method is outside the trust region, but the solution is in the interior of the trust-region. Thus, we cannot use the same methods considered in Sections 3.1.1 and 3.1.2.

Fortunately, in practice, theoretically inefficient methods can be very useful, especially since the theory of trust-region methods only requires the Cauchy point to guarantee convergence, and this can be computed rather easily even for ℓ_∞-norm problems.

3.2 A Multilevel Moré-Sorensen Method (MMS)

Our objective in this section is to consider multilevel techniques for the exact solution of the Euclidean-norm trust-region subproblem. This particular subproblem typically arises when optimizing a nonlinear objective function whose variables are discretized continuous functions. This is for instance the case if the local Hessian is given by a discretized Laplacian or other elliptic operator. The trust-region problem is considered at the highest level (corresponding to the finest discretization), but the different levels of discretization provide a natural multilevel context. When the function is locally convex, one can very naturally consider applying a classical multigrid linear solver to the system (3.2), yielding a very efficient method to compute the step. However, things become much less clear when the objective function is locally nonconvex, in which case a suitable step is no longer given by Newton's equations, as $\lambda = 0$ is no longer a solution. The existing techniques for computing a step in this case, such as the one presented in Section 3.1.1 are well-known for small dimensional problems (see Hebden [31], Moré and Sorensen [44]), and guarantee, in most cases, that every limit point of the sequence of iterates is a second-order stationary point. However, these techniques are unfortunately very often impractical for large discretized problems because they involve factorizing a Hessian matrix defined on the fine grid. This is particularly limiting if one considers the discretization of variational problems in three dimensions or more. Our objective here is to propose two multilevel variants of this algorithm that are suitable for these large problems but nevertheless guarantee convergence to second-order limit points.

The idea of applying multigrid methods to our problem is that we try to solve the residual equation (2.15) not for $H(\lambda)$, but for some simpler approximation of this matrix in a lower dimensional space where smooth components of the error appear oscillatory.

As we did in Chapter 2, assume that we have a collection of full rank operators $R_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i-1}}$ and $P_i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ for $i = 1, \ldots, p$ (the *restriction* and the *prolongation*, respectively) such that $P_i = \sigma_i R_i^T$, with $\sigma_i > 0$, for all $i = 1, \ldots, p$. We will call each $i = 0, \ldots, p$ a *level*, with $n_p = n$ such that $H_p(\lambda) = H(\lambda)$. In this case, we can construct a simpler representation of the matrix as the *Galerkin operator* for $H_i(\lambda)$ defined by

$$H_{i-1}(\lambda) = R_i H_i(\lambda) P_i. \tag{3.7}$$

This operator is not the only choice possible. However, it has many interesting properties, such as keeping the $i-1$ level operator symmetric and positive definite, if that is the case for the original $H_i(\lambda)$, and maintaining the structure created by the discretization.

Once this is done, we may redefine the residual equation in the lower level. Given $s_{i,k}$, the step in the current level, and call the right hand side of the equation we want to solve in this level by $b_{i,k}$. We then compute $r_{i,k}$, the residual, by

$$r_{i,k} = b_{i,k} - H_i(\lambda) s_{i,k}.$$

The residual equation (2.15) at this level then takes the form

$$H_i(\lambda)e_{i,k} = r_{i,k}. \tag{3.8}$$

If we now restrict this equation to level $i-1$, the right-hand side at this level is now given by $R_i r_{i,k}$ and the residual equation at level $i-1$ becomes

$$H_{i-1}(\lambda)e_{i-1} = H_{i-1}(\lambda)R_i s_{i,k} - R_i r_{i,k} \stackrel{\text{def}}{=} r_{i-1,0}. \tag{3.9}$$

If the norm of this restricted residual is not large enough compared with the norm of the residual at level i, i.e. if $\|r_{i-1,0}\| < \kappa_{\text{r}}\|r_{i,k}\|$ for some $\kappa_{\text{r}} < 1$, then there is no advantage in trying to solve the lower level system. In this case, we perform smoothing iterations similar to those used in classical multigrid methods. Otherwise, if

$$\|r_{i-1,0}\| \geq \kappa_{\text{r}}\|r_{i,k}\|, \tag{3.10}$$

we then compute a solution e_{i-1} of the lower level residual equation (3.9). The corresponding upper level step can now be recovered by $s_{i,k+1} = s_{i,k} + P_i e_{i-1}$. This procedure can be applied recursively, in that the solution of the residual equation in level $i-1$ itself can be computed recursively. At the coarsest level, which corresponds to the smallest system and where recursion is no longer possible, the solution may be computed exactly, for instance by using matrix factorization.

3.3 The Multilevel Moré-Sorensen Algorithm

We now wish to develop an algorithm for the solution of (3.1) that follows the general pattern of the Moré-Sorensen method presented in Section 3.1.1 but which, at the same time, exploits the ideas and techniques of multigrid. If the problem is convex and the multiplier λ^{M} is known, we propose to use a multigrid solver for the system (3.2), thereby exploiting the hierarchy of level-dependent problem formulations described in the previous section. If the multiplier is not known, we also face, as in the standard Moré-Sorensen method, the task to find its value, again exploiting the multilevel nature of the problem.

Thus, in addition to the multigrid solution of (3.2), we must, as in Algorithm 3.1.1, find a new value of λ if the step computed as the solution of (3.2) does not satisfy our stopping conditions. Finding the value of λ^{M} may in practice be considered as a two-stages process. We first need to find a lower bound $\lambda^L \geq 0$ such that $H_p(\lambda)$ is positive-semidefinite for all $\lambda \geq \lambda^L$. Assuming that $\lambda^{\text{M}} = 0$ does not solve the problem, the second is then to determine $\lambda^{\text{M}} \geq \lambda^L$ such that

$$\|s_p(\lambda^{\text{M}})\|_2 = \|H_p(\lambda^{\text{M}})^{-1}g\|_2 = \Delta, \tag{3.11}$$

where we have simply rewritten (3.3) at level p, the topmost in our hierarchy. In our multigrid context, we intend to exploit the restriction of that problem on the ith level where

$$\|s_i(\lambda^*)\|_i = \|M_i H_p(\lambda^{\text{M}})^{-1}Q_i M_i g\|_i = \|H_i(\lambda^{\text{M}})^{-1}g_i\|_i = \Delta, \tag{3.12}$$

where, as in Gratton et al. [25],

$$M_i \stackrel{\text{def}}{=} \prod_{\ell=i+1}^{p} R_\ell, \qquad Q_i \stackrel{\text{def}}{=} \prod_{\ell=p}^{i+1} P_\ell, \qquad g_i = M_i g$$

and

$$\|x\|_i \stackrel{\text{def}}{=} \|Q_i x\|_2.$$

The linear system implicit in (3.12) is then solved using the multigrid technique discussed in the previous section.

3.3.1 Exploiting the Level Structure to Find Bounds on $\lambda^{\mathbf{M}}$

Consider ensuring positive-semidefiniteness of $H_p(\lambda)$ first. Our structure exploiting approach for this question is based on the simple observation that $H_i(\lambda)$ $(i = 2, \ldots, p)$ cannot be positive-semidefinite if $H_{i-1}(\lambda)$ is not, as expressed by the following property.

Lemma 3.3.1 *Let $P \in \mathbb{R}^{n_i \times n_{i-1}}$ be a full (column) rank matrix. If $\lambda_1^i \leq \ldots \leq \lambda_{n_i}^i$ are the eigenvalues of $A \in \mathbb{R}^{n_i \times n_i}$, and $\lambda_1^{i-1} \leq \ldots \leq \lambda_{n_{i-1}}^{i-1}$ are the eigenvalues of $RAP \in \mathbb{R}^{n_{i-1} \times n_{i-1}}$, where $R = \frac{1}{\sigma} P^T$ for some $\sigma > 0$, then we have that*

$$\lambda_1^{i-1} \geq \frac{\sigma_{\min}^2}{\sigma} \lambda_1^i, \tag{3.13}$$

where $\sigma_{\min}$ is the smallest singular value of P.

Proof. Using the extremal properties of eigenvalues (see, for instance, (1.7) or Golub and Van Loan [21]), we see that

$$\lambda_1^{i-1} = \min_{\substack{x \in \mathbb{R}^{n_{i-1}} \\ \|x\|_2 = 1}} \frac{\langle x, P^T A P x \rangle}{\sigma} = \min_{\substack{x \in \mathbb{R}^{n_{i-1}} \\ \|x\|_2 = 1}} \frac{\langle Px, APx \rangle}{\sigma} = \min_{\substack{y = Px \\ \|x\|_2 = 1}} \frac{\langle y, Ay \rangle}{\sigma}.$$

But, since $\|y\|_2 = \|Px\|_2 \geq \sigma_{\min}$, we obtain that

$$\lambda_1^{i-1} = \min_{\substack{y = Px \\ \|x\|_2 = 1}} \frac{\sigma_{\min}^2 \langle y, Ay \rangle}{\sigma \sigma_{\min}^2} \geq \min_{\substack{y = Px \\ \|x\|_2 = 1}} \frac{\sigma_{\min}^2 \langle y, Ay \rangle}{\sigma \|y\|_2^2} \geq \min_{y \in \mathbb{R}^{n_i}} \frac{\sigma_{\min}^2 \langle y, Ay \rangle}{\sigma \|y\|_2^2} = \frac{\sigma_{\min}^2}{\sigma} \lambda_1^i.$$

□

This property thus implies that the value of the multiplier needed to make $H_{i-1}(\lambda)$ convex provides a computable lower bound on that needed to make $H_i(\lambda)$ convex. In many cases of interest, the value of $\sigma_{\min}$ is known and larger that one. This is for instance the case when P is the linear interpolation operator in 1, 2 or 3 dimensions. However, the exact value depends on the level considered and is typically costly to compute accurately, which leads us to consider the simpler case where we only assume that $\sigma_{\min} \geq 1$, in which case (3.13) can be rewritten, at level i as

$$\lambda_1^{i-1} \geq \frac{\lambda_1^i}{\sigma_i}.$$

Once this lower bound is computed, the algorithm then proceeds to increase λ^L (in a manner that we describe below) if evidence of indefiniteness of $H_p(\lambda)$ is found. We have considered two ways to obtain this evidence. The first is to attempt to solve the system $H_p(\lambda)s = -g$ for the step at level p by a multigrid technique, and to monitor the curvature terms $\langle d, H_i(\lambda)d\rangle$ occurring in the smoothing iterations at each level i. As soon as one of these terms is shown to be negative, we know from Lemma 3.3.1 that the lower bound λ^L must be increased. The second is to use a multilevel eigenvalue solver like the Rayleigh Quotient Minimization Multigrid (RQMG) Algorithm (see Mandel and McCormick [42]) to compute λ_1^p, the smallest eigenvalue of H_p, associated with the eigenvector u_1^p. The RQMG algorithm solves the variational problem

$$RQ(u_1^p) = \min_{u\neq 0} RQ(u) = \min_{u\neq 0} \frac{\langle H_p u, u\rangle}{\langle u, u\rangle}$$

by transferring the problem to coarser levels, using as is usual for multigrid methods the Galerkin operator

$$H_i = R_{i+1}A_{i+1}P_{i+1} = \frac{1}{\sigma}P_{i+1}^T A_{i+1}P_{i+1}$$

at each level i, and applying a smoothing strategy (such as the Gauss-Seidel method) to the Rayleigh quotient equation at all levels except the coarsest one, where we minimize the Rayleigh quotient exactly in each coordinate direction. This last exact minimization is, in fact, equivalent to finding the roots of a quadratic polynomial, and thus the method is not too expensive computationally. The solution to this problem can thus be used as an (upper) approximation to λ_1^p which, if negative, may therefore be used to deduce the bound $\lambda^L \geq -\lambda_1^p$. Observe that the RQMG algorithm (applied with sufficient accuracy) ensures that $H_p(\lambda^L)$ is, at least in inexact arithmetic, positive semidefinite. This is useful since we will only be able to check for positive-semidefiniteness in levels $i > 0$ (that is, all but the coarsest level where the factorization is used) using the RQMG method.

In addition to the lower bound λ^L (which applies to all levels), we compute an initial upper bounds λ_i^U for each level i as in the Moré-Sorensen algorithm (observe that no information can be obtained from lower levels about λ_i^U). This therefore provides intervals $[\lambda^L, \lambda_i^U]$ for acceptable λ at each level i.

3.3.2 Updating λ in the Positive Definite Case

If $\lambda^L = 0$, $H_p(0)$ is positive-definite (in inexact arithmetic) and $\|s(0)\|_2 \leq \Delta$, our problem is solved. If this is not the case, our second task is then to adjust $\lambda \geq \lambda^L$ such that (3.11) holds. We now describe this adjustment procedure at level i, our final intention being to solve it at level p.

Since we are looking for λ that solves the secular equation (3.4), we can apply the Newton method to this end as we did in (3.6). However, in our case, the Cholesky factor L for $H(\lambda)$ is only available at the lowest level. Fortunately, note that

$$\|w\|^2 = \langle w, w\rangle = \langle L^{-1}s, L^{-1}s\rangle = \langle s, L^{-T}L^{-1}s\rangle = \langle s, (H(\lambda))^{-1}s\rangle.$$

Thus, if we compute y as the solution to the positive-definite system

$$H(\lambda)y = s(\lambda), \tag{3.14}$$

the Newton step for the secular equation at the current level then takes the form

$$\lambda^{\text{new}} = \lambda + \left(\frac{\|s\|_i - \Delta}{\Delta}\right)\left(\frac{\|s\|_i^2}{\langle s, y\rangle}\right). \tag{3.15}$$

Since we may not rely on factorizations for an exact solution of the system (3.14), we therefore apply a multigrid method to solve for w. However, this solution may be considered as costly. An alternative option is to update λ by applying a secant method to the secular equation, which gives

$$\lambda^{+} = \lambda - \phi(\lambda)\left(\frac{\lambda - \lambda_{\text{old}}}{\phi(\lambda) - \phi(\lambda_{\text{old}})}\right). \tag{3.16}$$

(We use $\lambda_{\text{old}} = \lambda^U$ to start the iteration.)

As in the Moré-Sorensen algorithm, if λ^{new} lies outside the interval, we choose λ inside the interval. One way to do this is to take λ^{new} as the half of the interval $[\lambda^L, \lambda^U]$, which corresponds to a simple bisection step. But we can expect better results by choosing to follow Moré and Sorensen [44] and setting

$$\lambda^{\text{new}} = \max\left[\sqrt{\lambda^L, \lambda^U}, \lambda^L + \theta(\lambda^U - \lambda^L)\right], \tag{3.17}$$

for $\theta \in (0, 1)$, which ensures that λ^{new} is closer to λ^L.

3.3.3 The Complete Algorithm

We need to introduce three further comments before the formal statement of the algorithm.

We first note that once a restricted trust-region problem (3.12) has been solved at level i, this means that the corresponding λ can be used as a lower bound for all higher levels. No further updating of λ is therefore necessary at this level and all lower ones, but we may nevertheless continue to exploit level i in the multigrid solution of the linear systems occurring at higher levels. The fact that a solution at level i has already been computed is remembered in our algorithm by setting the flag $\texttt{issolved}_i$. (For coherence, we define these flags for levels $1, \ldots, p+1$.)

Our second comment is that we still need to define stopping criteria for the multigrid solution of (3.12). A first criterion is obviously to terminate the iterations when the residual of the system is sufficiently small. In practice, we choose to stop the solution of the system as soon as

$$\|r_{i,k}\| = \|-g_i - H_i(\lambda)s_{i,k}\| \leq \epsilon^r,$$

with $\epsilon^r \in (0, 1)$. However, we might need to introduce a second stopping rule. It may indeed happen that, for a current λ (too small), the step resulting from the system has a i-norm exceeding Δ. It is of course wasteful to iterate too long to discover, upon termination, that we have to throw the solution away. In order to avoid this wasteful calculation, we exploit the fact that the norm of the multigrid iterates is typically increasing as the iterations proceed. Thus, if this norm exceeds Δ by some threshold D^{++}, we decide to terminate the iterative process (and subsequently increase λ). However, we must be careful not to alter the lower and upper bounds on λ in this subsequent update, because of the possible inaccuracy generated by the early truncation of the system and the absence of any monotonicity guarantee (at variance with methods like

truncated conjugate-gradients, see Steihaug [62]). Unfortunately, it is also possible that no λ in the current interval produces a sufficiently small step. In this case, λ grows and becomes arbitrarily close to its upper bound. We avoid this situation by increasing our threshold whenever λ is within ϵ_i^λ of λ_i^U.

Finally, we have to propagate changes in λ between levels. Thus, if we have just updated λ^+ and the old one was λ^-, we have that

$$H_i(\lambda^+) = H_i(\lambda^-) + (\lambda^+ - \lambda^-)M_iQ_i. \tag{3.18}$$

Similarly, taking into account that each residual at level ℓ is computed with respect to the linear system at level $\ell + 1$, we have that the residual at iteration k, level i, can be computed by the formula

$$r_{i,k} = -M_ig + \sum_{\ell=i+1}^{p} M_\ell H_\ell(\lambda)s_\ell, \tag{3.19}$$

where g is the gradient, which is the right-hand side of the linear system at the topmost level, and s_ℓ is the current step computed at each level $\ell = i + 1, \ldots, p$. By substituting (3.18) into (3.19), one may verify that the residual update satisfies

$$\begin{aligned} r_{i,k+1} &= -M_ig + \sum_{\ell=i+1}^{p} M_\ell H_\ell(\lambda^+)s_\ell \\ &= -M_ig + \sum_{\ell=i+1}^{p} M_\ell[H_\ell(\lambda^-) + (\lambda^+ - \lambda^-)M_\ell Q_\ell]s_\ell \\ &= r_{i,k} - (\lambda^+ - \lambda^-)\sum_{\ell=i+1}^{p} M_\ell^2 Q_\ell s_\ell. \end{aligned} \tag{3.20}$$

We now present the complete multigrid algorithm for the solution of the trust-region subproblem, the Multigrid Moré-Sorensen (MMS) Algorithm on 3.3.1 on the next page. Note that for each level i, we start by unsetting $\texttt{issolved}_i$.

Some comments on this algorithm are necessary at this point.

1. The algorithm is called form the virtual level $p + 1$, after an initialization phase which computes, once and for all and for every level, the values of $D^+ = (1+\epsilon^\Delta)\Delta$, $D^- = (1-\epsilon^\Delta)\Delta$ and $D^{++} = \sigma_i D^+$ for some $\epsilon^\Delta \in (0, 1)$. A level-dependent feasible interval $[\lambda^L, \lambda_i^U]$ is also computed at this stage. The (global) lower bound λ^L is set to the maximum between 0 and the opposite of the approximation of the most negative eigenvalue produced by the RQMG algorithm; the upper bound is calculated, for each level, exactly as for the Moré-Sorensen algorithm (see Conn et al. [12], page 192), using the appropriate restrictions of the gradient and Hessian to the considered level. An initial value of $\lambda \in [\lambda^L, \lambda_i^U]$ is finally computed using (3.17) before the call to MMS proper.

2. We may essentially identify Steps 0 to 5 as a classical multigrid solver for a linear system when $\texttt{issolved}_i$ is set. The remaining contain the update to the λ parameter, broadly following the Moré-Sorensen method.

Algorithm 3.3.1: [$s_{i,*}$,λ_i] = MMS(i, H_i, $r_{i,0}$,Δ, λ^L, λ, $s_{i,0}$, $\texttt{issolved}_i$)

Step 0. Initialization. Set $k = 0$.

Step 1. Iteration Choice. If $i = 1$, go to Step 3. Otherwise, if (3.10) fails, go to Step 4 (Smoothing iteration). Else, choose to go to Step 2 or to Step 4.

Step 2. Recursive Iteration. Call MMS recursively as follows:

$$[e_{i-1,*}, \lambda_{i-1}] = \text{MMS}(i-1, H_{i-1}, r_{i-1,0}, \Delta, \lambda^L, \lambda, 0_{i-1}, \texttt{issolved}_{i-1})$$

where $r_{i-1,0}$ is computed as in (3.9). Compute $s_{i,k+1} = s_{i,k} + P_i e_{i-1,*}$. If $\texttt{issolved}_i$ is unset, i.e. this is the first time we perform a recursive iteration at this level, set $\lambda^L = \lambda_{i-1}$, choose $\lambda \in [\lambda^L, \lambda_i^U]$ using (3.17), update $H_i(\lambda)$ using (3.18) and $r_{i,k+1}$ using (3.20) and set $\texttt{issolved}_i$. Go to Step 5.

Step 3. Exact Iteration. If $\texttt{issolved}_{i+1}$ is unset, call the Moré-Sorensen algorithm (3.1.1), returning with solution $[s_{i,*},\lambda_i]$ = MS($H_i(\lambda)$,$r_{i,0}$,Δ,ϵ^Δ), and set $\texttt{issolved}_i$. Otherwise, just solve the system $H_i(\lambda)s_{i,*} = r_{i,0}$ exactly by Cholesky factorization of $H_i(\lambda)$ and return with solution $(s_{i,*}, \lambda)$.

Step 4. Smoothing Iteration. Apply μ smoothing cycles on the residual equation (3.8) yielding $s_{i,k+1}$, set $r_{i,k+1} = r_{i,k} + H_i(\lambda)(s_{i,k+1} - s_{i,k})$ and go to Step 5.

Step 5. Termination. If $\|r_{i,k+1}\| < \epsilon^r$ and $\texttt{issolved}_{i+1}$ is set, return $s_{i,k+1}$ and λ. Else, go to Step 1 if $\texttt{issolved}_{i+1}$ is set or if $\|r_{i,k+1}\| \geq \epsilon^r$ and $\|s_{i,k+1}\|_i \leq D^{++}$.

Step 6. Parameter update after full system solution.
If $\|r_{i,k+1}\| < \epsilon^r$ (and $\texttt{issolved}_{i+1}$ is unset),

Step 6.1: step threshold update. If $\lambda_i^U - \lambda < \epsilon_i^\lambda$, set $D^{++} = 2D^{++}$.

Step 6.2: interior solution test. If $\lambda = 0$ and $\|s_{i,k+1}\|_i < D^+$, or if $\lambda \geq 0$ and $D^- \leq \|s_{i,k+1}\|_i \leq D^+$, return with solution $s_{i,*} = s_{i,k+1}$ and $\lambda_i = \lambda$.

Step 6.3: parameter and interval updates. If $\|s_{i,k+1}\|_i > D^+$, set $\lambda^L = \lambda$. If $\|s_{i,k+1}\|_i < D^-$, set $\lambda_i^U = \lambda$. Compute a new $\lambda \in [\lambda^L, \lambda_i^U]$ using (3.15) or (3.16).

Step 6.4: reset the step. Set $s_{i,k+1} = 0$, $r_{i,k+1} = r_{i,0}$, update $H_i(\lambda)$ using (3.18), and go to Step 1.

Step 7: Parameter update after incomplete system solution.
If $\|r_{i,k+1}\| \geq \epsilon^r$ (and $\|s_{i,k+1}\|_i > D^{++}$),

Step 7.1: parameter update. compute a new $\lambda \in [\lambda, \lambda_i^U]$ using (3.15) or (3.16).

Step 7.2: reset the step. Set $s_{i,k+1} = 0$, $r_{i,k+1} = r_{i,0}$, update $H_i(\lambda)$ using (3.18), and go to Step 1.

3. As explained in Chapter 2, the linear system (3.2) is solved by computing a correction at coarse levels to the steps already computed at finer ones. Our restriction strategy produces an algorithm analog to the application, in our nonlinear context, of the Full Multigrid scheme.

4. We have not specified the details of the smoothing procedure in Step 4. In our experiments, we have used the Gauss-Seidel smoother, a classic in multigrid solvers (see Section 2.3.2 or Briggs et al. [3], page 10).

3.4 Numerical Experience

In this section we present some numerical results obtained by two variants of the MMS method applied in a trust-region algorithm (Algorithm 1.4.1 on page 23) for the four test problems described in Section A.1 involving three-dimensional discretizations. Some of these problems were also tested in Gratton et al. [26] in their two-dimensional formulation. All problems presented here are defined on the unit three-dimensional cube S_3 and tested with a fine discretization of 63^3 variables, and we used 4 levels of discretization. The Laplacian operator is obtained from the classical 7-points pencil. The prolongation operator is given by linear interpolation, and the restriction as its normalized (in the $\|\cdot\|_1$ norm) transpose, thereby defining $\sigma_i = \|P_i\|_1$. We briefly review these test problems below.

3.4.1 Numerical Results

We discuss here results obtained by applying the simple BTR trust-region method described in Algorithm 1.4.1 for the minimization of four problems described in Section A.1 of the Appendix, in which the subproblem is solved[(2)] at each iteration by one of three multigrid variants of the Moré-Sorensen algorithm. The first variant (MMS-secant) is the the MMS algorithm presented in this paper, where we use the secant approach (3.16) to solve the secular equation. The second (MMS-Newton) is the same method, but using Newton's method (3.15) instead of (3.16). The third (naive MMS-secant) is a simpler version of MMS-secant in which we do not use information on λ from lower levels. In this variant, we solve the Moré-Sorensen system (3.2) by multigrid instead of using Cholesky factorization of the Hessian, but we only change λ at the topmost level. This is equivalent to setting $\texttt{issolved}_i$ for all levels $i < p + 1$. We update λ by using the secant method on the secular equation, as described above. All runs were performed in Matlab v.7.1.0.183 (R14) Service Pack 3 on a 3.2 GHz Intel single-core processor computer with 2 Gbytes of RAM, using the parameters

$$\mu = 5, \qquad \epsilon^{\Delta} = 0.1, \qquad \epsilon^{r} = 10^{-6}, \qquad \theta = 10^{-4}, \quad \text{and} \quad \epsilon_i^{\lambda} = 0.01|\lambda_i^U - \lambda^L|.$$

Our results are shown in Table 3.1. In this table, $\#\lambda$ stands for the weighted number of λ-updates, where each update is weighted proportionally to the dimension of the subspace in which the update is performed. Similarly, $\#R$ stands for the weighted number of restrictions

[(2)] We require the Euclidean norm gradient of the objective function to be at most 10^{-6} for termination.

performed by the algorithm. This last number indicates how many recursive iterations were used to find the solution of the linear system over the course of optimization. The CPU reported (in seconds) is the average trust-region subproblem solution time over all optimization iterations.

	Naive MMS-secant			MMS-secant			MMS-Newton		
	$\#\lambda$	CPU	$\#R$	$\#\lambda$	CPU	$\#R$	$\#\lambda$	CPU	$\#R$
3D-1	17.3	5.7 (5)	65.6	9.2	4.8 (5)	46.4	7.5	10.5 (5)	34.1
3D-2	17.2	6.1 (6)	85.8	11.2	5.3 (7)	72.8	11.1	14.8 (6)	59.4
C-D	17.3	6.9 (6)	73.3	8.6	5.4 (6)	53.6	7.9	11.9 (6)	42.0
3D-BV	41.0	434.4 (18)	474.4	23.9	465.4 (20)	279.2	30.3	452.7 (18)	13.6

Table 3.1: Results for three variants of the MMS method.

These results clearly demonstrate that the MMS-secant version of our algorithm performs much better than the naive version in terms of the number of λ-updates required to solve all the trust-region subproblems in an optimization run. The conclusion in terms of CPU time remains favorable for MMS-secant, even if care must be exercised here given the inaccuracy of the Matlab timer. This suggests that information obtained at lower levels is, in fact, useful for the solution of the problem and should therefore be exploited. We also note that MMS-Newton does not offer a significant advantage over MMS-secant. Even if less λ-updates are needed to find the solution, these updates are computationally much more expensive than the simple secant ones since a linear system must be solved by multigrid for in each update, resulting in an overall slower algorithm. It is also important to note that the last problem is non-convex, and thus requires much more time to be solved. This is also due to the fact that, in this case, we have to compute an initial λ^L using an estimate of the smallest eigenvalue of the Hessian in each BTR iteration by means of the RQMG algorithm, as discussed in the previous section. We note again that this is not needed in other strategies based in factorization methods (as is the case in Dollar et al. [15]) since the factorization itself is able to detect indefiniteness.

This new method for the exact solution of the trust-region subproblem is suitable for large scale systems where the Moré-Sorensen method cannot be applied, for instance because factorizations are too costly or impossible. This method exploits the multigrid structure in order to extract curvature information from the coarse levels to speed up the computation of the Lagrange parameter associated with the subproblem.

We have presented some admittedly limited numerical experience, which shows the potential for the new method, both because it demonstrates that sizable three-dimensional applications can be considered and because it outperforms a too naive multigrid implementation of the basic Moré-Sorensen algorithm.

Chapter 4

Recursive Multilevel Trust-Region Methods

Despite the many advantages of trust-region methods already mentioned here and in the literature, it is clear that applications demand the ability to solve bigger problems every day. In Chapter 3, we have described one possibility for the exploitation of this structure that is inherent to the problem in a trust-region framework.

However, the application of the multigrid philosophy to the solution of the trust-region subproblem is not the only possible way of exploiting this multilevel structure. Recently, several authors have proposed methods that take multilevel hierarchies into account for the solution of optimization problems, such as Fisher [17], Nash [47], Lewis and Nash [40, 41], and Oh et al. [51]. Kornhuber [35, 36, 37] also developed a method of this type for possibly non-smooth convex bound-constrained problems in the finite-element context. Convergence of this multigrid method is ensured by the successive minimization along coordinate directions generated in Gauss-Seidel-like smoothers, thereby avoiding the need of explicit globalization.

On the other hand, the recursive Euclidean-norm trust-region algorithm for general multilevel unconstrained nonconvex minimization provides the first globally convergent framework for the application of multigrid-type mechanisms to this class of problems. Moreover, the numerical experiments with this algorithm are very good (see Gratton et al. [26]).

In this chapter, we will describe two versions of the Recursive Multilevel Trust-Region class of methods for nonlinear optimization problems. First, in Section 4.1, we briefly discuss the first multilevel strategy for the trust-region method, presented in Gratton et al. [25], which is essentially different from the strategy presented in Chapter 3. Then, in Section 4.2 we present a complete description of the ℓ_∞-norm version of the recursive multilevel trust-region method for both unconstrained and bound-constrained problems, and in Section 4.2.2 we present the convergence results obtained for this method, first presented in Gratton et al. [24]. Finally, we cite some of the numerical results that one might expect to obtain with this method in practice, presented originally by Gratton et al. [26].

4.1 The Recursive Multilevel Trust-Region Method in Euclidean Norm

Here, we are interested in the solution of problems of the type (1.1). This problem is viewed as an accurate representation of a more general underlying problem (such as, for instance, a contact problem in infinite dimensions). Since our interest is in the multilevel case, we also suppose, like we did in Section 3.2 in Chapter 3, that we know a set of functions $\{f_i\}_{i=0}^r$ which give alternative and potentially less accurate descriptions of the same underlying problem. Each of these f_i is assumed to be itself a twice continuously differentiable function from $\mathbb{R}^{n_i}$ to $\mathbb{R}$ (with $n_i \geq n_{i-1}$), $n_r = n$ and $f_r(x) = f(x)$ for all $x \in \mathbb{R}^n$. Each of these descriptions is said to define a *level*, which we index by i. As we did before, we also assume that, for each $i = 1, \ldots, r$, there exists an operator $R_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i-1}}$ (the *restriction*) and another operator $P_i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ (the *prolongation*) such that

$$R_i^T = \sigma_i P_i \tag{4.1}$$

for some known constant $\sigma_i > 0$. The prolongations and restrictions therefore define a hierarchy of levels, from lowest ($i = 0$) to highest ($i = r$).

As mentioned in Chapter 1, classical trust-region methods are based on a quadratic Taylor's model for f, given by

$$m_k(x_k + s) = f(x_k) + \langle g_k, s_k \rangle + \frac{1}{2}\langle s, H_k s \rangle. \tag{4.2}$$

In this case, since the solution of the problem at each level i is assumed to be more costly than the solution of the problem at level $i-1$, the idea is that we use f_{i-1} to build a model h_{i-1} for f_i. Thus, at iteration k at level i (with iterate $x_{i,k}$), if we choose to use the coarser-level model h_{i-1}, we must first restrict the iterate to obtain a starting point at level $i-1$ by defining

$$x_{i-1,0} = R_i x_{i,k}.$$

Then, we define the coarser-level model as

$$h_{i-1}(x_{i-1,0} + s_{i-1}) \stackrel{\text{def}}{=} f_{i-1}(x_{i-1,0} + s_{i-1}) + \langle v_{i-1}, s_{i-1} \rangle \tag{4.3}$$

where

$$v_{i-1} = R_i g_{i,k} - \nabla f_{i-1}(x_{i-1,0}),$$

with $g_{i,k} \stackrel{\text{def}}{=} \nabla h_i(x_{i,k})$ (and where $v_r \stackrel{\text{def}}{=} 0$).

The definition of h_{i-1} then enforces the relation

$$g_{i-1,0} = R_i g_{i,k}, \tag{4.4}$$

which is a property that will play a crucial role in the convergence theory of this method, and which ensures that the first-order behavior of h_i and h_{i-1} around $x_{i,k}$ is coherent. Furthermore, it implies that, if

$$s_i = P_{i-1} s_i, \tag{4.5}$$

then using (4.1) and (4.4)[(1)], we have that

$$\langle g_{i,k}, s_i\rangle = \langle g_{i,k}, P_i s_{i-1}\rangle = \langle R_i g_{i,k}, s_{i-1}\rangle = \langle g_{i-1,0}, s_{i-1}\rangle.$$

Thus, when entering level $i = 0, \ldots, r$, we must minimize h_i starting from $x_{i,0}$. At iteration k of this process, we can choose a model between (4.3) and

$$m_{i,k}(x_{i,k} + s_i) = h_i(x_{i,k}) + \langle g_{i,k}, s_i\rangle + \frac{1}{2}\langle s_i, H_{i,k} s_i\rangle, \tag{4.6}$$

which is just the rewriting of the usual Taylor model (4.2) in our multilevel context, and where $H_{i,k}$ approximates the second derivatives of h_i (which are also the second derivatives of f_i) at $x_{i,k}$.

Once the model is chosen, we compute a step $s_{i,k}$ that satisfies a sufficient decrease condition within a trust region defined by

$$\mathcal{B}_{i,k} \stackrel{\text{def}}{=} \{s_i \,|\, \|s_i\|_i \le \Delta_{i,k}\},$$

for some trust-region radius $\Delta_{i,k} > 0$, where the norm $\{\cdot\}_i$ is defined for some symmetric positive definite matrix M_i at each i as

$$\|s_i\|_i \stackrel{\text{def}}{=} \sqrt{\langle s_i, M_i s_i\rangle} = \|s_i\|_{M_i}.$$

If we choose model (4.6), it suffices to use one of the trust-region subproblem solving methods described in Chapter 3 to obtain $s_{i,k}$, which will then satisfy the sufficient decrease condition

$$m_{i,k}(x_{i,k} - m_{i,k}(x_{i,k} + s_{i,k}) \ge \kappa_{\text{red}} \|g_{i,k}\|_2 \min\left[\frac{\|g_{i,k}\|_2}{1 + \|H_{i,k}\|_2}, \Delta_{i,k}\right] \tag{4.7}$$

for some constant $\kappa_{\text{red}} \in (0, 1)$. This is, again, just (1.55) rewritten in a multilevel context.

On the other hand, if the model h_{i-1} is chosen, the minimization of this model yields a new point $x_{i-1,*}$ which is then prolongated into level i through (1.55). By defining

$$M_{i-1} \stackrel{\text{def}}{=} P_i^T M_i P_i \text{ for all } i,$$

we have then that

$$\|s_i\|_i = \|s_{i-1}\|_{i-1}$$

and thus the trust region at level $i - 1$ is defined by

$$\|x_{i-1,*} - x_{i-1,0}\|_{i-1} \le \Delta_{i,k}.$$

The coarser-level trust-region subproblem is then defined as

$$\min_{\|s_{i-1}\|_{i-1} \le \Delta_{i,k}} h_{i-1}(x_{i-1,0} + s_{i-1}). \tag{4.8}$$

[(1)]This gives us a hint as to why condition (4.1) was important also for multigrid methods in Chapter 2.

Some care must be taken in order to exploit this coarse-level definitions as it might be the case, for example, that $R_i g_{i,k}$ is zero, even if $g_{i,k}$ is not. Thus, it is useful to require that we only use the coarse level $i-1$ if

$$\|R_i g_{i,k}\|_2 \geq \kappa_g \|g_{i,k}\|_2 \text{ and } \|R_i g_{i,k}\|_2 > \epsilon_{i-1}^g, \tag{4.9}$$

for constants $\kappa_g \in (0, \min[1, \min_i \|R_i\|_2])$ and $\epsilon_{i-1}^g \in (0,1)$.

We describe the recursive multilevel trust-region method as defined in Gratton et al. [25] in Algorithm 4.1.1. The constants η_1, η_2, γ_1 and γ_2 are defined as in 1.51 on page 23. We assume that an initial trust-region radius $\Delta_i^s > 0$ is known for each level i, as well as $\epsilon_i^g \in (0,1)$ and trust-region tolerances $\epsilon_i^\Delta \in (0,1)$. For coherence, we consider the algorithm to be called from a virtual $r+1$-st level where $\Delta_{r+1,0} = \infty$.

The test (4.10) and the requirement (4.12) are imposed as a way to maintain the iterates inside the fine-level trust region. These two requirements can end up being overly restrictive, and will be one of the main motivations for our ℓ_∞-norm version of the RMTR method. Another motivation is the fact that computing the norm-matrices M_i and the norms $\|\cdot\|_i$ at each level i might be rather expensive.

Despite these possible problems, this method enjoys a complete global first-order convergence theory that includes a bound on the number of iterations that are required to find an approximate critical point of the objective function within a prescribed accuracy.

4.2 The ℓ_∞-norm Recursive Multilevel Trust-Region Method

While theoretically satisfying and practically acceptable, the choice of the Euclidean norm for the trust region definition is not without drawbacks. Firstly, the Euclidean trust regions do not mix naturally with bound-constrained problems, because the intersection of the trust region (a Euclidean ball) with the feasible domain for bounds (a box) has a complicated structure. Moreover, the combination of Gauss-Seidel-like smoothing iterations with the Euclidean trust region is unnatural because the smoothing steps consider one variable at a time and are therefore aligned with the coordinate directions. In addition, more technical complications also arise from the fact that the step at a lower level must at the same time be included in the current-level trust region and be such that its prolongation at higher level(s) is included in the higher level(s) trust region(s). As discussed in Gratton et al. [25], this double requirement implies the use of computationally expensive preconditioners and a special technique for updating the trust region radii which in turn sometimes inefficiently limits the step size.

In order to allow for bound constraints and avoid these technical difficulties, an alternative multilevel algorithm for bound-constrained optimization can be defined using the ℓ_∞ norm for the trust region definition. Moreover, smoothing iterations which explore directions aligned with the coordinate vectors can be easily integrated.

Unfortunately, the convergence theory presented in Gratton et al. [25] for RMTR cannot be applied to this case without significant modifications, not only because of the possible presence

Algorithm 4.1.1: RMTR(i, $x_{i,0}$, $g_{i,0}$, Δ_{i+1}, $\epsilon_i^{\mathbf{g}}$, ϵ_i^{Δ}, $\Delta_i^{\mathbf{s}}$)

Step 0: Initialization. Compute $v_i = g_{i,0} - \nabla f_i(x_{i,0})$ and $h_i(x_{i,0})$. Set $\Delta_{i,0} = \min[\Delta_i^{\mathrm{s}}, \Delta_{i+1}]$ and $k = 0$.

Step 1: Model choice. If $i = 0$ or if (4.9) is not satisfied, go to Step 3. Otherwise, choose to go to Step 2 (recursive step) or to Step 3 (Taylor step).

Step 2: Recursive step computation. Call Algorithm RMTR($i-1$, $R_i x_{i,k}$, $R_i g_{i,k}$, $\Delta_{i,k}$, ϵ_i^{g}, ϵ_i^{Δ}, $\Delta_{i-1}^{\mathrm{s}}$), yielding an approximate solution $x_{i-1,*}$ of (4.8). Then define $s_{i,k} = P_i(x_{i-1,*} - R_i x_{i,k})$, set $\delta_{i,k} = h_{i-1}(R_i x_{i,k}) - h_{i-1}(x_{i-1,*})$ and go to Step 4.

Step 3: Taylor step computation. Choose $H_{i,k}$ and compute a step $s_{i,k} \in \mathbb{R}^{n_i}$ that sufficiently reduces the model $m_{i,k}$ given by (4.6) in the sense of (4.7) and such that $\|s_{i,k}\|_i \leq \Delta_{i,k}$. Set $\delta_{i,k} = m_{i,k}(x_{i,k}) - m_{i,k}(x_{i,k} + s_{i,k})$.

Step 4: Acceptance of the trial point. Compute $h_i(x_{i,k} + s_{i,k})$ and define

$$\rho_{i,k} = \frac{h_i(x_{i,k}) - h_i(x_{i,k} + s_{i,k})}{\delta_{i,k}}.$$

If $\rho_{i,k} \geq \eta_1$, then define $x_{i,k+1} = x_{i,k} + s_{i,k}$. Otherwise, define $x_{i,k+1} = x_{i,k}$.

Step 5: Termination. Compute $g_{i,k+1}$. If $\|g_{i,k+1}\|_2 \leq \epsilon_i^{\mathrm{g}}$ or

$$\|x_{i,k+1} - x_{i,0}\|_i > (1 - \epsilon_i^{\Delta})\Delta_{i+1}, \tag{4.10}$$

then return with the approximate solution $x_{i,*} = x_{i,k+1}$.

Step 6: Trust-region radius update. Set

$$\Delta_{i,k}^{+} = \begin{cases} [\Delta_{i,k}, +\infty) & \text{if } \rho_{i,k} \geq \eta_2, \\ [\gamma_2 \Delta_{i,k}, \Delta_{i,k}] & \text{if } \rho_{i,k} \in [\eta_1, \eta_2), \\ [\gamma_1 \Delta_{i,k}, \gamma_2 \Delta_{i,k}] & \text{if } \rho_{i,k} < \eta_1, \end{cases} \tag{4.11}$$

and

$$\Delta_{i,k+1} = \min\left[\Delta_{i,k}^{+}, \Delta_{i+1} - \|x_{i,k+1} - x_{i,0}\|_i\right]. \tag{4.12}$$

of bounds, but also because the algorithm analyzed in these references is itself very dependent on the choice of the Euclidean norm. Our second purpose is thus to prove global convergence of the new algorithm to first-order critical points, that is convergence from arbitrary starting points to limit points satisfying the first-order optimality conditions.

4.2.1 The problem and algorithm

In what follows, we wish to solve the bound-constrained minimization problem

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x), \\ \text{such that} \quad & l \le x \le u \end{aligned} \tag{4.13}$$

where the $\le$ signs (here and below) are understood component-wise, where l and u are vectors in $\mathbb{R}^n$ such that $l \le u$, and where $f : \mathbb{R}^n \to \mathbb{R}$ is a twice continuously differentiable function which is bounded below on the feasible set $\{x \in \mathbb{R}^n \mid l \le x \le u\}$.

Trust-region methods compute a step by minimizing a model of the objective function in the trust region, but since we are dealing with bound-constrained problems, we also choose to keep our iterates feasible throughout the process, which implies that the model minimization must take place in the intersection of the feasible set of (4.13) with the trust region. As already mentioned in Section 3.1.3 in Chapter 3, since the feasible set is a box, it is much simpler to define the trust region with the ℓ_∞ norm. Thus, in the classical (single level) case, the step from the iterate x_k (at iteration k) is thus obtained from the (possibly approximate) solution of the subproblem

$$\begin{aligned} \min_{s\in\mathcal{B}_k} \quad & m_k(x_k + s), \\ \text{such that} \quad & l \le x_k + s \le u \end{aligned}$$

where $m_k(x_k + s)$ is the objective function's model around x_k and the trust region $\mathcal{B}_k$ is defined as

$$\mathcal{B}_k = \{s \in \mathbb{R}^n \,|\, \|s\|_\infty \le \Delta_k\},$$

and where the feasible set is defined as

$$\mathcal{F} \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n \mid l \le x \le u\}.$$

We then minimize the (potentially nonquadratic) model f_{r-1} *using a trust-region algorithm* at level $r-1$, whose iteration ℓ therefore features its own box-shaped trust-region $\mathcal{B}_{r-1,\ell}$. This minimization is carried under a set of constraints inherited from level r and from the initial point $x_{r-1,0} = R_r x_{r,k}$, until some approximate constrained minimizer $x_{r-1,*}$ is found. The resulting step is then prolongated to level r by computing

$$s_{r,k} = P_r(x_{r-1,*} - x_{r-1,0}).$$

The main difficulty is to specify the form of the constraints inherited from the upper level. First of all, the resulting feasible set (at the lower level) must be a box in order to preserve the coherence and efficiency of the algorithm across levels. We also wish to guarantee the feasibility *at the upper level* of the prolongated trial point $x_{r,k} + s_{r,k}$ with respect to the bound constraints. Finally, we would like to ensure that this trial step lies within the upper-level trust region $\mathcal{B}_{r,k}$. Unfortunately, *the prolongation of the restriction of a box at level r back to level r is in general* not *included in the original box*, as shown in Figure 4.1.

We are thus forced to alter our technique for representing an upper-level box at the lower level if we insist that its prolongation satisfies the constraints represented by the upper-level

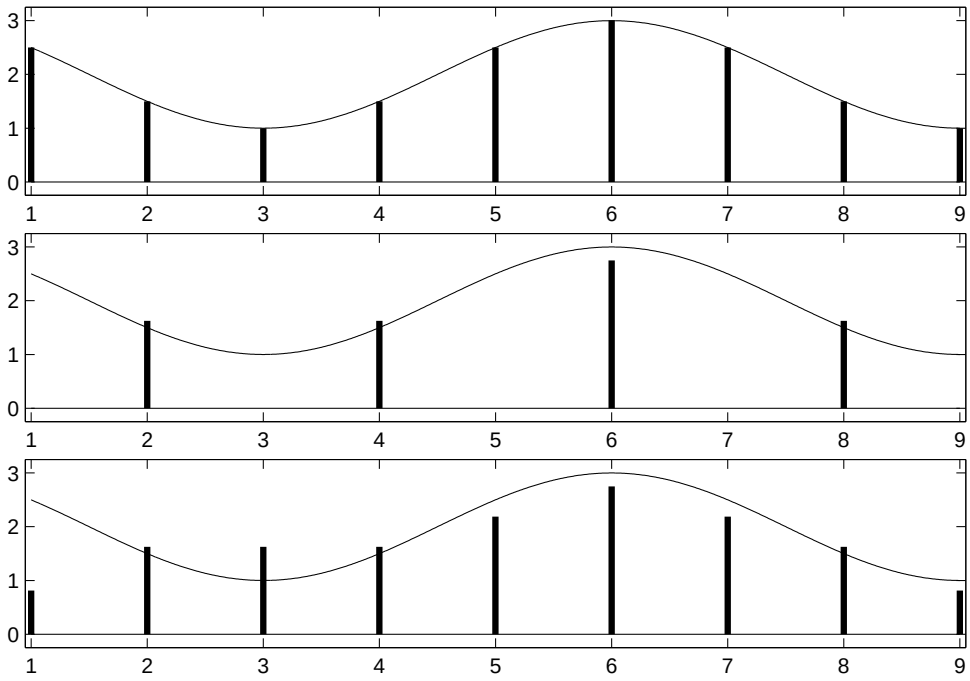


Figure 4.1: Prolongation and restriction of bounds. In this figure, one considers the set of continuous functions $\phi(t)$ for $t \in [1, 9]$ with a zero lower bound and an upper bound given by $2 + \cos(\pi t/3)$. The vertical bars in the upper graph show the possible ranges for the values $\phi(1), \ldots, \phi(9)$ for such functions, considered here as problem variables. The vertical bars in the middle graph show the ranges obtained by applying the restriction operator (corresponding to the normalized transpose of the linear interpolation for a coarser grid of 4 discretization points) to the set of bounds obtained in the upper graph. The vertical bars in the lower graph finally correspond to applying the prolongation (linear interpolation) to the bounds obtained in the middle graph. One notices that these latter ranges are *not* always included in the original ranges of the upper graph.

box. This is highly desirable for the upper-level box $\mathcal{F}_r$ defining the original bound constraints of the problem, because we wish to preserve feasibility at all levels. On the other hand, we might accept some flexibility for the lower-level box corresponding to the upper-level trust region $\mathcal{B}_{r,k}$, because one expects that a step whose norm is proportional to the trust-region size would be enough to ensure convergence (even if strict inclusion does not hold) without being unduly restrictive. Thus we are lead to a two-pronged strategy, where we separately represent, on one hand, the bound constraints at the lower level in a way guaranteeing feasibility of the prolongated step, and, on the other hand, the upper trust region, possibly more loosely.

If $\mathcal{F}_{r-1}$ is the representation of the bound constraints at the lower-level and $\mathcal{A}_{r-1}$ that of the upper trust region, then the step at iteration ℓ of the lower-level minimization must be included in the box

$$\mathcal{W}_{r-1,\ell} \stackrel{\text{def}}{=} \mathcal{F}_{r-1} \cap \mathcal{A}_{r-1} \cap \mathcal{B}_{r-1,\ell}. \tag{4.14}$$

We discuss below how $\mathcal{F}_{r-1}$ and $\mathcal{A}_{r-1}$ are computed.

If more than two levels are available ($r > 1$), the same technique can be applied recursively, the process stopping at level 0, where there is no coarser model, and thus Taylor's model is always used. Let us consider the details of this process in this more general situation. Consider iteration k at level i, and assume that $x_{i,k}$ is an iterate in the minimization of f_i inside an iteration q at level $i + 1$ where f_i has been chosen as a model for f_{i+1} (i.e. a recursive iteration).

We start by considering the representation of the problem's bounds at lower levels. At level i, we define

$$\mathcal{F}_i \stackrel{\text{def}}{=} \{x \mid l_i \leq x \leq u_i\} \tag{4.15}$$

the "restricted" feasible domain, where

$$[l_i]_j \stackrel{\text{def}}{=} [x_{i,0}]_j + \frac{1}{\|P_{i+1}\|_\infty} \max_{t=1,\ldots,n_{i+1}} \left\{ \begin{array}{lll} [l_{i+1} - x_{i+1,q}]_t & \text{when} & [P_{i+1}]_{tj} > 0 \\ [x_{i+1,q} - u_{i+1}]_t & \text{when} & [P_{i+1}]_{tj} < 0 \end{array} \right\} \tag{4.16}$$

and

$$[u_i]_j \stackrel{\text{def}}{=} [x_{i,0}]_j + \frac{1}{\|P_{i+1}\|_\infty} \min_{t=1,\ldots,n_{i+1}} \left\{ \begin{array}{lll} [u_{i+1} - x_{i+1,q}]_t & \text{when} & [P_{i+1}]_{tj} > 0 \\ [x_{i+1,q} - l_{i+1}]_t & \text{when} & [P_{i+1}]_{tj} < 0 \end{array} \right\} \tag{4.17}$$

for $j = 1, \ldots, n_i$. The idea behind this generalization of the definition by Gelman and Mandel [18], originally stated for more specific prolongation operators[(2)], is to use the structure of P_{i+1} to compute a coarse set of bounds $\mathcal{F}_i$ *in order to guarantee that its prolongation is feasible for the fine level*, that is

$$l_{i+1} \leq x_{i+1} + P_{i+1}(l_i - x_i) \leq x_{i+1} + P_{i+1}(u_i - x_i) \leq u_{i+1}$$

for all $x_{i+1} \in \mathcal{F}_{i+1}$, for all $x_i \in \mathcal{F}_i$. This property is proved in Lemma 4.2.3 below. Figure 4.2 on the facing page shows the application of the (generalized) Gelman-Mandel's coarse bounds and their prolongation on the example of Figure 4.1.

We now turn to the representation of the upper trust region at the lower level. At level i we also define

$$\mathcal{A}_i = \{x \mid v_i \leq x \leq w_i\}, \tag{4.18}$$

the restriction of the trust-region constraints inherited from levels r to $i+1$ through $x_{i+1,q}$, computed using the restriction operator R_{i+1}. The j-th components of v_i and w_i are

$$\begin{aligned} [v_i]_j \quad = \quad & \sum_{u=1,[R_{i+1}]_{ju}>0}^{n_{i+1}} [R_{i+1}]_{ju} [\max(v_{i+1}, x_{i+1,q} - \Delta_{i+1,q} e)]_u \\ & + \sum_{u=1,[R_{i+1}]_{ju}<0}^{n_{i+1}} [R_{i+1}]_{ju} [\min(w_{i+1}, x_{i+1,q} + \Delta_{i+1,q} e)]_u \end{aligned} \tag{4.19}$$

and

$$\begin{aligned} [w_i]_j \quad = \quad & \sum_{u=1,[R_{i+1}]_{ju}>0}^{n_{i+1}} [R_{i+1}]_{ju} [\min(w_{i+1}, x_{i+1,q} + \Delta_{i+1,q} e)]_u \\ & + \sum_{u=1,[R_{i+1}]_{ju}<0}^{n_{i+1}} [R_{i+1}]_{ju} [\max(v_{i+1}, x_{i+1,q} - \Delta_{i+1,q} e)]_u \end{aligned} \tag{4.20}$$

[(2)]The original formulation is restricted to the case where $\|P_{i+1}\|_\infty \leq 1$ and $P_{i+1} > 0$, and is given by

$$[l_i]_j \stackrel{\text{def}}{=} [x_{i,0}]_j + \max_{t=1,\ldots,n_{i+1}:[P_{i+1}]_{tj}>0} [l_{i+1} - x_{i+1,q}]_t,$$

$$[u_i]_j \stackrel{\text{def}}{=} [x_{i,0}]_j + \max_{t=1,\ldots,n_{i+1}:[P_{i+1}]_{tj}>0} [x_{i+1,q} - u_{i+1}]_t.$$

We extend this definition to cover prolongation operators with $\|P_{i+1}\|_\infty > 1$ and also to handle negative elements in P_{i+1} (as in cubic interpolation, for instance), which imposes taking both upper and lower bounds at the upper level into account for the definition of the upper and lower bounds at the coarse level.

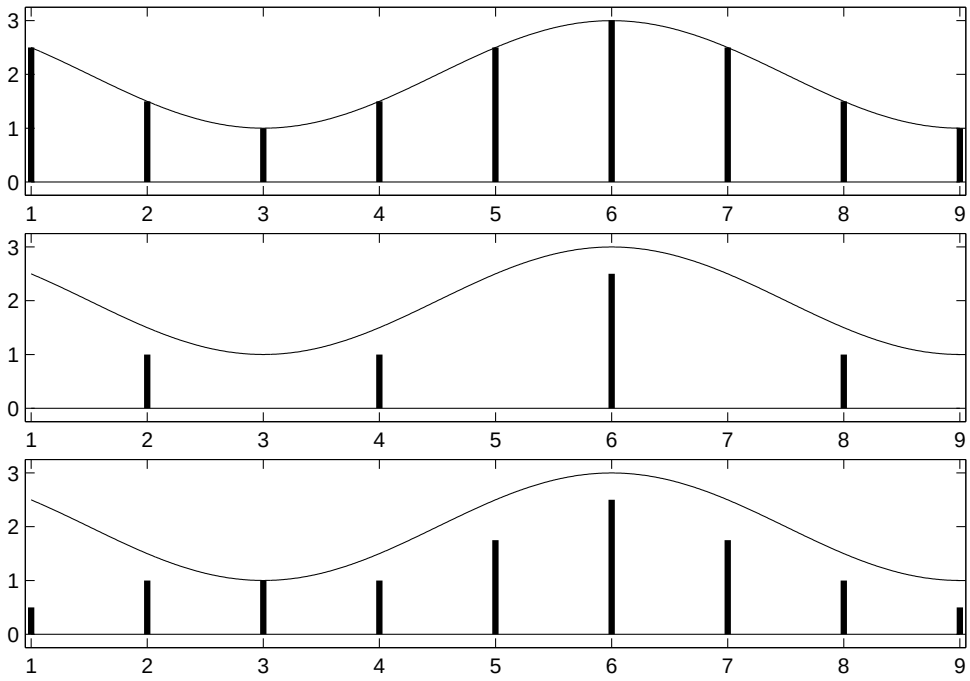


Figure 4.2: Prolongation of Gelman and Mandel's bounds for the same example as in Figure 4.1. As in this figure, the vertical bars in the upper graph show the possible ranges for the values $\phi(1), \ldots, \phi(9)$. The vertical bars in the middle graph now show the ranges obtained by deriving the generalized Gelman and Mandel's bounds from the set of bounds obtained in the upper graph, and the vertical bars in the lower graphs finally correspond to applying the prolongation (linear interpolation) to the bounds obtained in the middle graph.

(we define $v_r = -\infty$ and $w_r = +\infty$ for consistency).

Notice that, as allowed in our above discussion, the choice of using R_i to restrict these bounds implies that iterates at level i are not necessarily included in the level i trust region anymore but cannot be very far from it. Indeed, recalling that $\|R_i\|_\infty = 1$ for $i = 1, ..., r$, we have that

$$\|x_{i,k+1} - x_{i,k}\|_\infty \leq \|P_i\|_\infty \|x_{i-1,*} - x_{i-1,0}\|_\infty. \tag{4.21}$$

If the trust region at level i around iterate $x_{i,k}$ is defined by

$$\mathcal{B}_{i,k} = \{x_{i,k} + s \in \mathbb{R}^{n_i} \mid \|s\| \leq \Delta_{i,k}\},$$

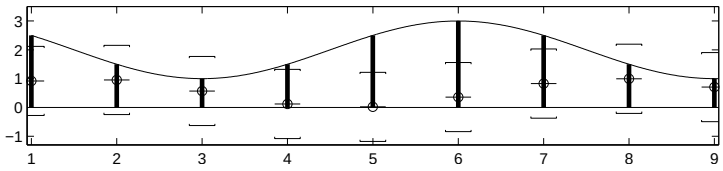
we then have to find a step $s_{i,k}$ which sufficiently reduces a model of f_i in the region

$$\mathcal{W}_{i,k} = \mathcal{F}_i \cap \mathcal{A}_i \cap \mathcal{B}_{i,k}. \tag{4.22}$$

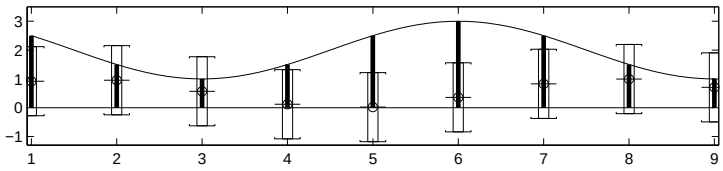
Observe that the set $\mathcal{W}_{i,k}$ can either be viewed both as $\mathcal{W}_{i,k} = \mathcal{L}_i \cap \mathcal{B}_{i,k}$, the intersection of a level dependent domain $\mathcal{L}_i = \mathcal{F}_i \cap \mathcal{A}_i$ with an iteration dependent trust-region $\mathcal{B}_{i,k}$, or as $\mathcal{W}_{i,k} = \mathcal{F}_i \cap \mathcal{S}_{i,k}$, the intersection of $\mathcal{F}_i$, the feasible set for hard constraints, with $\mathcal{S}_{i,k} \stackrel{\text{def}}{=} \mathcal{A}_i \cap \mathcal{B}_{i,k}$, the feasible set for soft ones. This last set can be interpreted as a "composite" trust region which includes all constraints imposed by trust regions at level i and higher. Note that all the involved sets are boxes, which makes their representation and intersection computationally easy.

Figure 4.3 on the next page illustrates the process to compute a recursive step in the example already used in Figures 4.1 and 4.2. In this figure, the values of the variables at successive

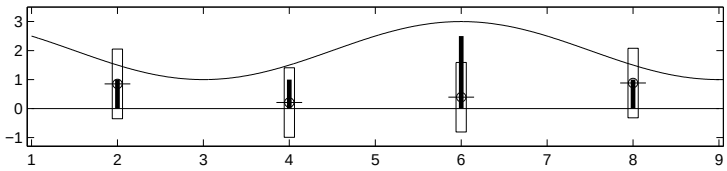
The iterate $x_{r,k}$, and the sets $\mathcal{F}_r$ (thick lines), $\mathcal{A}_r = \mathbb{R}^9$ and $\mathcal{B}_{r,k}$ (brackets) are given at level r:



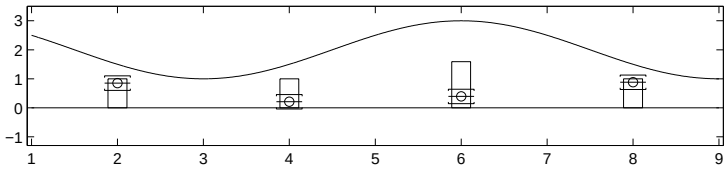
1) compute $S_{r,k} = \mathcal{A}_r \cap \mathcal{B}_{r,k}$ (thin boxes) at level r:



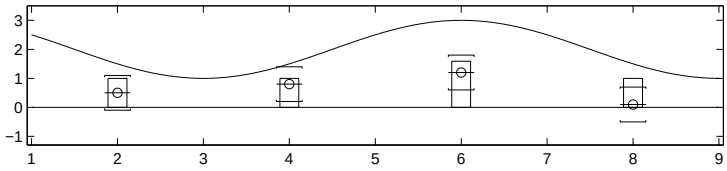
2) restrict the problem: compute $x_{r-1,0} = Rx_{r,k}$, $\mathcal{F}_{r-1}$ (thick lines) and $\mathcal{A}_{r-1}$ (thin boxes) at level $r-1$:



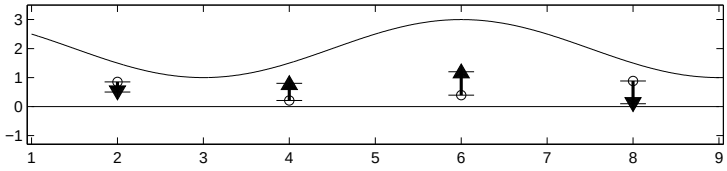
3) compute $\mathcal{L}_{r-1} = \mathcal{F}_{r-1} \cap \mathcal{A}_{r-1}$ (fat boxes) and add $\mathcal{B}_{r-1,0}$ (brackets) at level $r-1$:



4) perform some iterations at level $r-1$, yielding $x_{r-1,*}$ (circle) and $\mathcal{B}_{r-1,*}$(new brackets):



5) compute $x_{r-1,*} - x_{r-1,0}$ (arrows), the total step at level $r-1$:



6) prolongate the step (arrows) and compute the level-r trial point $x_{r,k} + P(x_{r-1,*} - x_{r-1,0})$.

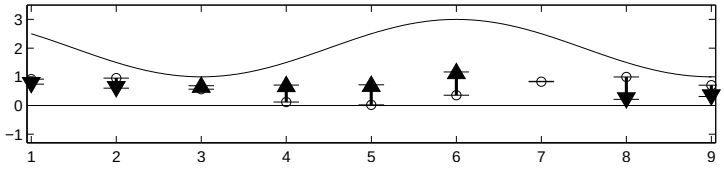


Figure 4.3: The definition of the various sets and the step computation for the example of Fig. 4.1.

iterates are shown by horizontally barred circles and the steps by arrows. Trust-region bounds on each variable are shown with vertical brackets, the sets $\mathcal{S}_{r,k}$ and $\mathcal{A}_{r-1}$ by thin vertical boxes, the set $\mathcal{L}_{r-1}$ by fatter vertical boxes and the sets $\mathcal{F}_r$ and $\mathcal{F}_{r-1}$ by thick lines. At stage 3, $\mathcal{W}_{r-1,0}$ is given by the intersection of the fat boxes representing $\mathcal{L}_r$ with the brackets representing $\mathcal{B}_{r-1,0}$.

Once $\mathcal{W}_{i,k}$ is known, we then choose a model for f_{i+1} as one of

$$m_{i+1,q}(x_{i+1,q} + s_{i+1}) = f_{i+1}(x_{i+1,q}) + \langle g_{i+1,q}, s_{i+1}\rangle + \frac{1}{2}\langle s_{i+1}, H_{i+1,q} s_{i+1}\rangle, \tag{4.23}$$

the usual truncated Taylor series for f_{i+1} (with $g_{i+1,q} = \nabla f_{i+1}(x_{i+1,q})$ and $H_{i+1,q}$ being a symmetric approximation of $\nabla^2 f_{i+1}(x_{i+1,q})$), or f_i. As discussed below, this freedom of choice is crucial for the application of multigrid-type techniques in our context. In the latter case, we assume that f_{i+1} and its coarse model f_i are *first order coherent*, that is $g_{i,0} = R_{i+1} g_{i+1,q}$. This assumption is not restrictive, as we can always choose a first order coherent coarse model of f_{i+1} by adding a gradient correction term to f_i as in

$$f_i(x_{i,0} + s_i) + \langle R_{i+1} g_{i+1,q} - \nabla f_i(x_{i,0}), s_i\rangle.$$

If one chooses the model f_i (which is only possible if $i > 0$), the determination of the step then consists in (approximately) solving the lower-level bound-constrained problem

$$\min_{x_{i,0}+\tilde{s}_i \in \mathcal{L}_i} f_i(x_{i,0} + \tilde{s}_i). \tag{4.24}$$

This minimization produces a step s_i such that $f_i(x_{i,0} + s_i) < f_i(x_{i,0})$ which must be then brought back to level $i+1$ by the prolongation P_{i+1}, i.e. $s_{i+1} = P_{i+1} s_i$. Note that

$$\langle g_{i+1,q}, s_{i+1}\rangle = \langle g_{i+1,q}, P_{i+1} s_i\rangle = \frac{1}{\sigma_{i+1}}\langle R_{i+1} g_{i+1,q}, s_i\rangle. \tag{4.25}$$

As the decrease of f_i achieved by s_i can be approximated to first-order by $f_i(x_{i,0}) - f_i(x_{i,0} + s_i) \approx \langle g_{i,0}, s_i\rangle = \langle R_{i+1} g_{i+1,q}, s_i\rangle$, the decrease of the model at level $i+1$ when computing steps at level i is computed, using (4.25), as $[f_i(x_{i,0}) - f_i(x_{i,0} + s_i)]/\sigma_{i+1}$.

But does it always make sense to use the lower level model? The answer obviously depends on the benefit expected from the solution of (4.24). In RMTR, as described in the previous section, it sufficed to test if $\|g_{i,0}\|_2 = \|R_{i+1} g_{i+1,q}\|_2$ was large enough compared to $\|g_{i+1,q}\|_2$. However, this criticality measure is inadequate in our context because (4.24) is now a bound-constrained problem. Thus, we choose to follow Conn et al. [12] and use, for each $x_{i+1,q} \in \mathcal{L}_{i+1}$, the criticality measure[3] defined by

$$\chi_{i+1,q} \stackrel{\text{def}}{=} \chi(x_{i+1,q}) = |\min_{\substack{x_{i+1,q}+d \in \mathcal{L}_{i+1} \\ \|d\| \le 1}} \langle g_{i+1,q}, d\rangle| \stackrel{\text{def}}{=} |\langle g_{i+1,q}, d_{i+1,q}\rangle|. \tag{4.26}$$

Then if the restriction of the problem from the non-critical iterate $x_{i+1,q}$ at level $i+1$ to level i is not already first-order critical, that is if

$$\chi_{i,0} \ge \kappa_\chi \chi_{i+1,q}, \tag{4.27}$$

[3] Other criticality measures are possible, such as $\mu_{i+1,q} = \|\mathrm{Proj}_{i+1,q}(x_{i+1,q} - g_{i+1,q}) - x_{i+1,q}\|_2$ where $\mathrm{Proj}_{i+1,q}$ is the orthogonal projection onto the box $\mathcal{L}_{i+1}$ but we will not investigate this alternative here.

for some constant $\kappa_\chi \in (0, 1)$, we may proceed at this lower level. Otherwise, the recursion is useless and we should use (4.23) instead.

Once we have decided to approximately solve (4.24), we must also decide what we mean by "approximately". We choose to terminate the minimization at level r if $\chi_{r,k} \leq \epsilon_r$ for some $\epsilon_r > 0$ and, in the spirit of (4.27), to terminate the lower level minimization at iterate (i, p) as soon as the inequality

$$\chi_{i,p} < \epsilon_i \stackrel{\text{def}}{=} \kappa_\chi \epsilon_{i+1}, \tag{4.28}$$

holds. We then define $x_{i,*} = x_{i,p}$, $s_i = x_{i,*} - x_{i,0}$ and $s_{i+1,q} = P_{i+1} s_i$.

If, on the other hand, we decide at iteration $(i + 1, q)$ to use Taylor's model $m_{i+1,q}$ given by (4.23), a step $s_{i+1,q}$ is then computed that produces a sufficient decrease in the value of this model in its usual meaning for trust-region methods with convex constraints (defined here by the set $\mathcal{L}_{i+1}$), that is, $s_{i+1,q}$ is such that it satisfies

$$m_{i+1,q}(x_{i+1,q}) - m_{i+1,q}(x_{i+1,q} + s_{i+1,q}) \geq \kappa_{\text{red}} \chi_{i+1,q} \min\left[1, \frac{\chi_{i+1,q}}{\beta_{i+1,q}}, \Delta_{i+1,q}\right], \tag{4.29}$$

for some constant $\kappa_{\text{red}} \in (0, \frac{1}{2})$ and $\beta_{i+1,q} \stackrel{\text{def}}{=} 1 + \|H_{i+1,q}\|_{\infty,1}$, where $\|A\|_{\infty,1} \stackrel{\text{def}}{=} \max_{x \neq 0} \left\{\frac{\|Ax\|_1}{\|x\|_\infty}\right\}$, for all matrices A. Despite its apparently technical character, this requirement, known as the modified Cauchy condition, is not overly restrictive and can be guaranteed in practical algorithms, as described for instance in Section 12.2.1 of Conn et al. [12].

We now specify our algorithm formally, as Algorithm RMTR$_\infty$ on the facing page. It uses the constants $0 < \eta_1 \leq \eta_2 < 1$ and $0 < \gamma_1 \leq \gamma_2 < 1$ and Δ_i^s $(i = 0, \ldots, r)$.

Algorithm 4.2.1: RMTR$_\infty$ $(i, x_{i,0}, g_{i,0}, \chi_{i,0}, \mathcal{F}_i, \mathcal{A}_i, \epsilon_i)$

Step 0: Initialization. Compute $f_i(x_{i,0})$. Set $k = 0$ and

$$\mathcal{L}_i = \mathcal{F}_i \cap \mathcal{A}_i \quad \text{and} \quad \mathcal{W}_{i,0} = \mathcal{L}_i \cap \mathcal{B}_{i,0},$$

where $\mathcal{B}_{i,0} = \{x_{i,0} + s \in \mathbb{R}^{n_i} \,|\, \|s\|_\infty \leq \Delta_{i,0} = \Delta_i^s\}$.

Step 1: Model choice. If $i = 0$, go to Step 3. Else, compute $\mathcal{L}_{i-1}$ and $\chi_{i-1,0}$. If (4.27) fails, go to Step 3. Otherwise, choose to go to Step 2 or to Step 3.

Step 2: Recursive step computation. Call Algorithm

$$\textbf{RMTR}_\infty(i-1, R_i x_{i,k}, R_i g_{i,k}, \chi_{i-1,0}, \mathcal{F}_{i-1}, \mathcal{A}_{i-1}, \kappa_\chi \epsilon_i),$$

yielding an approximate solution $x_{i-1,*}$ of (4.24). Then define $s_{i,k} = P_i(x_{i-1,*} - R_i x_{i,k})$, set $\delta_{i,k} = \frac{1}{\sigma_i}[f_{i-1}(R_i x_{i,k}) - f_{i-1}(x_{i-1,*})]$ and go to Step 4.

Step 3: Taylor step computation. Choose $H_{i,k}$ and compute a step $s_{i,k} \in \mathbb{R}^{n_i}$ that sufficiently reduces the model $m_{i,k}$ given by (4.23) in the sense of (4.29) and such that $x_{i,k} + s_{i,k} \in \mathcal{W}_{i,k}$. Set $\delta_{i,k} = m_{i,k}(x_{i,k}) - m_{i,k}(x_{i,k} + s_{i,k})$.

Step 4: Acceptance of the trial point. Compute $f_i(x_{i,k} + s_{i,k})$ and

$$\rho_{i,k} = [f_i(x_{i,k}) - f_i(x_{i,k} + s_{i,k})]/\delta_{i,k}. \tag{4.30}$$

If $\rho_{i,k} \geq \eta_1$, then define $x_{i,k+1} = x_{i,k} + s_{i,k}$; otherwise, define $x_{i,k+1} = x_{i,k}$.

Step 5: Termination. Compute $g_{i,k+1}$ and $\chi_{i,k+1}$. If $\chi_{i,k+1} \leq \epsilon_i$ or $x_{i,k+1} \notin \mathcal{A}_i$, then return with the approximate solution $x_{i,*} = x_{i,k+1}$.

Step 6: Trust-Region Update. Set

$$\Delta_{i,k+1} \in \begin{cases} [\Delta_{i,k}, +\infty) & \text{if } \rho_{i,k} \geq \eta_2, \\ [\gamma_2 \Delta_{i,k}, \Delta_{i,k}] & \text{if } \rho_{i,k} \in [\eta_1, \eta_2), \\ [\gamma_1 \Delta_{i,k}, \gamma_2 \Delta_{i,k}] & \text{if } \rho_{i,k} < \eta_1, \end{cases} \tag{4.31}$$

and $\mathcal{W}_{i,k+1} = \mathcal{L}_i \cap \mathcal{B}_{i,k+1}$ where

$$\mathcal{B}_{i,k+1} = \{x_{i,k+1} + s \in \mathbb{R}^{n_i} \,|\, \|s\|_\infty \leq \Delta_{i,k+1}\}.$$

Increment k by one and go to Step 1.

Some comments are now necessary for a full understanding of this algorithm.

1. The test for the value of i at the beginning of Step 1 is designed to identify the lowest level, at which no further recursion is possible. In this case, a Taylor's iteration is the only choice left.

2. As a result of the discussion preceding (4.21), $x_{i,k+1}$ may not belong to the composite trust region $\mathcal{A}_i$ when the step $s_{i,k}$ is computed by a recursive iteration. However, as indicated above, we wish to limit the length of the step at level $i+1$ to a multiple of the trust-region size. Because of (4.21) and the definition of $\mathcal{A}_i$, we may achieve this objective by stopping our iteration at level i as soon as the iterates leave the composite trust-region $\mathcal{A}_i$. This explains the second termination test in Step 5 of the algorithm and is discussed in detail in Lemma 4.2.4.

3. The difference between the "restriction formulae" (4.15)-(4.17) for the hard bounds and (4.18)-(4.20) for the soft ones makes it necessary to pass both $\mathcal{A}_i$ and $\mathcal{F}_i$ to the algorithm at level i, as it is necessary to compute $\mathcal{L}_i$ at each level independently.

4. The original problem (4.13) is solved by calling RMTR$_\infty$ from a virtual $(r+1)$-rst level at which we assume the trust region to be infinite.

As usual in trust-region algorithms, iterations at which $\rho_{i,k} \geq \eta_1$ are called *successful*. At such iterations, the trial point $x_{i,k} + s_{i,k}$ is accepted as the new iterate and the radius of the corresponding trust region is possibly enlarged. If the iteration is unsuccessful, the trial point is rejected and the radius is reduced.

4.2.2 Convergence theory

Having motivated our interest in the new method, both as an efficient solver for bound-constrained problems and as an improvement on the existing RMTR algorithm for the unconstrained case, we are now interested in obtaining a theoretical guarantee that RMTR$_\infty$ converges to a first-order critical point of the problem from any starting point. The theory proposed in this section differs significantly from the proof for the RMTR algorithm in Gratton et al. (2008), mostly because of the form of the new criticality measure (imposed by the bounds and the choice of the infinity norm) and because the new algorithm allows for potentially very asymmetric trust regions.

We start by making our assumptions more formal. First, we assume that the Hessians of each f_i and their approximations are bounded above by the constant $\kappa_{\mathrm{H}} \geq 1$, so that, for $i = 0, \ldots, r$,

$$1 + \|\nabla^2 f_i(x_i)\|_{\infty,1} \leq \kappa_{\mathrm{H}} \tag{4.32}$$

for all $x_i \in \mathcal{F}_i$ and

$$\beta_{i,k} \leq \kappa_{\mathrm{H}} \tag{4.33}$$

for all k, where $\beta_{i,k}$ is as in (4.29). We also assume that all gradients at all levels remain uniformly bounded, which is to say that there exists $\kappa_{\mathrm{g}} \geq 1$ such that

$$\|\nabla f_i(x_i)\|_1 \leq \kappa_{\mathrm{g}} \quad \text{for all} \quad i = 0, \ldots, r, \quad \text{and all} \quad x_i \in \mathcal{F}_i. \tag{4.34}$$

In addition, we assume that the criticality measure $\chi(\cdot)$ satisfies, for all iterations (i, ℓ) inside a recursive iteration (i, k), that

$$\chi_{i-1,0} = \chi(x_{i-1,0}) \leq 2\kappa_g n_{i-1} \Delta_{i,k}. \tag{4.35}$$

These assumptions are not overly restrictive and, for instance, (4.34) automatically holds by continuity if all iterates $x_{j,\ell}$ remain in a bounded domain, which is the case if both l and u are finite in (4.13). We next prove a useful level-independent property of the criticality measure $\chi(\cdot)$ in our context.

Lemma 4.2.1 *Consider the optimization problem (4.13) and define the function $\chi(x)$ by*

$$\chi(x) = |\min_{\substack{x+d\in\mathcal{F} \\ \|d\|\leq 1}} \langle \nabla f(x), d\rangle| \tag{4.36}$$

(as in (4.26)). Then, for all $x, y \in \mathcal{F}$, we have that

$$|\chi(x) - \chi(y)| \leq \kappa_L \|x - y\|_\infty.$$

with $\kappa_L = 2(\kappa_H + \kappa_g)$.

Proof. Let x and y be in $\mathcal{F}$. The optimization problem (4.36) may be written as

$$\max_{\max(-1,l_i-x_i)\leq d_i\leq \min(1,u_i-x_i)} \langle -\nabla f(x), d\rangle. \tag{4.37}$$

Now denote by $m(x)$ the vector of average of the bounds on the variables in (4.37), whose i-th component is given by

$$m_i(x) = \tfrac{1}{2}[\max(-1, l_i - x_i) + \min(1, u_i - x_i)], \tag{4.38}$$

and by $r(x)$ the vector of "radii" whose i-th component is

$$r_i(x) = \tfrac{1}{2}[\min(1, u_i - x_i) - \max(-1, l_i - x_i)]. \tag{4.39}$$

Then, for $i = 1, \ldots n$,

$$2|r_i(x)| \leq |\min(1, u_i - x_i)| + |\max(-1, l_i - x_i)| \leq 2$$

and similarly, $2|m_i(x)| \leq 2$, which shows that both functions $|r_i(x)|$ and $|m_i(x)|$ are bounded by 1 for x in $\mathcal{F}$.

We now show that the functions $x \mapsto \min(1, u_i - x_i)$ and $x \mapsto \max(-1, l_i - x_i)$ are both unit Lipschitz continuous, that is Lipschitz continuous with constant 1. Consider x and y in $\mathcal{F}$, and define

$$\delta = |\min(1, u_i - x_i) - \min(1, u_i - y_i)|.$$

For $1 \leq u_i - x_i$ and $1 \leq u_i - y_i$, we have $\delta = 0$. If $1 \leq u_i - x_i$, and $1 \geq u_i - y_i$, we see that $1 - u_i + y_i \geq 0$, and that $x_i \leq y_i$. Therefore we have that

$$\delta = |1 - u_i + y_i| = 1 - u_i + y_i \leq u_i - x_i - u_i + y_i = y_i - x_i = |x_i - y_i|,$$

and we also deduce by symmetry that $\delta = |x_i - y_i|$ whenever $1 \geq u_i - x_i$, and $1 \leq u_i - y_i$. Finally, if $1 \geq u_i - x_i$, and $1 \geq u_i - y_i$, we obtain that

$$\delta = |u_i - x_i - u_i + y_i| = |x_i - y_i|.$$

Hence the function $x \mapsto \min(1, u_i - x_i)$ is unit Lipschitz continuous. The result for $x \mapsto \max(-1, l_i - x_i)$ is obtained from the same arguments. Combining these results with (4.38) and (4.39), we obtain that both $r_i(x)$ and $m_i(x)$ are also unit Lipschitz continuous.

Now defining $\tilde{d}$ such that $d = m(x) + r(x) \circ \tilde{d}$ where $\circ$ is the (Hadamard) component-wise product, i.e. $x \circ y = [x_1 y_1, \ldots, x_n y_n]^T$, we observe that the minimization problem (4.37) may also be written as

$$\max_{\|\tilde{d}\|_\infty \leq 1} \langle -\nabla f(x), m(x) + r(x) \circ \tilde{d} \rangle,$$

whose solution is then analytically given by

$$\chi(x) = \langle -\nabla f(x), m(x) \rangle + \|\nabla f(x) \circ r(x)\|_1.$$

Using this formula, we now show that $\chi(x)$ is Lipschitz continuous in $\mathcal{F}$. From the mean-value theorem, we know that

$$\nabla f(x) = \nabla f(y) + G_{[x,y]}(x - y), \tag{4.40}$$

where, from (4.32),

$$\|G_{[x,y]}\|_{\infty,1} = \| \int_0^1 \nabla^2 f(x + t(y - x))\, dt \|_{\infty,1} \leq \max_{z \in [x,y]} \|\nabla^2 f(z)\|_{\infty,1} \leq \kappa_{\mathrm{H}}. \tag{4.41}$$

Hence, using $|\langle u, v \rangle| \leq \|u\|_1 \|v\|_\infty$, the inequality $\|m(x)\|_\infty \leq 1$, (4.34) and the unit Lipschitz continuity of $m(x)$, we obtain that

$$\begin{aligned} |\langle \nabla f(x), m(x) \rangle - \langle \nabla f(y), m(y) \rangle| & \\ &\leq |\langle \nabla f(x) - \nabla f(y), m(x) \rangle + \langle \nabla f(y), m(x) - m(y) \rangle| \\ &\leq (\kappa_{\mathrm{H}} + \kappa_{\mathrm{g}}) \|x - y\|_\infty. \end{aligned}$$

In addition,

$$\begin{aligned} \|\nabla f(x) \circ r(x)\|_1 - \|\nabla f(y) \circ r(y)\|_1 & \\ &\leq \|\nabla f(x) \circ r(x) - \nabla f(y) \circ r(y)\|_1 \\ &\leq \|\nabla f(x) \circ (r(x) - r(y))\|_1 + \|(\nabla f(x) - \nabla f(y)) \circ r(y)\|_1. \end{aligned}$$

Using now the inequality $\|u \circ v\|_1 \leq \|u\|_1 \|v\|_\infty$, we obtain from $\|r(y)\|_\infty \leq 1$, (4.40) and (4.41) that

$$\|(\nabla f(x) - \nabla f(y)) \circ r(y)\|_1 \leq \|\nabla f(x) - \nabla f(y)\|_1 \|r(y)\|_\infty \leq \kappa_{\mathrm{H}} \|x - y\|_\infty$$

and, similarly, from the unit Lipschitz continuity of $r(x)$, and (4.34), that

$$\|\nabla f(x) \circ (r(x) - r(y))\|_1 \leq \|\nabla f(x)\|_1 \|r(x) - r(y)\|_\infty \leq \kappa_{\mathrm{g}} \|x - y\|_\infty.$$

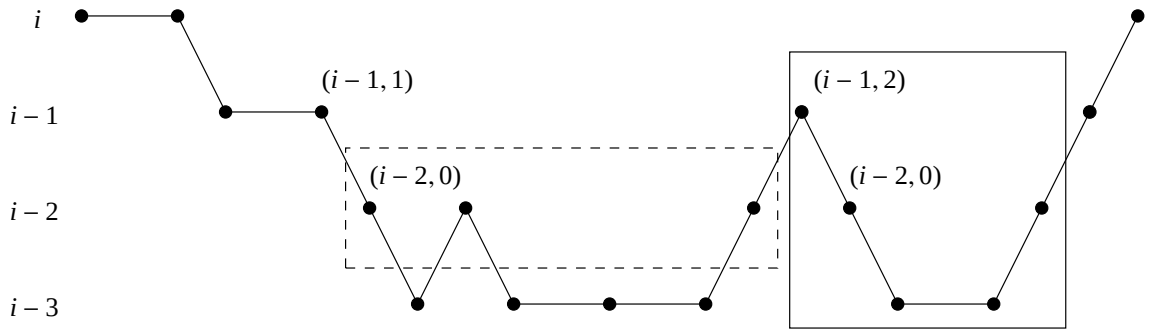


Figure 4.4: Illustration of some multilevel notations. The dashed rectangle area contains a minimization sequence at level $i-2$ initiated at iteration $(i-1,1)$ and the solid line rectangle contains $\mathcal{R}(i-1,2)$.

Putting together the above results yields that $|\chi(x) - \chi(y)| \leq 2(\kappa_{\rm H} + \kappa_{\rm g})\|x-y\|_\infty$. □

We now define some additional notation and concepts. We first choose the constant $\kappa_{\rm P} \geq 1$ such that

$$\|P_i\|_\infty \leq \kappa_{\rm P} \qquad \text{for all} \quad i = 1, \ldots, r. \tag{4.42}$$

If we choose to go to Step 2 (i.e. we choose to use the model f_{i-1} at iteration (i,k), we say that this iteration initiates a *minimization sequence* at level $i-1$, which consists of all successive iterations *at this level* (starting from the point $x_{i-1,0} = R_i x_{i,k}$) until a return is made to level i within iteration (i,k). In this case, we say that iteration (i,k) is the *predecessor* of the minimization sequence at level $i-1$. If $(i-1,\ell)$ belongs to this minimization sequence, this is written as $(i,k) = \pi(i-1,\ell)$. We also denote by p_{i-1} the index of the penultimate iterate in the minimization sequence $\{x_{i-1,0}, \ldots, x_{i-1,p_{i-1}}, x_{i-1,*}\}$. Note that (4.22) implies that $\mathcal{W}_{i,k} \subseteq \mathcal{B}_{i,k}$. To each iteration (i,k) at level i, we now associate the set

$$\mathcal{R}(i,k) \stackrel{\rm def}{=} \{(j,\ell) \mid \text{iteration } (j,\ell) \text{ occurs within iteration } (i,k)\}.$$

This set always contains the pair (i,k) and contains only that pair if a Taylor step is used at iteration (i,k). If we choose a recursive step, then it also contains the pairs of level and iteration number of all iterations that occur in the potential recursion started in Step 2 and terminating on return within iteration (i,k), but it does not contain the pairs of indices corresponding to the terminating iterates $(j,*)$ of its internal minimization sequences. It is easy to verify that $j \leq i$ for every j such that $(j,\ell) \in \mathcal{R}(i,k)$ for some non-negative k and ℓ. Note also that $\mathcal{R}(i,k)$ contains at most one minimization sequence at level $i-1$, but may contain more than one at level $i-2$ and below, since each iteration at level $i-1$ may generate its own. Associated with $\mathcal{R}(i,k)$, we also define

$$\mathcal{T}(i,k) \stackrel{\rm def}{=} \{(j,\ell) \in \mathcal{R}(i,k) \mid (j,\ell) \text{ is a Taylor iteration}\}.$$

The algorithm also ensures the following technical lemma.

Lemma 4.2.2 *There exists an* $\epsilon_{\min} \in (0,1]$ *such that, for each iteration* $(i,k) \neq (i,*)$ *(i.e., for all iterates at level* i *but the last one),*

$$\chi_{i,k} \geq \epsilon_{\min}. \tag{4.43}$$

Proof. The inequality (4.28), which is the stopping criteria for minimization at level j, in Step 5 of the algorithm, implies that for all (i,k) and all $(j,\ell) \in \mathcal{R}(i,k)$,

$$\chi_{j,\ell} \geq \epsilon_j = \kappa_\chi \chi_{\pi(j,\ell)} \geq \kappa_\chi \epsilon_{j+1} = \kappa_\chi^2 \chi_{\pi^2(j,\ell)} \geq \cdots \geq \kappa_\chi^{j-j} \chi_{i,k} \geq \cdots \geq \kappa_\chi^r \epsilon_r.$$

This proves (4.43) with $\epsilon_{\min} = \min[1, \kappa_\chi^r \epsilon_r]$. □

We now prove the general version of the Gelman and Mandel's result stating that "bound constraints are preserved" by the prolongation operator.

Lemma 4.2.3 *The definitions (4.16)–(4.17) enforce the inclusion*

$$x_{i,k} + P_i(x_{i-1} - x_{i-1,0}) \in \mathcal{F}_i \qquad \text{for all} \quad x_{i-1} \in \mathcal{F}_{i-1} \tag{4.44}$$

for $i = 1, \ldots, r$. As a consequence $x_{i,k} \in \mathcal{F}_i$ for all $i = 0, \ldots, r$ and all $k \geq 0$.

Proof. For $t = 1, \ldots, n_i$, define $\phi_{i,t} = \sum_{j=1}^{n_{i-1}} |[P_i]_{t,j}|$ and observe that $\phi_{i,t} \leq \|P_i\|_\infty$ for all t. Consider now any $x_{i-1} \in \mathcal{F}_{i-1}$ and the corresponding lower level step $s_{i-1} = x_{i-1} - x_{i-1,0}$. Then (4.16) and (4.17) imply that

$$\begin{aligned}
&[x_{i,k}]_t + \sum_{j=1}^{n_{i-1}} [P_i]_{tj} [s_{i-1}]_j \\
&\quad = [x_{i,k}]_t + \sum_{j=1, [P_i]_{tj}<0}^{n_{i-1}} |[P_i]_{tj}| (-[s_{i-1}]_j) + \sum_{j=1, [P_i]_{tj}>0}^{n_{i-1}} |[P_i]_{tj}| [s_{i-1}]_j \\
&\quad \geq [x_{i,k}]_t + \sum_{j=1, [P_i]_{tj}<0}^{n_{i-1}} |[P_i]_{tj}| \frac{(-\min_t [x_{i,k} - l_i]_t)}{\|P_i\|_\infty} + \sum_{j=1, [P_i]_{tj}>0}^{n_{i-1}} |[P_i]_{tj}| \frac{\max_t [l_i - x_{i,k}]_t}{\|P_i\|_\infty} \\
&\quad \geq [x_{i,k}]_t + \sum_{j=1, [P_i]_{tj}<0}^{n_{i-1}} |[P_i]_{tj}| \frac{[l_i - x_{i,k}]_t}{\|P_i\|_\infty} + \sum_{j=1, [P_i]_{tj}>0}^{n_{i-1}} |[P_i]_{tj}| \frac{[l_i - x_{i,k}]_t}{\|P_i\|_\infty} \\
&\quad \geq [x_{i,k}]_t + \phi_{i,t} \frac{[l_i - x_{i,k}]_t}{\|P_i\|_\infty} \\
&\quad = \frac{\phi_{i,t}}{\|P_i\|_\infty} [l_i]_t + \left(1 - \frac{\phi_{i,t}}{\|P_i\|_\infty}\right) [x_{i,k}]_t \\
&\quad \geq [l_i]_t
\end{aligned}$$

where the last inequality results from the fact that $[x_{i,k}]_t \geq [l_i]_t$. A similar reasoning gives that

$$[x_{i,k}]_t + \sum_{j=1}^{n_{i-1}} [P_i]_{t,j} [s_{i-1}]_j \leq [u_i]_t$$

for all t, thereby concluding the proof of (4.44). The feasibility of every iterate with respect to the level-dependent bound constraints then results from the fact that all trial points at level i belong to $\mathcal{F}_i$ by construction. □

We next show that the distance from all iterates in a single minimization sequence at level i to the starting point of that sequence is bounded above by a multiple of the trust-region radius at the predecessor's level.

Lemma 4.2.4 *The definitions (4.19)-(4.20) imply that, for* $0 \leq j < r$,

$$\|x - x_{j,0}\|_\infty \leq 2\Delta_{\pi(j,0)} \tag{4.45}$$

for all $x \in \mathcal{L}_j$.

Proof. Consider an $x \in \mathcal{L}_j \subseteq \mathcal{A}_j$. If we now denote the bounds defining the set $\mathcal{S}_{\pi(j,0)}$ by

$$\overline{v}_{j+1} \stackrel{\text{def}}{=} \max\left[v_{j+1}, x_{\pi(j,0)} - \Delta_{\pi(j,0)}e\right] \quad \text{and} \quad \overline{w}_{j+1} \stackrel{\text{def}}{=} \min\left[w_{j+1}, x_{\pi(j,0)} + \Delta_{\pi(j,0)}e\right],$$

we then verify that

$$\begin{aligned}
[w_j - v_j]_t &= \sum_{u=1,[R_{j+1}]_{tu}>0}^{n_{j+1}} [R_{j+1}]_{tu}[\overline{w}_{j+1}]_u + \sum_{u=1,[R_{j+1}]_{tu}<0}^{n_{j+1}} [R_{j+1}]_{tu}[\overline{v}_{j+1}]_u \\
&\qquad - \sum_{u=1,[R_{j+1}]_{tu}>0}^{n_{j+1}} [R_{j+1}]_{tu}[\overline{v}_{j+1}]_u - \sum_{u=1,[R_{j+1}]_{tu}<0}^{n_{j+1}} [R_{j+1}]_{tu}\overline{w}_{j+1}]_u \\
&= \sum_{u=1,[R_{j+1}]_{tu}>0}^{n_{j+1}} [R_{j+1}]_{tu}[\overline{w}_{j+1} - \overline{v}_{j+1}]_u + \sum_{u=1,[R_{j+1}]_{tu}<0}^{n_{j+1}} [R_{j+1}]_{tu}[\overline{v}_{j+1} - \overline{w}_{j+1}]_u \\
&\stackrel{\text{def}}{=} [R_{j+1}z(t)]_t,
\end{aligned}$$

where we have used (4.19) and (4.20), and where, for $t = 1, \ldots, n_{j+1}$,

$$[z(t)]_u = \text{sign}([R_{j+1}]_{tu})[\overline{w}_{j+1} - \overline{v}_{j+1}]_u.$$

This last definition implies that $\|z(t)\|_\infty = \|\overline{w}_{j+1} - \overline{v}_{j+1}\|_\infty$ for $t = 1, \ldots, n_{j+1}$. Taking norms and using the identity $\|R_{j+1}\|_\infty = 1$, we therefore obtain that

$$\begin{aligned}
\|w_j - v_j\|_\infty &= \max_t |[R_{j+1}z(t)]_t| \\
&\leq \max_t \|R_{j+1}z(t)\|_\infty \\
&\leq \max_t \|z(t)\|_\infty \\
&= \|\overline{w}_{j+1} - \overline{v}_{j+1}\|_\infty.
\end{aligned} \tag{4.46}$$

Remembering now the definition of $\overline{w}_{j+1}$ and $\overline{v}_{j+1}$, we see that

$$\begin{aligned}
\|\overline{w}_{j+1} - \overline{v}_{j+1}\|_\infty &= \|\min\left[w_{j+1}, x_{\pi(j,0)} + \Delta_{\pi(j,0)}e\right] - \max\left[v_{j+1}, x_{\pi(j,0)} - \Delta_{\pi(j,0)}e\right]\|_\infty \\
&\leq \|\min\left[w_{j+1}, x_{\pi(j,0)} + \Delta_{\pi(j,0)}e\right] - x_{\pi(j,0)}\|_\infty \\
&\quad + \|x_{\pi(j,0)} - \max\left[v_{j+1}, x_{\pi(j,0)} - \Delta_{\pi(j,0)}e\right]\|_\infty \\
&\leq 2\Delta_{\pi(j,0)}.
\end{aligned}$$

Combining now this bound with (4.46) and our assumption that $x \in \mathcal{A}_j$, we obtain that

$$\|x - x_{j,0}\|_\infty \leq \|w_j - v_j\|_\infty \leq 2\Delta_{\pi(j,0)}.$$

□

Our next proposition indicates that, if $\Delta_{i,k}$ becomes too small, then the method reduces, at level i, to the standard trust-region method using Taylor's iterations only.

Lemma 4.2.5 *Assume that, for some iteration* (i,k)*,*

$$\Delta_{i,k} \leq \frac{1}{2}\min\left[1, \frac{\epsilon_{\min}}{2\kappa_g}, \Delta^s_{\min}\right] \stackrel{\text{def}}{=} \kappa_2 \in (0,1), \tag{4.47}$$

where $\Delta^s_{\min} \stackrel{\text{def}}{=} \min_{i=0,\ldots,r} \Delta^s_i$ *Then no recursion occurs in iteration* (i,k) *and* $\mathcal{R}(i,k) = \mathcal{T}(i,k) = \{(i,k)\}$.

Proof. Assume that iteration (i,k) is recursive and that iteration $(i-1,0)$ exists. Since (4.47) implies that $2\Delta_{i,k} \leq 1$, we deduce from (4.45) (with $x = x_{i-1,0} + d \in \mathcal{L}_{i-1}$) that $\mathcal{L}_{i-1} \subseteq \{x_{i-1,0} + d \mid \|d\|_\infty \leq 1\}$ and thus that

$$\chi_{i-1,0} = |\min_{x_{i-1,0}+d\in\mathcal{L}_{i-1}} \langle g_{i-1,0}, d\rangle| = |\langle g_{i-1,0}, d_{i-1,0}\rangle| \tag{4.48}$$

with

$$\|d_{i-1,0}\|_\infty \leq 2\Delta_{i,k}. \tag{4.49}$$

Using (4.47), (4.43), (4.48), the inequality $|\langle u,v\rangle| \leq \|u\|_1\|v\|_\infty$, (4.34) and (4.49) successively, we conclude that

$$\chi_{i-1,0} = |\langle g_{i-1,0}, d_{i-1,0}\rangle| \leq \|g_{i-1,0}\|_1\|d_{i-1,0}\|_\infty \leq 2\kappa_g\Delta_{i,k}$$

and thus that

$$\Delta_{i,k} \leq \frac{\epsilon_{\min}}{4\kappa_g} \leq \frac{\chi_{i-1,0}}{4\kappa_g} = \frac{2\kappa_g\Delta_{i,k}}{4\kappa_g} \leq \tfrac{1}{2}\Delta_{i,k}$$

which is impossible. Hence our initial assumption that iteration (i,k) is recursive cannot hold and the proof is complete. □

This lemma essentially states that when the trust-region becomes too small compared to the current criticality level, then too little can be gained from lower level iterations to allow recursion. This has the following important consequence.

Lemma 4.2.6 *Consider an iteration* (i,k) *for which* $\chi_{i,k} > 0$ *and*

$$\Delta_{i,k} \leq \min\left[\kappa_2, \kappa_3\chi_{i,k}\right], \tag{4.50}$$

where κ_2 *is defined in (4.47) and* $\kappa_3 \in (0,1)$ *is given by*

$$\kappa_3 = \min\left[1, \frac{\kappa_{red}(1-\eta_2)}{\kappa_H}\right].$$

Then iteration (i,k) *is very successful and* $\Delta_{i,k+1} \geq \Delta_{i,k}$.

Proof. Because of (4.47) and Lemma 4.2.5, we know that iteration (i,k) is a Taylor iteration. Thus, using (4.29), and the definition of $\delta_{i,k}$ in Step 3 of the algorithm,

$$\delta_{i,k} \geq \kappa_{\text{red}}\chi_{i,k}\min\left[1, \frac{\chi_{i,k}}{\beta_{i,k}}, \Delta_{i,k}\right].$$

But, because $\kappa_{\rm red} \in (0, \frac{1}{2})$ and thus $\kappa_{\rm red}(1-\eta_2) \leq 1$, and also because of (4.33) and the definition of $\beta_{i,k}$, (4.50) implies that $\Delta_{i,k} \leq \min\left[1, \frac{\chi_{i,k}}{\beta_{i,k}}\right]$ and hence that

$$\delta_{i,k} \geq \kappa_{\rm red}\chi_{i,k}\Delta_{i,k}. \tag{4.51}$$

We now observe that the mean-value theorem, (4.23) and the definition of $g_{i,k}$ ensure that

$$f_i(x_{i,k}+s_{i,k}) - m_{i,k}(x_{i,k}+s_{i,k}) = \tfrac{1}{2}\langle s_{i,k}, [\nabla^2 f_i(\xi_{i,k}) - H_{i,k}]s_{i,k}\rangle$$

for some $\xi_{i,k} \in [x_{i,k}, x_{i,k}+s_{i,k}]$, and thus using (4.32), (4.33), the inequality $|\langle u, v\rangle| \leq \|u\|_1\|v\|_\infty$ and the bound $\|s_{i,k}\|_\infty \leq \Delta_{i,k}$, we obtain that

$$|f_i(x_{i,k}+s_{i,k}) - m_{i,k}(x_{i,k}+s_{i,k})| \leq \frac{1}{2}[\,\|\nabla^2 f_i(\xi_{i,k})\|_{\infty,1} + \|H_{i,k}\|_{\infty,1}\,]\|s_{i,k}\|_\infty^2 \leq \kappa_{\rm H}\Delta_{i,k}^2.$$

Combining now (4.30), the definition of $s_{i,k}$, (4.50), (4.51) and this last inequality, we verify that

$$|\rho_{i,k} - 1| \leq \left|\frac{f_i(x_{i,k}+s_{i,k}) - m_{i,k}(x_{i,k}+s_{i,k})}{\delta_{i,k}}\right| \leq \frac{\kappa_{\rm H}}{\kappa_{\rm red}\chi_{i,k}}\Delta_{i,k} \leq 1-\eta_2.$$

Thus iteration (i,k) must be very successful and, because of (4.31), the trust-region radius cannot decrease. □

This last result implies the following useful consequence.

Lemma 4.2.7 *Each minimization sequence contains at least one successful iteration.*

Proof. This follows from the fact that unsuccessful iterations cause the trust-region radius to decrease, until (4.50) is eventually satisfied and a (very) successful iteration occurs because of Lemma 4.2.6. □

The attentive reader will have noticed that the term in $\Delta^{\rm s}_{\rm min}$ in the minimum defining κ_2 in (4.47) has not been used in Lemma 4.2.5. This term is however crucial in the following further consequence of (4.47).

Lemma 4.2.8 *For every iteration (j,ℓ), with $j = 0, \ldots, r$ and $\ell > 0$, we have that*

$$\Delta_{j,\ell} \geq \Delta_{min} \stackrel{\rm def}{=} \gamma_1 \min[\kappa_2, \kappa_3\epsilon_j]. \tag{4.52}$$

Proof. Suppose that (j,ℓ) is the first iteration such that

$$\Delta_{j,\ell} < \gamma_1 \min[\kappa_2, \kappa_3\epsilon_j]. \tag{4.53}$$

Since $\gamma_1 < 1$ and $\kappa_2 \leq \Delta^{\rm s}_{\rm min}$, we then obtain that

$$\Delta_{j,0} = \Delta^{\rm s}_j \geq \Delta^{\rm s}_{\rm min} \geq \gamma_1\Delta^{\rm s}_{\rm min} \geq \gamma_1 \min[\kappa_2, \kappa_3\epsilon_j],$$

and, because of (4.53), we have that $\ell > 0$. This in turn implies that $\Delta_{j,\ell}$ is computed using Step 6 of the algorithm. But, the mechanism of the algorithm imposes that $\Delta_{j,\ell} \geq \gamma_1 \Delta_{j,\ell-1}$ an thus (4.53) also yields that

$$\Delta_{j,\ell-1} < \min[\kappa_2, \kappa_3 \epsilon_j] \leq \min[\kappa_2, \kappa_3 \chi_{j,\ell-1}],$$

where we have used the mechanism of the algorithm to derive the last inequality. Hence, we may apply Lemma 4.2.6 to conclude that iteration $(j, \ell - 1)$ is very successful and that $\Delta_{j,\ell} \geq \Delta_{j,\ell-1}$. Thus, iteration (j, ℓ) cannot be the first such that (4.53) holds. This implies that (4.53) is impossible, which completes the proof. □

We next show the crucial result that the algorithm is well defined, and that all the recursions are finite.

Theorem 4.2.9 *The number of iterations in each level is finite. Moreover, there exists $\kappa_h \in (0, 1)$ such that, for every minimization sequence at level $i = 0, \ldots, r$ and every $t \geq 0$,*

$$f_i(x_{i,0}) - f_i(x_{i,t+1}) \geq \tau_{i,t} \mu^{i+1} \kappa_h,$$

where $\tau_{i,t}$ is the total number of successful Taylor iterations in $\bigcup_{\ell=0}^{t} \mathcal{R}(i, \ell)$ and $\mu = \eta_1 / \sigma_{\max}$ with $\sigma_{\max} = \max[1, \max_{i=1,\ldots,r} \sigma_i]$.

Proof. We will show this by induction on the levels, starting from level 0. First, let us define $\omega_{i,t}$ as the number of successful Taylor iterations in $\mathcal{R}(i, t)$. Thus,

$$\tau_{i,t} = \sum_{\ell=0}^{t} \omega_{i,\ell}.$$

Note that, if iteration (i, ℓ) is successful, then $\omega_{i,\ell} \geq 1$.

Consider first a minimization sequence started at level 0, and assume without loss of generality, that it belongs to $\mathcal{R}(r, k)$ for some $k \geq 0$. Every iteration in this minimization sequence has to be a Taylor iteration, which implies that the sufficient decrease condition (4.29) is satisfied, and in particular, for all successful iterations,

$$\begin{aligned} f_0(x_{0,\ell}) - f_0(x_{0,\ell+1}) &\geq \eta_1 \delta_{0,\ell} \geq \eta_1 \kappa_{\text{red}} \chi_{0,\ell} \min\left[1, \frac{\chi_{0,\ell}}{\beta_{0,\ell}}, \Delta_{0,\ell}\right] \\ &\geq \omega_{0,\ell} \eta_1 \kappa_{\text{red}} \epsilon_{\min} \min\left[1, \frac{\epsilon_{\min}}{\kappa_{\text{H}}}, \Delta_{\min}\right] \end{aligned}$$

where we used Lemma 4.2.8, (4.33), (4.43) and the fact that $\omega_{0,\ell} = 1$ for every successful iteration $(0, \ell)$, since $\mathcal{R}(0, \ell) = \{(0, \ell)\}$. Since we know from Lemma 4.2.7 that every minimization sequence has at least one successful iteration, we can sum up the reductions obtained at level 0, which gives us

$$f_0(x_{0,0}) - f_0(x_{0,t+1}) = \sum_{\ell=0}^{t} {}^{(\text{S})} \left[f_0(x_{0,\ell}) - f_0(x_{0,\ell+1}) \right] \geq \tau_{0,t} \eta_1 \kappa_{\text{h}} \geq \tau_{0,t} \mu \kappa_{\text{h}} \tag{4.54}$$

where the superscript (S) indicates that the sum is restricted to successful iterations and where

$$\kappa_{\mathrm{h}} \stackrel{\mathrm{def}}{=} \kappa_{\mathrm{red}}\epsilon_{\min}\min\left[1, \frac{\epsilon_{\min}}{\kappa_{\mathrm{H}}}, \Delta_{\min}\right] = \kappa_{\mathrm{red}}\epsilon_{\min}\min\left[\frac{\epsilon_{\min}}{\kappa_{\mathrm{H}}}, \Delta_{\min}\right], \tag{4.55}$$

where the last equality results from the inequalities $\epsilon_{\min} \leq 1$ and $\kappa_{\mathrm{H}} \geq 1$. If $r = 0$, since $f_0 = f$ is bounded below by assumption, then (4.54) implies that $\tau_{0,t}$ is finite. If $r > 0$, f_0 is continuous, and thus it is bounded below on the set $\{x \in \mathbb{R}^{n_0} \mid \|x - x_{0,0}\|_\infty \leq 2\Delta_{r,k}\}$, and again, $\tau_{0,t}$ has to be finite. Since $\tau_{0,t}$ accounts for all successful iterations in the minimization sequence, we obtain that there must be a last finite successful iteration $(0, p_0)$. For the purpose of obtaining a contradiction, let us assume that the sequence is infinite. Then, all iterations $(0, \ell)$ would be unsuccessful for $\ell > p_0$, causing $\Delta_{0,\ell}$ to converge to zero, which is impossible in view of Lemma 4.2.8. Hence, the minimization sequence is finite. The same reasoning may be applied to every such sequence at level 0.

Now, consider an arbitrary minimization sequence at level i within $\mathcal{R}(r, k)$ for some $k > 0$, and assume that each minimization sequence at level $i-1$ is finite and also that each successful iteration $(i-1, u)$ in every minimization sequence at this lower level satisfies

$$f_{i-1}(x_{i-1,u}) - f_{i-1}(x_{i-1,u+1}) \geq \omega_{i-1,u}\mu^i\kappa_{\mathrm{h}}. \tag{4.56}$$

Consider a successful iteration (i, ℓ), whose existence is ensured by Lemma 4.2.7. If it is a Taylor iteration, we obtain that

$$f_i(x_{i,\ell}) - f_i(x_{i,\ell+1}) \geq \eta_1\kappa_{\mathrm{h}} \geq \mu^{i+1}\kappa_{\mathrm{h}} = \omega_{i,\ell}\mu^{i+1}\kappa_{\mathrm{h}}, \tag{4.57}$$

since $\eta_1 \in (0, 1)$, $\sigma_{\max} > 1$ and $\omega_{i,\ell} = 1$ for every successful Taylor iteration (i, ℓ). If, on the other hand, iteration (i, ℓ) uses Step 2, then we obtain that

$$\begin{aligned} f_i(x_{i,\ell}) - f_i(x_{i,\ell+1}) &\geq \frac{\eta_1}{\sigma_i}\left[f_{i-1}(x_{i-1,0}) - f_{i-1}(x_{i-1,*})\right] \\ &\geq \mu \sum_{u=0}^{p_{i-1}{}^{(S)}} \left[f_{i-1}(x_{i-1,u}) - f_{i-1}(x_{i-1,u+1})\right]. \end{aligned}$$

Since $\omega_{i,\ell} = \tau_{i-1,p_{i-1}}$, the definition of $\tau_{i-1,t}$ and (4.56) give that

$$f_i(x_{i,\ell}) - f_i(x_{i,\ell+1}) \geq \mu^{i+1}\kappa_{\mathrm{h}} \sum_{u=0}^{p_{i-1}} \omega_{i-1,u} = \tau_{i-1,p_{i-1}}\mu^{i+1}\kappa_{\mathrm{h}} = \omega_{i,\ell}\mu^{i+1}\kappa_{\mathrm{h}}. \tag{4.58}$$

Combining (4.57) and (4.58), we see that (4.56) again holds at level i instead of $i-1$. Moreover, as above,

$$f_i(x_{i,0}) - f_i(x_{i,t+1}) = \sum_{\ell=0}^{t^{(S)}} \left[f_i(x_{i,\ell}) - f_i(x_{i,\ell+1})\right] \geq \tau_{i,t}\mu^{i+1}\kappa_{\mathrm{h}}, \tag{4.59}$$

for the minimization sequence including iteration (i, ℓ). If $i = r$, $f_i = f$ is bounded below by assumption and (4.59) imposes that the number of successful iterations in this sequence must again be finite. The same conclusion holds if $i < r$, since f_i is continuous and hence bounded below on the set $\{x \in \mathbb{R}^{n_i} \mid \|x - x_{i,0}\|_\infty \leq 2\Delta_{r,k}\}$ which contains $x_{i,t+1}$ because of Lemma 4.2.4.

As for level 0, we may then conclude that the number of iterations (both successful and unsuccessful) in the minimization sequence is finite. Moreover, the same reasoning holds for every minimization sequence at level i, and the induction is complete. □

Corollary 4.2.10 *Assume that one knows a constant $f_{\rm low}$ such that $f_r(x_r) = f(x) \geq f_{\rm low}$ for every $x \in \mathbb{R}^n$. Then Algorithm RMTR$_\infty$ needs at most*

$$\left\lceil \frac{f(x_{r,0}) - f_{\rm low}}{\theta(\epsilon_{\rm min})} \right\rceil \tag{4.60}$$

successful Taylor iterations at any level to obtain an iterate $x_{r,k}$ such that $\chi_{r,k} < \epsilon_r$, where

$$\theta(\epsilon) = \mu^{r+1}\kappa_{red}\epsilon \min\left[\frac{\epsilon}{\kappa_H}, \gamma_1 \min\left[\kappa_2, \kappa_3\epsilon\right]\right]. \tag{4.61}$$

Proof. The desired bound directly follows from Theorem 4.2.9, (4.55), (4.52) and the definition of $\epsilon_{\rm min}$. □

This complexity result for general nonconvex problems is similar to Corollary 3.8 in Gratton et al. (2008), and may also be very pessimistic. It is of the same order as the corresponding bound for the pure gradient method (see [49], page 29). This is not surprising given that it is based on the Cauchy condition, which itself results from a step in the steepest-descent direction. Note that the bound is in terms of iteration numbers, and only implicitly accounts for the cost of computing a Taylor step satisfying (4.29). As was the case for the Euclidean norm, this suggests several comments.

1. The bound (4.60) is expressed in terms of the number of successful Taylor iterations, that is successful iterations where the trial step is computed without resorting to further recursion. This provides an adequate measure of the linear algebra effort for all successful iterations, since successful iterations using the recursion of Step 2 cost little beyond the evaluation of the level-dependent objective function and its gradient. Moreover, the number of such iterations is, by construction, at most equal to r times that of Taylor iterations (in the worst case where each iteration at level r includes a full recursion to level 0 with a single successful iteration at each level $j > 0$). Hence the result shows that the number of necessary successful iterations, all levels included, is of order $1/\epsilon^2$ for small values of ϵ. This order is not qualitatively altered by the inclusion of unsuccessful iterations either, provided we replace the very successful trust-region radius update (top case in (4.31)) by

$$\Delta_{i,k}^{+} \in [\Delta_{i,k}, \gamma_3\Delta_{i,k}] \quad \text{if} \quad \rho_{i,k} \geq \eta_2,$$

for some $\gamma_3 > 1$. Indeed, Lemma 4.2.8 imposes that the decrease in radius caused by unsuccessful iterations must asymptotically be compensated by an increase at successful ones. This is to say that, if α is the average number of unsuccessful iterations per successful one at any level, then one must have that $\gamma_3\gamma_2^{\alpha} \geq 1$, and therefore that $\alpha \leq -\log(\gamma_3)/\log(\gamma_2)$. Thus the complexity bound in $1/\epsilon^2$ for small ϵ is only modified

by a constant factor if all iterations (successful and unsucessful) are considered. This therefore also gives a worst case upper bound on the number of function and gradient evaluations.

2. Moreover, (4.60) involves the number of successful Taylor iterations *summed up on all levels* (as a result of Theorem 4.2.9). Thus such successful iterations at cheap low levels decrease the number of necessary expensive ones at higher levels, and the multilevel algorithm requires (at least in the theoretical worst case) fewer Taylor iterations at the upper level than the single-level variant. This provides theoretical backing for the practical observation that the structure of multilevel bound-constrained optimization problems can be used to advantage.

3. The definition of $\theta(\epsilon)$ in (4.61) is interesting in that it does not depend on the problem dimension, but rather on the properties of the problem or of the algorithm itself. Thus, if we consider the case where different levels correspond to different discretization meshes and make the mild assumption that r and κ_{H} are uniformly bounded above, we deduce that our complexity bound is mesh-independent.

A second important consequence of Theorem 4.2.9 is that the algorithm is globally convergent, in the sense that, if ϵ_r is "driven to zero", it generates a subsequence of iterates that are asymptotically first-order critical. More specifically, we examine the sequence of iterates $\{x_{r,k}\}$ generated as follows. We consider, at level r, a sequence of tolerances $\{\epsilon_{r,j}\} \in (0,1)$ monotonically converging to zero, start the algorithm with $\epsilon_r = \epsilon_{r,0}$ and alter slightly the mechanism of Step 5 (at level r only) to reduce ϵ_r from $\epsilon_{r,j}$ to $\epsilon_{r,j+1}$ as soon as $\chi_{r,k+1} \leq \epsilon_{r,j}$. The calculation is then continued with this more stringent threshold until it is also attained, ϵ_r^g is then again reduced and so on.

Theorem 4.2.11 *Assume that ϵ_r is "driven to zero" in Algorithm RMTR$_\infty$. Then*

$$\liminf_{k\to\infty} \chi_{r,k} = 0. \tag{4.62}$$

Proof. Since $\Delta_{r+1,0} = \infty$ ensures that $\mathcal{L}_r = \mathcal{F}_r$, Lemma 4.2.3 implies that each successive minimization at level r can only stop at iteration k if

$$\chi_{r,k+1} \leq \epsilon_{r,j}. \tag{4.63}$$

Theorem 4.2.9 then implies that there are only finitely many successful iterations between two reductions of ϵ_r. We therefore obtain that for each $\epsilon_{r,j}$ there is an arbitrarily large k such that (4.63) holds. The desired result then follows immediately from our assumption that $\{\epsilon_{r,j}\}$ converges to zero. □

Of course, the interest of this result is mostly theoretical, since most practical applications of Algorithm RMTR$_\infty$ consider a nonzero gradient tolerance ϵ_r.

Observe that our definition of ϵ_i in (4.28) implies that, if ϵ_r is driven to zero, then so is $\epsilon_i = \kappa_\chi^{r-i}\epsilon_r$. As for the Euclidean case, and assuming the trust region becomes asymptotically

inactive at every level (as is most often the case in practice), each minimization sequence in the algorithm becomes infinite (as if it were initiated with a zero gradient threshold and an infinite initial radius). Recursion to lower levels then remains possible for arbitrarily small gradients, and may therefore occur arbitrarily far in the sequence of iterates. Moreover, we may still apply Theorem 4.2.11 at each level and deduce that, if the trust region becomes asymptotically inactive,

$$\liminf_{k\to\infty} \chi_{i,k} = 0 \tag{4.64}$$

for all $i = 0, \ldots, r$.

As is the case for single-level trust-region algorithms, we now would like to prove that the limit inferior in (4.62) and (4.64) can be replaced by a true limit. This requires the notion of a *recursively successful iteration*. We say that iteration $(j, \ell) \in \mathcal{R}(i, k)$ is *recursively successful for* (i, k) whenever iterations $(j, \ell), \pi(j, 0), \pi^2(j, 0), \ldots, \pi^{i-j}(j, 0) = (i, k)$ are all successful. This is to say that the decrease in the objective function obtained at iteration (j, ℓ) effectively contributes to the reduction obtained at iteration (i, k). We start by stating a result on the relative sizes of the objective function decreases in the course of a recursive iteration.

Lemma 4.2.12 *Assume that some iteration* $(j, \ell) \in \mathcal{R}(i, k)$ *is recursively successful for* (i, k). *Then*

$$f_j(x_{j,\ell}) - f_j(x_{j,\ell+1}) \le f_j(x_{j,0}) - f_j(x_{j,*}) \le \mu^{j-i} [\, f_i(x_{i,k}) - f_i(x_{i,k+1}) \,].$$

Proof. The first inequality immediately results from the monotonicity of the sequence of objective function values in a minimization sequence. To prove the second inequality, consider iteration $(j + 1, q) = \pi(j, 0)$. Then

$$f_j(x_{j,0}) - f_j(x_{j,*}) = \sigma_{j+1}\delta_{j+1,q} \le \eta_1^{-1}\sigma_{\max} [\, f_{j+1}(x_{j+1,q}) - f_{j+1}(x_{j+1,q+1}) \,]$$

where we used the definition of $\delta_{j+1,q}$, the definition of $\sigma_{\max}$ and the fact that iteration $(j + 1, q)$ must be successful since (j, ℓ) is recursively successful for (i, k). But this argument may now be repeated at level $j + 2, \ldots, i$, yielding the desired bound, given that $\mu = \eta_1/\sigma_{\max} < 1$. □

This lemma then allows us to express a simple relation between the size of Taylor steps at recursively successful iterations and the associated objective decrease.

Lemma 4.2.13 *Assume that the Taylor iteration* $(j, \ell) \in \mathcal{R}(i, k)$ *is recursively successful for* (i, k) *and that, for some* $\epsilon \in (0, 1)$,

$$\chi_{j,\ell} \ge \epsilon \tag{4.65}$$

and

$$f_i(x_{i,k}) - f_i(x_{i,k+1}) < \frac{\mu^r \eta_1 \kappa_{red} \epsilon^2}{\kappa_H}. \tag{4.66}$$

Then

$$\|x_{j,\ell} - x_{j,\ell+1}\|_\infty \le \frac{1}{\kappa_{red}\eta_1\epsilon} [\, f_j(x_{j,\ell}) - f_j(x_{j,\ell+1}) \,]. \tag{4.67}$$

Proof. We know from (4.29), (4.33), (4.65) and the successful nature of iteration (j,ℓ) that

$$\begin{aligned} f_j(x_{j,\ell}) - f_j(x_{j,\ell+1}) &\geq \eta_1\kappa_{\rm red}\chi_{j,\ell}\min\left[\frac{\chi_{j,\ell}}{\kappa_{\rm H}},\Delta_{j,\ell},1\right] \\ &\geq \eta_1\kappa_{\rm red}\epsilon\min\left[\frac{\epsilon}{\kappa_{\rm H}},\Delta_{j,\ell},1\right] \\ &= \eta_1\kappa_{\rm red}\epsilon\min\left[\frac{\epsilon}{\kappa_{\rm H}},\Delta_{j,\ell}\right] \end{aligned} \tag{4.68}$$

where we used (4.33) and the inequality $\epsilon < 1$ to deduce the last equality. But Lemma 4.2.12 gives that

$$\begin{aligned} f_j(x_{j,\ell}) - f_j(x_{j,\ell+1}) &\leq \mu^{j-i}\left[\, f_i(x_{i,k}) - f_i(x_{i,k+1})\,\right] \\ &\leq \mu^{-r}\left[\, f_i(x_{i,k}) - f_i(x_{i,k+1})\,\right] \\ &\leq \frac{\eta_1\kappa_{\rm red}\epsilon^2}{\kappa_{\rm H}}, \end{aligned}$$

where we used (4.66) to deduce the last inequality. Hence we see that only the second term in the last minimum of (4.68) can be active, which gives that

$$f_j(x_{j,\ell}) - f_j(x_{j,\ell+1}) \geq \eta_1\kappa_{\rm red}\epsilon\Delta_{j,\ell}.$$

We then obtain (4.67) from the observation that $x_{j,\ell+1} = x_{j,\ell} + s_{j,\ell} \in \mathcal{W}_{j,\ell} \subseteq \mathcal{B}_{j,\ell}$. □

We next prove the following useful technical lemma.

Lemma 4.2.14 *Assume that a minimization sequence at level j $(0 \leq j \leq r)$ is such that*

$$\chi_{j,0} \geq \epsilon_{ncr} \tag{4.69}$$

for some $\epsilon_{ncr} \in (0,1)$, but also that

$$\|s_{j,\ell}\|_\infty \leq \kappa_{ncr}\left[\, f_j(x_{j,\ell}) - f_j(x_{j,\ell+1})\,\right] \tag{4.70}$$

for some $\kappa_{ncr} > 0$ as long as iteration (j,ℓ) is successful and $\chi_{j,\ell} \geq \frac{1}{2}\epsilon_{ncr}$. Assume finally that

$$f_j(x_{j,0}) - f_j(x_{j,*}) \leq \frac{\epsilon_{ncr}}{2\kappa_{ncr}\kappa_L}. \tag{4.71}$$

Then $\chi_{j,\ell} \geq \frac{1}{2}\epsilon_{ncr}$ and (4.70) holds for all $\ell \geq 0$.

Proof. Assume that there exists a (first) successful iteration (j,s) such that

$$\chi_{j,s} < \tfrac{1}{2}\epsilon_{\rm ncr}, \tag{4.72}$$

which implies that $\chi_{j,\ell} \geq \frac{1}{2}\epsilon_{\rm ncr}$ for all $0 \leq \ell < s$. We now use (4.70) and the triangle inequality, and sum on all successful iterations (at level j) from 0 to $s-1$, yielding

$$\|x_{j,0} - x_{j,s}\|_\infty \leq \sum_{\ell=0}^{s-1}{}^{(S)}\|x_{j,\ell} - x_{j,\ell+1}\|_\infty \leq \kappa_{\rm ncr}\left[\, f_j(x_{j,0}) - f_j(x_{j,s})\,\right]. \tag{4.73}$$

Applying now Lemma 4.2.1, the monotonicity of f_j within the minimization sequence, the bound $n_j \leq n_r$ and (4.71), we obtain from (4.73) that

$$\begin{aligned} |\chi_{j,0} - \chi_{j,s}| &\leq \kappa_{\rm ncr}\kappa_{\rm L}\,[\, f_j(x_{j,0}) - f_j(x_{j,s})\,] \\ &\leq \kappa_{\rm ncr}\kappa_{\rm L}\,[\, f_j(x_{j,0}) - f_j(x_{j,*})\,] \\ &\leq \tfrac{1}{2}\epsilon_{\rm ncr}. \end{aligned}$$

But this last inequality is impossible since we know from (4.69) and (4.72) that $\chi_{j,0} - \chi_{j,s} > \frac{1}{2}\epsilon_{\rm ncr}$. Hence our assumption (4.72) is itself impossible and we obtain that, for all $\ell \geq 0$, $\chi_{j,\ell} \geq \frac{1}{2}\epsilon_{\rm ncr}$. This and the lemma's assumptions then ensure that (4.70) also holds for all $j \geq 0$. □

We now consider the case of recursive iterations.

Lemma 4.2.15 *Assume that, for some recursive successful iteration* (i,k)*,*

$$\chi_{i,k} \geq \epsilon_{rsi} \tag{4.74}$$

and

$$f_i(x_{i,k}) - f_i(x_{i,k+1}) \leq \frac{\kappa_\chi \epsilon_{rsi}}{2\kappa_{rsi}\kappa_L} \tag{4.75}$$

for some $\epsilon_{rsi} \in (0,1)$ *and some* $\kappa_{rsi} > 0$*. Assume also that*

$$\|s_{i-1,\ell}\|_\infty \leq \kappa_{rsi}\,[\, f_{i-1}(x_{i-1,\ell}) - f_{i-1}(x_{i-1,\ell+1})\,] \tag{4.76}$$

for all (recursively) successful iterations in the minimization sequence initiated at level $i-1$ *by iteration* (i,k) *as long as*

$$\chi_{i-1,\ell} \geq \tfrac{1}{2}\kappa_\chi \epsilon_{rsi}.$$

Then

$$\|s_{i,k}\|_\infty \leq \mu^{-1}\kappa_{\rm P}\kappa_{rsi}\,[\, f_i(x_{i,k}) - f_i(x_{i,k+1})\,].$$

Proof. Consider the minimization sequence initiated at level $i-1$ by iteration (i,k). Because of (4.27) and (4.74), we have that $\chi_{i-1,0} \geq \kappa_\chi \epsilon_{\rm rsi}$. We may now apply Lemma 4.2.14 with $\epsilon_{\rm ncr} = \kappa_\chi\epsilon_{\rm rsi}$ and $\kappa_{\rm ncr} = \kappa_{\rm rsi}$, given that (4.75) ensures (4.71). As a result, we know that $\chi_{i-1,\ell} \geq \frac{1}{2}\kappa_\chi\epsilon_{\rm rsi}$ and (4.76) hold for all successful iterations $(i-1,\ell)$ $(\ell \geq 0)$. Using the triangle inequality and summing on all successful iterations at level $i-1$, we find that

$$\|x_{i-1,0} - x_{i-1,*}\|_\infty \leq \sum_{\ell=0}^{p_{i-1}} {}^{(S)}\|x_{i-1,\ell} - x_{i-1,\ell+1}\|_\infty \leq \kappa_{\rm rsi}\,[\, f_{i-1}(x_{i-1,0}) - f_{i-1}(x_{i-1,*})\,].$$

This inequality, the definition of $s_{i,k}$, (4.42) and Lemma 4.2.12 in turn imply that

$$\begin{aligned} \|s_{i,k}\|_\infty &\leq \|P_i\|_\infty \|x_{i-1,0} - x_{i-1,*}\|_\infty \\ &\leq \kappa_{\rm P}\kappa_{\rm rsi}\,[\, f_{i-1}(x_{i-1,0}) - f_{i-1}(x_{i-1,*})\,] \\ &\leq \mu^{-1}\kappa_{\rm P}\kappa_{\rm rsi}\,[\, f_i(x_{i,k}) - f_i(x_{i,k+1})\,]. \end{aligned}$$

□

Our next step is to consider the cumulative effect of all the complete recursion for an iteration at the finest level.

Lemma 4.2.16 *Assume that, for some successful iteration (r,k) $(k \geq 0)$,*

$$\chi_{r,k} \geq \epsilon \tag{4.77}$$

and

$$f(x_{r,k}) - f(x_{r,k+1}) < \frac{\eta_1 \kappa_{red} (\frac{1}{2}\kappa_\chi)^{2r} \epsilon^2}{2\kappa_L} \tag{4.78}$$

for some $\epsilon \in (0,1)$. Then

$$\|s_{r,k}\|_\infty \leq \kappa_{acc} \, [\, f(x_{r,k}) - f(x_{r,k+1}) \,], \tag{4.79}$$

where

$$\kappa_{acc} \stackrel{\rm def}{=} \left(\frac{\kappa_P}{\mu}\right)^r \frac{1}{\kappa_{red}\eta_1(\frac{1}{2}\kappa_\chi)^r \epsilon}.$$

Proof. Assume that (4.77) and (4.78) hold at the successful iteration (r,k) and consider the subset of iterations given by $\mathcal{R}(r,k)$. If (r,k) is a Taylor iteration, then $\mathcal{R}(r,k) = \{(r,k)\}$ and the desired result follows from Lemma 4.2.13 and the inequality

$$\frac{1}{\kappa_{\rm red}\eta_1\epsilon} \leq \kappa_{\rm acc}.$$

If iteration (r,k) is recursive, consider a minimization sequence containing a recursively successful iteration for (r,k) at the deepest possible level in $\mathcal{R}(r,k)$. Let the index of this deepest level be d and note that every successful iteration in this minimization sequence must be recursively successful for (r,k). We will now prove the result by induction on the levels, from $d+1$ up to r. First, let $(d+1,q) = \pi(d,0)$ and assume that

$$\chi_{d+1,q} \geq (\tfrac{1}{2}\kappa_\chi)^{r-d-1}\epsilon, \tag{4.80}$$

which gives, in view of (4.27), that $\chi_{d,0} \geq (\frac{1}{2})^{r-d-1}\kappa_\chi^{r-d}\epsilon$. Each (recursively) successful iteration of our deepest minimization sequence must thus be a Taylor iteration. Because of Lemma 4.2.13, we then obtain that, as long as $\chi_{d,\ell} \geq (\frac{1}{2}\kappa_\chi)^{r-d}\epsilon$ and iteration (d,ℓ) is successful, we have that

$$\|s_{d,\ell}\|_\infty = \|x_{d,\ell} - x_{d,\ell+1}\|_\infty \leq \frac{1}{\kappa_{\rm red}\eta_1(\frac{1}{2}\kappa_\chi)^{r-d}\epsilon} \, [\, f_d(x_{d,\ell}) - f_d(x_{d,\ell+1}) \,],$$

We could then apply Lemma 4.2.15 for iteration $(d+1,q) = \pi(d,0)$ with

$$\epsilon_{\rm rsi} = (\tfrac{1}{2}\kappa_\chi)^{r-d-1}\epsilon \quad \text{and} \quad \kappa_{\rm rsi} = \frac{1}{\kappa_{\rm red}\eta_1(\frac{1}{2}\kappa_\chi)^{r-d}\epsilon},$$

if (4.75) holds. But note that Lemma 4.2.12 implies that

$$f_{d+1}(x_{d+1,q}) - f_{d+1}(x_{d+1,q+1}) \leq \mu^{d+1-r} \, [\, f(x_{r,k}) - f(x_{r,k+1}) \,]$$

which in turn gives (4.75) in view of (4.78), as desired. As a result of Lemma 4.2.15, we then deduce that

$$\begin{aligned} \|s_{d+1,q}\|_\infty &\leq \mu^{-1}\kappa_{\rm P}\kappa_{\rm rsi} \, [\, f_{d+1}(x_{d+1,q}) - f_{d+1}(x_{d+1,q+1}) \,] \\ &= \left(\frac{\kappa_{\rm P}}{\mu}\right) \frac{1}{\kappa_{\rm red}\eta_1(\frac{1}{2}\kappa_\chi)^{r-d}\epsilon} \, [\, f_{d+1}(x_{d+1,q}) - f_{d+1}(x_{d+1,q+1}) \,]. \end{aligned}$$

Consider now a minimization sequence at level j such that $d < j < r$, and such that this minimization sequence belongs to $\mathcal{R}(r,k)$. Then define $(j+1,t) = \pi(j,0)$ and assume, in line with (4.80), that $\chi_{j+1,t} \geq (\frac{1}{2}\kappa_\chi)^{j-1}\epsilon$ which yields in particular that $\chi_{j,0} \geq (\frac{1}{2})^{j-1}\kappa_\chi^j\epsilon$. Assume now that

$$\chi_{j,\ell} \geq (\tfrac{1}{2}\kappa_\chi)^j\epsilon,$$

that iteration (j,ℓ) is (recursively) successful, and that

$$\|s_{j,\ell}\|_\infty \leq \left(\frac{\kappa_{\rm p}}{\mu}\right)^j \frac{1}{\kappa_{\rm red}\eta_1(\frac{1}{2}\kappa_\chi)^j\epsilon}\,[\,f_j(x_{j,\ell}) - f_j(x_{j,\ell+1})\,].$$

Applying Lemma 4.2.12 and using (4.78), we may then apply Lemma 4.2.15 for iteration $(j+1,t)$, with

$$\epsilon_{\rm rsi} = (\tfrac{1}{2}\kappa_\chi)^{j-1}\epsilon \quad\text{and}\quad \kappa_{\rm rsi} = \left(\frac{\kappa_{\rm p}}{\mu}\right)^j \frac{1}{\kappa_{\rm red}\eta_1(\frac{1}{2}\kappa_\chi)^j\epsilon}.$$

This ensures that

$$\begin{aligned}\|s_{j+1,t}\|_\infty &\leq \mu^{-1}\kappa_{\rm p}\kappa_{\rm rsi}\,[\,f_{j+1}(x_{j+1,t}) - f_{j+1}(x_{j+1,t+1})\,]\\ &= \left(\frac{\kappa_{\rm p}}{\mu}\right)^{j+1} \frac{1}{\kappa_{\rm red}\eta_1(\frac{1}{2}\kappa_\chi)^j\epsilon}\,[\,f_{j+1}(x_{j+1,t}) - f_{j+1}(x_{j+1,t+1})\,].\end{aligned}$$

The induction is then completed, and the desired result follows since $d < j < r$. □

We finally prove the main result.

Theorem 4.2.17 *Assume that ϵ_r is "driven to zero" in Algorithm RMTR$_\infty$. Then*

$$\lim_{k\to\infty}\chi_{r,k} = 0.$$

Proof. As in Theorem 4.2.11, we identify our sequence of iterates with that generated by considering a sequence of tolerances $\{\epsilon_{r,j}\} \in (0,1)$ monotonically converging to zero. We start our proof by observing that the monotonic nature of the sequence $\{f(x_{r,\ell})\}_{\ell\geq 0}$ and the fact that $f(x)$ is bounded below impose that

$$f(x_{r,k}) - f(x_{r,k+1}) \to 0$$

for all successful iterations (r,k). Assume now, for the purpose of deriving a contradiction, that

$$\limsup_{k\to\infty}\chi_{r,k} \geq 3\epsilon > 0 \tag{4.81}$$

for some $\epsilon \in (0,1)$ and consider a $k_0 > 0$ such that $\chi_{r,k_0} \geq 2\epsilon$ and such that both (4.78) and

$$f(x_{r,k}) - f(x_{r,k+1}) \leq \frac{\epsilon}{\kappa_{\rm acc}\kappa_{\rm L}n_r} \tag{4.82}$$

hold for all $k \geq k_0$. Without loss of generality, we may assume that the minimization sequence at level r starts at iteration k_0. But Lemma 4.2.16 ensures that (4.79) holds for each successful iteration (r,k) $(k \geq k_0)$ as long as (4.77) holds. We may therefore apply Lemma 4.2.14 with

$$\epsilon_{\rm ncr} = 2\epsilon \quad\text{and}\quad \kappa_{\rm ncr} = \kappa_{\rm acc}$$

to the (truncated) minimization sequence at level r and deduce that (4.82) implies (4.71) and that (4.77) holds for all $k \geq k_0$, which is impossible in view of Theorem 4.2.11. Hence (4.81) is impossible and our proof complete. □

Theorem 4.2.17 implies, in particular, that any limit point of the infinite sequence $\{x_{r,k}\}$ is first-order critical for problem (4.13). But we may draw stronger conclusions: if we additionally assume that the trust region becomes asymptotically inactive at all levels, then, as explained above, each minimization sequence in the algorithm becomes infinite, and we may apply Theorem 4.2.17 to each of them, concluding that

$$\lim_{k \to \infty} \chi_{i,k} = 0$$

for every level $i = 0, \ldots, r$. The behavior of Algorithm RMTR$_\infty$ is therefore truly coherent with its multilevel formulation, since the same convergence results hold for each level.

The convergence results at the upper level are unaffected if minimization sequences at lower levels are "prematurely" terminated, provided each such sequence contains at least one successful iteration. Indeed, none of the proofs depends on the actual stopping criterion used. Thus, one might think of stopping a minimization sequence after a preset number of successful iterations: in combination with the freedom left at Step 1 to choose the model whenever (4.27) holds, this strategy allows a straightforward implementation of fixed lower-iterations patterns, like the V- or W-cycles in multigrid methods.

Our theory also remains essentially unchanged if we merely insist on first-order coherence (i.e., (4.25)) to hold only for small enough trust-region radii $\Delta_{i,k}$, or only up to a perturbation of the order of $\Delta_{i,k}$ or $\|g_{i,k}\|\Delta_{i,k}$. Other generalizations may be possible. Similarly, although we have assumed for motivation purposes that each f_i is "more costly" to minimize than f_{i-1}, we have not used this feature in the theory presented above, nor have we used the form of the lower levels' objective functions. Nonconstant prolongation and restriction operators of the form $P_i(x_{i,k})$ and $R_i(x_{i,k})$ may also be considered, provided the singular values of these operators remain uniformly bounded. We refer the reader to Gratton et al. [23] for a discussion of convergence properties of multilevel trust-region methods to second-order critical points.

4.3 Practical Implementation

The ℓ_∞ version of the RMTR method has been successfully implemented in the Fortran programming language, yielding a very powerful method for the solution of large-scale unconstrained and bound-constrained optimization problems. Although it has not been done by this author, we feel it is important to motivate our results by mentioning the excellent numerical experiments obtained with this method. The reader is referred to Gratton et al. [26] and Tomanos [65] for the complete results and details on the practical implementation. In this implementation, the method is slightly modified from Algorithm 4.2.1. In this case, when we have reached the lowest level (that is, the number of variables is very low), we can apply any of the methods mentioned in Chapter 3; in particular, the (Projected) Truncated Conjugate Gradient method can be very efficient (see Conn et al., [8] and [9]).

At finer levels ($i > 0$), an adaptation of smoothing techniques (such as the Gauss-Seidel method, described in Section 2.3.2 on Chapter 2) to has been devised that generates a step which satisfies the sufficient decrease condition (4.29). Here, this method proceeds by successively minimizing the model (4.23) along each coordinate axes, taking into account the bound constraints on the problem, provided that the curvature of this model along each axis is positive. It is, in essence, an adaptation of the Sequential Coordinate Minimization method (SCM) to bound-constrained problems (see, for example, Ortega and Rheinboldt [52] for a description of this method).

Thus, consider the minimization of model (4.23) along the jth axis starting from s such that $\nabla m_{i,k}(x_{i,k} + s) \stackrel{\text{def}}{=} g$. If the jth diagonal entry of $H_{i,k}$ is positive, this minimization results in updating

$$\alpha_j = \text{Proj}_{\mathcal{W}_{i,k}}\left(\frac{-[g]_j}{[H_{i,k}]_{jj}}\right), \tag{4.83}$$

$$[s]_j = [s]_j + \alpha_j \tag{4.84}$$

$$[g]_j = [g]_j + \alpha_j H_{i,k} e_i^{[j]}, \tag{4.85}$$

where $\text{Proj}_{\mathcal{W}_{i,k}}(\cdot)$ is the orthogonal projection on the intersection of all the constraints at level i, and where we denote by $[v]_j$ the jth component of vector v and $[M]_{jj}$ the jth diagonal entry of matrix M, and where $e_i^{[j]}$ is the jth vector of the canonical basis of $\mathbb{R}^{n_i}$.

On the other hand, if $[H_{i,k}]_{jj} \leq 0$, then we take a step along the jth coordinate direction which intersects the boundary of the $\mathcal{W}_{i,k}$ and update the gradient of the model. We refer to each set of n_i successive coordinate minimizations as a *smoothing cycle*, and a sequence of one or more cycles defines a *smoothing iteration*.

In order to guarantee that the step computed by one or more of such cycles satisfies the sufficient model decrease condition, we must start the first smoothing cycle by selecting the j_mth axis, where

$$j_m = \underset{j}{\text{argmin}}\, [g_{i,k}]_j [d_{i,k}]_j, \tag{4.86}$$

where

$$d_{i,k} = \underset{\substack{x_{i,k}+d\in\mathcal{L}_i \\ \|d\|_\infty \leq 1}}{\text{argmin}} \langle g_{i,k}, d\rangle. \tag{4.87}$$

By doing this, the minimization of this model is guaranteed to yield a generalized Cauchy step such as the one described in Section 1.4.1.1, Chapter 1, as shown in the following result, which appears in Gratton et al. [26]:

Theorem 4.3.1 *Assume that the first unidimensional minimization in the first smoothing cycle at iteration (i,k) is performed along the j_mth coordinate axis, where j_m is determined by (4.86) and (4.87), and results in a step size α_{j_m}. Then, (4.29) holds for $s_{i,k} = \alpha_{j_m} e_i^{[j_m]}$.*

Figure 4.5 shows the performance profile obtained by Gratton et al. [26] with this new method.

It is clear this method is an excellent alternative to classical trust-region methods in a large-scale context, and a practical implementation in Fortran is included in the most recent version of the GALAHAD library of nonlinear solvers (see Gould et al. [22]).

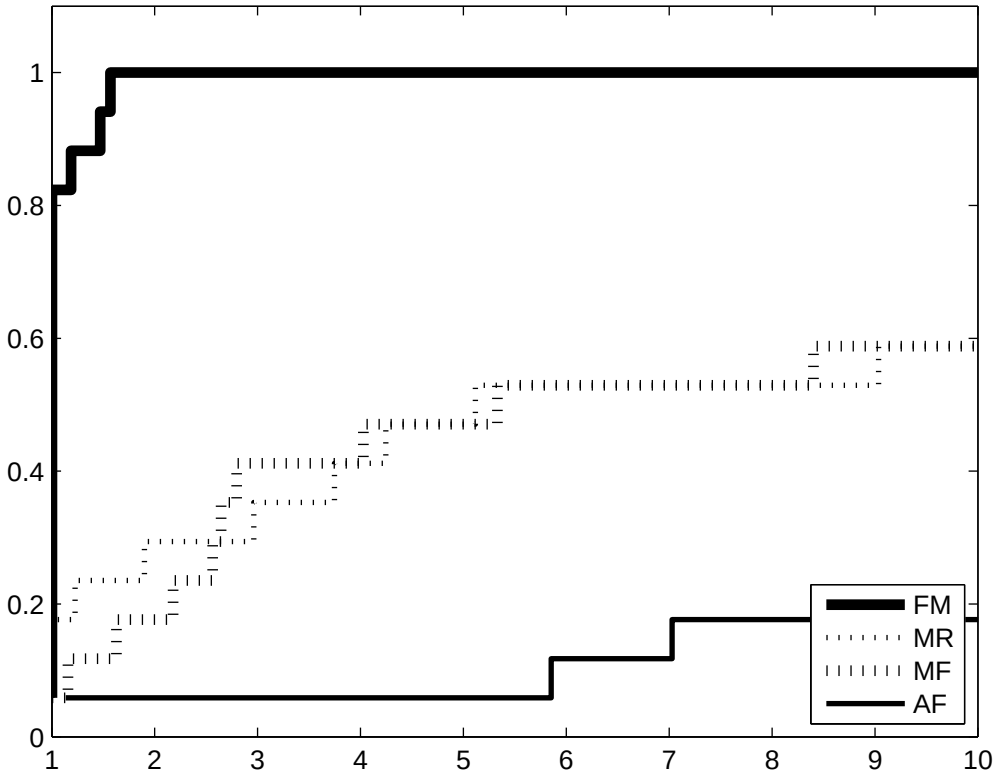


Figure 4.5: The performance profile shows the results obtained for 4 versions of the algorithm. The All-on-Finest version (equivalent to the classical trust-region method) is denoted by AF. The Multigrid-on-Finest version, where we start the resolution of the problem in the finest level, and from then on use the multilevel strategy just described, is denoted by MF. The Mesh Refinement version is equivalent to the technique described in Chapter 2, where the resolution of the problem starts at the lowest level, and we do not use recursive steps in the algorithm. It is denoted by MR. Finally, the Full Multigrid version, denoted by FM, can be seen as the combination of the Mesh Refinement and Multigrid-on-Finest techniques.

4.4 Conclusions

We have presented a variant of the recursive multilevel RMTR algorithm for unconstrained nonlinear optimization that appears to have advantages over the original method in terms of computational costs and flexibility. The use of the infinity norm (as opposed to the Euclidean norm used in the original algorithm) removes the need for costly preconditioning of the trust-region and adapts very naturally to bound constrained problems. However, and despite the conceptual similarity between RMTR and the new algorithm, their convergence theories differ significantly. Fortunately, the same strong global convergence results can be proved (with somewhat simpler arguments) for the new algorithm, which makes it very attractive for practical use.

Chapter 5

Multilevel Derivative-Free Optimization

In unconstrained optimization, one of the problems that may arise in the implementation of a practical algorithm is the difficulty in computing the derivatives of the objective function. Indeed, in many applications, these derivatives may be unavailable or very costly, for example, if they are the result of an actual simulation or the solution of another complicated problem. This is typically the case in problems where the objective function can only be obtained by a "black box" procedure, and there is no information available for the computation of its derivatives. In this case, one is interested in algorithms that do not require the derivatives of the objective function. These methods belong to a class called Derivative-Free Optimization (DFO).

Here, we are interested in solving the unconstrained optimization problem (1.1), but we assume that although the first and second derivatives of f exist, they are unavailable.

Several methods have been proposed to solve this class of problems. These methods can be divided into three distinct classes. First, there are those that seek to simulate the derivatives of the function, either by approximation, for example by finite differences (see Gill et al. [19], Dennis and Schnabel [14] and Nocedal and Wright [50]) or by automatic differentiation procedures (see Griewank and Corliss [28] and Griewank [27] for a survey on the subject).

Another class of methods is based on sampling, that is, it is based on available information obtained by the computation of the objective function in sample points inside the region where we must minimize the objective function. Important examples of this class of methods include the Nelder-Mead Algorithm (Nelder and Mead [48]) and, more recently, Pattern Search and Generalized Pattern Search methods (see, for example, Torczon [66]).

Our interest here, however, is in *model-based* approaches, that is, methods in which we try to approximate the objective function using a *surrogate* model, and expecting this model to simulate the behavior of the objective function in a region around each iterate. Among these methods, we focus more precisely on the ones based on trust-region approaches, for example those of Powell ([53], [54], [57], [58], [59]), and those of Conn and Toint [7], and Conn et al. ([10], [11]). In these methods, the Taylor model is replaced by a more general quadratic model, which consists, at each iteration, in a model of the objective function built using quadratic interpolation on a set of sampled points contained in a region around the current iterate.

The main drawback of this type of method is that it tends to be very slow and cost too many function evaluations in problems with a large number of variables. In fact, most derivative-

free methods available today are not able to solve a problem with more than a few hundred variables. In this chapter, we aim to present a new algorithm for unconstrained derivative-free optimization that is also based on trust-region techniques, while exploring some of the multilevel ideas presented earlier in this thesis, in order to improve on the performance of this method.

This chapter is organized as follows. In Section 5.1, we will briefly describe the ideas behind the derivative-free trust-region algorithm. Then, in Section 5.2 we will discuss a possible multilevel implementation of this algorithm. Finally, in Section 5.3 we will present some preliminary results that, we hope, will show the relevance of this new implementation.

5.1 Derivative-Free Trust-Region Optimization

As we have seen previously, when solving the unconstrained minimization problem (1.1) by using a trust-region method, we must build, at each iteration, a model that can be minimized inside the trust region, so that we can compute a step. When the gradient and Hessian of f are not available, we can still build a quadratic model that interpolates the objective function in a set of points chosen around the current iterate.

More formally, starting from a given point $y_0 = x_k$ (which we will call the *base point* for the interpolation set), we choose a set of $p_1 = p + 1$ points

$$Y_k = \{y_0, y_1, \ldots, y_p\},$$

and try to define a quadratic model q_k such that

$$q_k(y_j) = f(y_j), \qquad \text{for } j = 0, ..., p. \tag{5.1}$$

If

$$p + 1 = 1 + n + \tfrac{1}{2}n(n+1) = \tfrac{1}{2}(n+1)(n+2) \stackrel{\text{def}}{=} \hat{p} + 1,$$

then a quadratic interpolation function q_k can be entirely determined by the equations (5.1), with

$$q_k(x) = f(x_k) + \langle g_k, x - x_k\rangle + \tfrac{1}{2}\langle x - x_k, H_k(x - x_k)\rangle, \tag{5.2}$$

where $g_k \in \mathbb{R}^n$ and $H_k \in \mathbb{R}^{n\times n}$ is symmetric. This model can then be minimized inside a trust-region framework in order to generate the next iterate for the method.

5.1.1 Interpolation model

Let us drop the iteration indices for the sake of simplicity for now, defining $q = q_k$ as the quadratic model we are trying to determine. In order to determine $g = g_k$ and $H = H_k$, suppose that we have a basis $\{\phi_i\}_{i=0}^{p}$ for the space of quadratic functions from $\mathbb{R}^n$ to $\mathbb{R}$. Then, any quadratic function in this space can be written as a linear combination of these basis functions, i.e.

$$q(x) = \sum_{i=0}^{p} \alpha_i \phi_i(x),$$

where $\alpha_i \in \mathbb{R}$ for all $i = 0, \ldots, p$. Thus, from (5.1), we must have that

$$f(y_j) = \sum_{i=0}^{p} \alpha_i \phi_i(y_j), \qquad \text{for } j = 0, \ldots, p. \tag{5.3}$$

This is a linear system in the coefficients $\alpha_i, i = 0, \ldots, p$ which is nonsingular if and only if the determinant

$$\delta(Y) = \det \begin{pmatrix} \phi_0(y_0) & \cdots & \phi_p(y_0) \\ \vdots & \ddots & \vdots \\ \phi_0(y_p) & \cdots & \phi_p(y_p) \end{pmatrix} \tag{5.4}$$

is nonzero. In this case, we say that the set of interpolation points $Y = \{y_0, \ldots, y_p\}$ is *poised*, which means that the quadratic interpolation polynomial q is uniquely determined by the interpolation conditions (5.1).

The notion of poisedness is very important in the definition of a derivative-free optimization method, since it is usually not sufficient to have a complete set of interpolation points. Indeed, these points must satisfy some *geometry* requirement, so that the interpolation function obtained with these points represents well the objective function.

5.1.2 Basis Functions

There are several possible choices for the definition of the basis $\{\phi_i\}_{i=0}^{p}$ of quadratic polynomial functions that we can use in order to interpolate the objective function in a given set of interpolation points Y.

One classical choice is to use the *Lagrange polynomials*, which satisfy the relationship

$$L_i(y_j) = \delta_{ij} \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise} \end{cases}, \text{ for all } y_j \in Y, \text{ for } i = 0, \ldots, p.$$

The Lagrange interpolation polynomials are, however, not the only choice possible. Another possibility that has proven to be extremely efficient is that of *Newton Fundamental Polynomials*.

Suppose that we want to build a model of degree $d = 2$, and that we have $p + 1$ points. Suppose also that we can organize these points in $d + 1$ blocks, such that

$$Y^{[\ell]} = \{y_1^{[\ell]}, \ldots, y_{|Y^{[\ell]}|}^{[\ell]}\}, \quad (\ell = 0, 1, 2),$$

where the $\ell-$th block contains

$$|Y^{[\ell]}| = \begin{pmatrix} \ell + n - 1 \\ \ell \end{pmatrix}$$

points. To each point $y_i^{[\ell]} \in Y^{[\ell]}$ corresponds a single *Newton Fundamental Polynomial* of degree ℓ satisfying

$$N_i^{[\ell]}(y_j^{[m]}) = \delta_{ij}\delta_{\ell m} \text{ for all } y_j^{[m]} \in Y^{[m]} \text{ with } m \leq \ell.$$

The main advantage of these polynomials is that they are very easy to compute. In fact, the procedure that builds these polynomials can be seen as a Gram-Schmidt orthogonalization procedure applied to the initial polynomial basis (usually chosen as the monomial basis) with

respect to the inner product defined in this space as

$$\langle P, Q\rangle = \sum_{y\in Y} P(y)Q(y).$$

This procedure is presented in Algorithm 5.1.1.

Algorithm 5.1.1: Newton Fundamental Polynomial Basis Computation

Step 0: Initialization. Set the $N_i^{[\ell]}, i = 1, \ldots, |Y^{[\ell]}|, \ell = 0, 1, 2$ to the chosen initial polynomial basis. Set $Y_{\text{temp}} = \emptyset$.

Step 1: Loop over the polynomials. For $\ell = 0, 1, 2$ and $i = 1, \ldots, |Y^{[\ell]}|$,

- Choose some $y_i^{[\ell]} \in Y \setminus Y_{\text{temp}}$ such that $|N_i^{[\ell]}(y_i^{[\ell]})| \neq 0$; if no such $y_i^{[\ell]}$ exists in $Y \setminus Y_{\text{temp}}$, reset $Y = Y_{\text{temp}}$ and stop prematurely with an incomplete Newton polynomial basis.
- Update the interpolation set by $Y_{\text{temp}} = Y_{\text{temp}} \cup \{y_i^{[\ell]}\}$
- Normalize the current polynomial by

$$N_i^{[\ell]}(x) = \frac{N_i^{[\ell]}(x)}{N_i^{[\ell]}(y_i^{[\ell]})}$$

- Update all Newton polynomials in block ℓ and above by

$$N_j^{[k]}(x) = N_j^{[k]}(x) - N_j^{[k]}(y_i^{[\ell]})N_i^{[\ell]}(x),$$

for $j = 1, \ldots, |Y^{[m]}|$, $m = \ell, \ell + 1, \ldots, d$, $i \neq j$.

Now, we must build a polynomial interpolant for our objective function using the polynomials obtained by this procedure. This is done by a procedure that uses so-called *generalized finite differences*, defined recursively by

$$\begin{aligned} \lambda_0(x) &\stackrel{\text{def}}{=} f(x) \\ \lambda_{\ell+1}(x) &\stackrel{\text{def}}{=} \lambda_\ell(x) - \sum_{i=1}^{|Y^{[\ell]}|} \lambda_\ell(y_i^{[\ell]})N_i^{[\ell]}(x), \text{ for } \ell = 0, 1. \end{aligned} \tag{5.5}$$

With this definition, Theorem 5.1.1 shows how we can build our interpolation model.

Theorem 5.1.1 *Suppose that the Newton fundamental polynomials $N_i^{[\ell]}(x)$ are defined for $\ell = 0, 1, 2$ and $i = 1, \ldots, |Y^{[\ell]}|$. Then,*

$$q(x) = \sum_{\ell=0}^{2} \sum_{i=1}^{|Y^{[\ell]}|} \lambda_\ell(y_i^{[\ell]})N_i^{[\ell]}(x) \tag{5.6}$$

is well defined and satisfies the interpolation conditions (5.1).

Although they seem reasonable, definitions (5.5) and (5.6) result in a very slow and inefficient procedure. Fortunately, we can devise a more efficient procedure, described in Algorithm 5.1.2.

Algorithm 5.1.2: Generalized Finite Differences Computation

Step 0: Initialization. For $i = 1, \ldots, |Y^{[\ell]}|$ and $\ell = 0, 1, 2$, set $\lambda_{i,\ell} = f(y_i^{[\ell]})$.

Step 1: Consider the blocks by increasing index. For $k = 1, 2$, compute successively

$$\lambda_{i,\ell} = \lambda_{i,\ell} - \sum_{j=1}^{|Y^{[k-1]}|} \lambda_{j,k-1} N_j^{[k-1]}(y_i^{[\ell]}),$$

for $i = 1, \ldots, |Y^{[\ell]}|$ and $\ell = 0, 1, 2$.

5.1.3 Model computation and Algorithm

Another advantage of the Newton fundamental polynomials is that the determinant (5.4) is never computed directly in order to check for poisedness. In fact, since we will divide every polynomial in the basis by $N_i^{[\ell]}(y_i^{[\ell]})$, which we call a *pivot*, we must check if this quantity is positive enough. On the other hand, if there is a $\theta > 0$ such that, for all $\ell = 0, 1, 2$ and $i = 1, \ldots, |Y^{(\ell)}|$,

$$|N_i^{[\ell]}(y_i^{[\ell]})| \geq \theta, \tag{5.7}$$

then it is possible to show that Y is poised (see Sauer and Xu [61], for example, or Theorem 9.4.2, p. 330 in Conn et al. [12]). Thus, checking for poisedness is automatic and we do not need to do any additional computations in order to do this. If one or more of these pivots are too small, then the algorithm stops and the interpolation model is not complete; however, the resulting incomplete model is poised, even if the original set was not. Moreover, we can obtain an estimate for the interpolation error in this case. For this, we must ensure that the model is *adequate* in a certain region $\mathcal{Q}(\delta)$, which is a hypersphere of radius $\delta > 0$; this means that

- The model is at least fully linear, that is $|Y| \geq n + 1$;
- $y \in \mathcal{Q}(\delta)$, for all $y \in Y$;
- $|N_i^{[\ell]}(y_j^{[\ell+1]})| \leq \kappa_1$, $i = 1, \ldots, |Y^{[\ell]}|$, $j = 1, \ldots, |Y^{[\ell+1]}|$, $\ell = 0, 1$;
- $|N_i^{[2]}(x)| \leq \kappa_1$, for $i = 1, \ldots, |Y^{[2]}|$, $x \in \mathcal{Q}(\delta)$, with $\kappa_1 > 1$.

This test is essential in the algorithm, in that it guarantees the accuracy of the interpolated gradient and thus of the criticality measure used as a stopping rule. Furthermore, it allows us to prove (see Theorem 9.4.4 in p. 333 on Conn et al. [12]) that

$$|f(x) - q_k(x)| \leq \kappa \max[\delta^2, \delta^3],$$

for all $x \in \mathcal{Q}(\delta)$ and some constant $\kappa > 0$ independent of k.

Now, once the interpolation model has been built, we must compute a step using one of the many algorithms available for the solution of the trust-region subproblem, for example, the Moré-Sorensen method or the Truncated Conjugate-Gradient algorithm, described in Sections 3.1.1 and 1.3.3 in Chapter 3. When the step has been computed, we compute the new function value at the new point $x_k + s_k$, and if the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{q_k(x_k) - q_k(x_k + s_k)}$$

is greater than a constant $1 \geq \eta_0 > 0$, the step is accepted as the new iterate. Otherwise, the step is rejected and we will possibly decrease the trust-region radius in order to compute a new step.

In both cases, we must decide how to include these new points in the interpolation set. Indeed, since we are looking to use as many points as possible to form our interpolation model, while evaluating the objective function as little as possible, since the value of f has already been computed in both cases, it might be useful to include this points in the interpolation set, either if it is not yet complete, or if including this point in place of another would improve on the poisedness of Y.

If the step is accepted, then the new iterate $x_{k+1} = x_k + s_k$ must be chosen as the base point for the new interpolation set. In this case, we include the new iterate on the set directly and recompute the Newton fundamental polynomials if the set is incomplete. If the set is already complete, we must decide which point to drop in order to include the new one. Since we have seen that the pivots in Algorithm 5.1.1 indicate the quality of the interpolation set Y, one possible choice is to drop the point $y_i^{[\ell]}$ which gives the smallest pivot in the interpolation set, that is

$$|N_i^{[\ell]}(y_i^{[\ell]})| = \min_{\substack{t=0,1,2 \\ j=1,\ldots,|Y^{[t]}|}} |N_j^{[t]}(y_j^{[t]})|. \tag{5.8}$$

If the step $x_k + s_k$ is rejected, we should also check if including it in the interpolation set improves on the geometry of Y. Thus, we will look for $y_i^{[\ell]}$ that satisfies (5.8), and we will check if the replacement is worthwhile, by computing the ratio

$$\frac{|N_i^{[\ell]}(x_k + s_k)|}{|N_i^{[\ell]}(y_i^{[\ell]})|}.$$

If this is larger than some predefined constant c_1 (for example, $c_1 = 2$), we can decide to replace $y_i^{[\ell]}$ with $x_k + s_k$ and continue with our algorithm. This strategy has also been discussed in [10].

Another point which we must discuss is that of geometry improvements on the interpolation set. Indeed, if on a given iteration the interpolation model is not *adequate* in $\mathcal{Q}(\delta)$, we might choose to improve on the geometry of Y by replacing some of its points by new ones. Again, the strategies vary, but one must, firstly, make sure that $y \in \mathcal{Q}(\delta)$, for all $y \in Y$. Thus, all points y_j such that

$$\|y_j - y_0\|_2 > \delta$$

are removed from the interpolation set. Additionally, one could choose to eliminate points for which the pivots are too small, that is, $y_j^{[\ell]}$ such that

$$|N_j^{[\ell]}(y_j^{[\ell]})| < c_2\theta,$$

where $c_2 > 0$ is some predefined constant.

Once we have done this, we must include new points in the set in order to make it (at least) fully linear. One possible strategy is to replace each $y_i^{[\ell]}$ removed from the interpolation set with

$$y_+ = \arg\max_{x \in \mathcal{Q}(\delta)} |N_i^{[\ell]}(x)|.$$

By doing this, it is possible to show that in a finite number of such improvement steps are guaranteed to make the model valid in a region $\mathcal{Q}(\delta)$. This can be found in Conn et al. [10], along with the global convergence theory for this type of methods.

We present in Algorithm 5.1.3 on page 104 a basic derivative-free trust-region algorithm. The constants

$$0 < \eta_0 \leq \eta_1 < 1,\ 0 < \gamma_0 \leq \gamma_1 < 1 \leq \gamma_2,\ \epsilon_g > 0 \text{ and } \mu \geq 1$$

are given.

5.1.4 Extensions

In order to be able to treat larger problems, one could make use of the sparse structure of the problem to be solved. In particular, in discretized problems, this structure is usually well defined, in that we know the sparsity pattern of the given Hessian. In this case, there is a symmetric index set defined as

$$\mathcal{S} = \{(i, j) | 1 \leq i, j \leq n \text{ and } \langle e^{[i]}, \nabla^2 f(x) e^{[j]} \rangle = 0 \forall x \in \mathbb{R}^n\}.$$

In this case, as has been shown by Colson and Toint [5], it is very advantageous to eliminate from the initial monomial basis functions those polynomials corresponding to pairs in the set $\mathcal{S}$, that is, those of the type $x_i x_j$, for all $(i, j) \in \mathcal{S}$. This results in a partial basis of Newton Fundamental Polynomials, which results in much less computational effort in order to compute the interpolation model.

Another very useful strategy can be used when the function to be minimized is *partially separable*, which means it can be written as

$$f(x) = \sum_{i=1}^{M} f_i(U_i x), \tag{5.9}$$

where each f_i is called an *element function* which depends on the so called *internal variables* $U_i x$, with U_i a $n_i \times n$ matrix, and where usually, n_i is much smaller than n. This decomposition has been introduced by Griewank and Toint [30].(1)

Now, since we have M functions that define the objective function f, we might choose to interpolate this function f by interpolating each f_i, using different interpolation sets Y_i, and building M different quadratic models

$$q_{i,k}(x_k + s) = f_i(x_k) + \langle g_{i,k}, s \rangle + \frac{1}{2} \langle s, H_{i,k} s \rangle,$$

(1) In particular, it has been shown in Griewank and Toint [29] that every twice-continuously differentiable function with a sparse Hessian is partially separable.

Algorithm 5.1.3: Basic DFO Algorithm

Step 0: Initialization. Given x_0 and $f(x_0)$, choose an interpolation set Y_0 containing x_0 with $p+1$ ($1 \leq p \leq \hat{p}$) points. Choose an initial trust-region radius Δ_0, and set $k = 0$.

Step 1: Model and Step Computation.

1.a: Model Computation. Setting $y_0 = x_k$, build a model $q_k(x_k + s_k)$ of the form (5.2), using Y_k and such that the interpolation conditions (5.1) are satisfied.

1.b: Criticality test. If $\|g_k\| \leq \epsilon_g$, test if q_k can be improved (i.e. made *valid*) in some region $\mathcal{Q}_k(\delta)$ for some $\delta > 0$, possibly increasing $|Y_k|$, and return to Step 1.a. If the model cannot be further improved, return with solution $x^* = x_k$. Otherwise, go to Step 1.c.

1.c: Step Computation. Compute a step s_k such that

$$q_k(x_k + s_k) = \min_{\|s\| \leq \Delta_k} q_k(x_k + s).$$

Compute $f(x_k + s_k)$ and

$$\rho_k \stackrel{\text{def}}{=} \frac{f(x_k) - f(x_k + s_k)}{q_k(x_k) - q_k(x_k + s_k)}.$$

Step 2: Interpolation set update. If $\rho_k \geq \eta_1$, define $x_{k+1} = x_k + s_k$ and include x_{k+1} in the interpolation set Y_k, replacing one of the existing points if $p = \hat{p}$ and define $\hat{\rho}_k = \rho_k$. Else, try to include $x_k + s_k$ in Y_k. If this is not possible, try to improve the geometry of the interpolation set. If new points are added to the interpolation set, define $\hat{y} \in Y_k$ such that $f(\hat{y}) = \min_{y \in Y_k} f(y)$. Compute

$$\hat{\rho}_k = \frac{f(x_k) - f(\hat{y})}{q_k(x_k) - q_k(x_k + s_k)}.$$

Step 3: Trust-region radius update. Set

$$\Delta_{k+1} = \begin{cases} [\Delta_k, \gamma_2 \Delta_k) & \text{if } \hat{\rho}_k \geq \eta_1, \\ [\gamma_0 \Delta_k, \gamma_1 \Delta_k] & \text{if } \hat{\rho}_k < \eta_1 \text{ and } Y_k \text{ is valid}, \\ \Delta_k & \text{otherwise.} \end{cases}$$

Increment k by 1 and go to Step 1.

where each $g_i, k \in \mathbb{R}^{n_i}$ and $H_{i,k}$ is a matrix in $\mathbb{R}^{n_i \times n_i}$. Thus, each model $q_{i,k}$ approximates f_i around the *projection* of x_k into R^{n_i}, defined by $U_i x_k$.

Once all M models $q_{i,k}$ have been computed, we will build g_k and H_k from the partial $g_{i,k}$

and $H_{i,k}$, ultimately building a complete model of the form

$$q_k(x_k + s) = \sum_{i=1}^{M} q_{i,k}(x_k + s) = f(x_k) + \langle g_k, s\rangle + \frac{1}{2}\langle s, H_k\rangle.$$

This model can then be minimized inside the trust-region, and the method can proceed as described in the previous section.

However, since we must manage M separate interpolation sets, it is not clear how to do the model improvement described in the previous section. In particular, while adding new points to each interpolation set, these new points will be vectors of n_i components, and it is unclear which values to assign to the remaining $n - n_i$ components of this vector.

One alternative that has been discussed by Colson and Toint [6] is the CPR procedure by Curtis et al. [13], which has been adapted by Powell and Toint [60] to estimate sparse Hessians. For this, assume that we can write a $M \times M$ matrix D which contains

$$d_{ij} = \begin{cases} 1, & \text{if the function } f_i \text{ depends on } x_j \\ 0, & \text{otherwise.} \end{cases}$$

where $i = 1, \ldots, M$ and $j = 1, \ldots, n$. At iteration k, we define a set L_k of all indices $i \in [1, \ldots, M]$ such that a geometry improvement is requested for set $Y_{i,k}$. If $L_k \neq \emptyset$, we can partition the columns of D into subsets containing columns associated with individual functions whose index sets have an empty intersection. That is, in a given subset $L_{k,\cdot}$, we find columns for which the associated individual functions do not share a common internal variable $U_i x$.

One procedure that can be used to generate these partition of the columns of D can be described as follows. Consider $L_k = \{i_1, \ldots, i_L\}$.

- Create a group $L_{k,1} = \{i_1\}$.
- Check if f_{i_2} and f_{i_1} do not depend on the same variables. If this is the case, include i_2 in $L_{k,1}$. Otherwise, create a new group $L_{k,2}$.
- Consider all other indices in L_k, repeating the procedure until all indices have been included in either an old subset, or a new one.

This procedure is referred to as the *greedy* approach. Other more sophisticated techniques can be used, such as the graph coloring method of Coleman and Moré.

These techniques can be very useful if one wants to take advantage of structure, but there are other kinds of structure that can be exploited by a DFO algorithm. Here, the main cost of each trust-region iteration is not the computation of the step, as it is with most trust-region methods, but rather the construction of the model. In fact, for each quadratic model that we wish to build, we must solve the interpolation equations (5.3). Since this is equivalent to solving a linear system of dimension p, which is typically of the order of n^2, this can be very costly. This is the main motivation for the Section 5.2, where we describe possible "cheaper" models that can be constructed when the problem has an underlying multilevel structure.

5.2 Multilevel Alternatives

Now, suppose once more that we have a set of different descriptions of the objective function $\{f_i\}_{i=0}^r$ where each f_i is a function from $\mathbb{R}^{n_i}$ to $\mathbb{R}$ which is twice continuously differentiable. Each function f_{i-1} is assumed to be simpler than f_i, for all $i = 1, \ldots, r$. Suppose also that, for each i, there are full-rank operators $P_i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ and $R_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i-1}}$ (the prolongation and restriction, respectively). In this section, the prolongation is the linear interpolation operator and the restriction is taken as $R_i = P_i^T$, such that $\sigma_i = 1$.

We want to use the simpler representations of $f_r \stackrel{\text{def}}{=} f$ to compute a step for the trust-region subproblem at each iteration k. Thus, we will suppose that, at each level i and at each iteration k, we can find an interpolation set $Y_{i,k}$ containing $x_{i,k}$, such that f_i can be interpolated at this level by a (linear or quadratic) model $q_{i,k}$ around $x_{i,k}$ that satisfies

$$q_{i,k}(y_{i,j}) = f_i(y_{i,j}), \text{ for all } y_{i,j} \in Y_i. \tag{5.10}$$

5.2.1 Model Choices

As in any trust-region method, at iteration k at level i we want to compute a step $s_{i,k}$. When we have two levels of description for the objective function, the idea is that we can use f_{i-1} to build a model for f_i at level $i-1$. The minimization of such a model yields a step s_{i-1}, such that the prolongated step $s_i \stackrel{\text{def}}{=} P_i s_{i-1}$ can be used to compute a new iterate at level i.

There are several ways this can be done, and we will discuss a few of them here. For future reference, we define the quadratic interpolation model at level i around $x_{i,k} \stackrel{\text{def}}{=} y_{i,0}$ as

$$q_i(x_{i,k} + s_i) = f_i(x_{i,k}) + \langle g_i, s_i \rangle + \tfrac{1}{2}\langle s_i, H_i s_i \rangle, \tag{5.11}$$

and the linear interpolation model at level i around $x_{i,k}$ as

$$q_i(x_{i,k} + s_i) = f_i(x_{i,k}) + \langle g_i, s_i \rangle. \tag{5.12}$$

In practice, Step 1 of Algorithm 5.1.3 will be replaced by some other mechanism for the step computation, possibly involving the construction of more than one interpolation model. We will restrict ourselves to the two-level case here, but the algorithm can easily be expanded into a recursive algorithm if desired. We will denote by $q_{r,k}$ the interpolation model at level r and iteration k, around the current iterate $x_{r,k}$, and the initial base point in the lower-level interpolation set $Y_{r-1,0}$ is defined as $x_{r-1,0} = R_i x_{i,k}$.

It is also important to note that, since we are using an Euclidean-norm trust region in our algorithm, we must follow the ideas of Gratton et al. [25] (presented in Section 4.1, Chapter 4 of this thesis) in the definition of the lower-level trust-region. In other words, if the upper-level trust region is defined as

$$\mathcal{B}_{r,k} = \{x \in \mathbb{R}^{n_r} \,|\, \|x - x_{r,k}\|_r \leq \Delta_{r,k}\},$$

then the trust-region used at level $r-1$ to compute the step for iteration k at level r is defined by

$$\mathcal{B}_{r-1} = \{x \in \mathbb{R}^{n_{r-1}} \,|\, \|x - x_{r-1,0}\|_{r-1} \leq \Delta_{r,k}\},$$

where

$$\|s\|_{r-1} = \sqrt{\langle s, M_{r-1}s\rangle} = \|s\|_{M_{r-1}},$$

for some symmetric matrix $M_{r-1} \stackrel{\text{def}}{=} Q_{r-1}^T Q_{r-1}$, where $Q_i = P_r \ldots P_{i+2}P_{i+1}$ and $M_r = I$. In this case, $\|s\|_r \stackrel{\text{def}}{=} \|s\|_2$. As in the RMTR method, since we have to make sure that the step $P_r s_{r-1}$ stays inside the upper-level trust region, we must stop the computation of the lower-level step if

$$\|x_{r-1,q} - x_{r-1,0}\|_{r-1} > (1 - \epsilon_{r-1}^{\Delta})\Delta_{r,k}, \tag{5.13}$$

with $\epsilon_{r-1}^{\Delta} \in (0, 1)$.

The model to be minimized at level $r - 1$, which can combine both the interpolation models of levels r and $r - 1$, will be denoted by h_{r-1}. This is done to distinguish from the pure interpolation model at level $r - 1$ at iteration k, denoted by $q_{r-1,k}$. We also denote by

$$\mathcal{Q}_{i,k}(\delta) \stackrel{\text{def}}{=} \{x \in \mathbb{R}^{n_i} | \|x - x_{i,k}\| \leq \delta_i\}$$

the validity region of the model at each iterate k at level i, for some $\delta_i > 0$.

Algorithm 5.2.1 in page 108 describes how we build our model and compute the step at each iteration.

Algorithm 5.2.1: Multilevel Model Choice and Step Computation.

1.a: Model Choice. Set $y_0 = x_{i,k}$. If we are at the lowest level, go to Step 1.a.1. Otherwise, choose between 1.a.1 and 1.a.2.

1.a.1: Interpolation model. Build an interpolation model $q_{i,k}(x_{i,k} + s_i)$, using $Y_{i,k}$ and such that the interpolation conditions (5.10) are satisfied for f_i.

1.b.1: Criticality test. If $\|g_{i,k}\| \leq \epsilon_g$, test if $q_{i,k}$ can be improved in some region $\mathcal{Q}_{i,k}(\delta_i)$ for some $\delta_i > 0$, possibly increasing $|Y_{i,k}|$, and return to Step 1.a.1. If the model cannot be further improved, return with solution $x_i^* = x_{i,k}$.

1.c.1: Step Computation. Compute a step $s_{i,k}$ that (approximately) minimizes $q_{i,k}(x_{i,k}+s_i)$ inside the trust region defined by $\Delta_{i,k}$ and compute $f_i(x_{i,k} + s_{i,k})$ and

$$\rho_{i,k} \stackrel{\text{def}}{=} \frac{f_i(x_{i,k}) - f_i(x_{i,k} + s_{i,k})}{q_{i,k}(x_{i,k}) - q_{i,k}(x_{i,k} + s_{i,k})}.$$

Go to Step 2.

1.a.2: Lower-level model.

- Build a linear interpolation model $q_{i,k}(x_{i,k} + s_i)$ using $Y_{i,k}$ and such that the interpolation conditions (5.10) are satisfied.
- Set $x_{i-1,0} = R_i x_{i,k}$ and $g_i = R_i g_{i,k}$. Set $\ell = 0$. Build $Y_{i-1,\ell} \in \mathbb{R}^{n_{i-1}}$ containing $x_{i-1,0}$ and a model $h_{i-1,\ell}(x_{i-1,0} + s_{i-1})$.

Go to step 1.b.2.

1.b.2: Lower-level criticality test. If $\|g_{i-1,\ell}\| \leq \epsilon_g$, test if $h_{i-1,\ell}$ can be improved (i.e. made *valid*) in some region $\mathcal{Q}_{i-1}(\delta_{i-1})$ for some $\delta_{i-1} > 0$, possibly increasing $|Y_{i-1,\ell}|$, set $\ell = \ell + 1$ and return to Step 1.a.2. If the model cannot be further improved, or if

$$\|x_{i-1,\ell} - x_{i-1,0}\|_{r-1} > (1 - \epsilon_{i-1}^{\Delta})\Delta_{i,k},$$

then return with solution $x_{i-1}^* = x_{i-1,\ell}$ and prolongate the step, obtaining

$$s_{i,k} = P_i(x_{i-1}^* - x_{i-1,0}). \tag{5.14}$$

1.c.2: Step Computation. Compute a step $s_{i-1,\ell}$ that (approximately) minimizes the model $h_{i-1,\ell}(x_{i-1,0} + s_{i-1})$ inside the trust region defined by $\Delta_{i,k}$ and compute $h_{i-1,\ell}(x_{i-1,0} + s_{i-1,\ell})$ and

$$\rho_{i-1,\ell} \stackrel{\text{def}}{=} h_{i-1,\ell}(x_{i-1,0}) - h_{i-1,\ell}(x_{i-1,0} + s_{i-1,\ell}).$$

If $\rho_{i-1,\ell} > \eta_0$, define $x_{i-1}^* = x_{i-1,0} + s_{i-1,\ell}$ and prolongate the step, obtaining $s_{i,k}$ as in (5.14). Go to Step 2.

The step that has been prolongated after being computed in the lower level is treated exactly as if it were a step computed in the upper level; this means that, if the iterate generated by this step is not successful, we will try to include it in the interpolation set $Y_{r,k}$. Apart from this fact and from the use of $x_{i,k}$ and $R_i x_{i,k}$ as the base points for $Y_{i,k}$ and $Y_{i-1,\ell}$, $\ell = 0, \ldots, t$, t being the last iterate before we decide to return to the upper level, the interpolation sets at each level are different and do not contain much information about each other. In other words, we cannot infer geometry properties or any other information about the upper-level interpolation set from the lower-level interpolation set.

Another point that must be clarified is the decision to perform the lower-level step computation. Indeed, the model choice at Step 1.a is a delicate one, and depends on which type of model we choose to use at the lower level. Thus, we will discuss this point further in the next subsections.

Some DFO methods have been proposed which consider the use of incomplete interpolation models (for example, linear models or incomplete quadratic models) or even of models which are not necessarily pure interpolation models, such as the ones discussed by Powell [55] or Alexandrov and Lewis [1]. With this in mind, we propose here different possibilities for the formulation of the lower-level model $h_{i-1,\ell}$, which are, obviously, not the only ones. In our experience, however, these models have been the best in terms of performance and thus we have chosen to focus on them in our implementation.

5.2.1.1 Linear coherent models

One possibility for the computation of the model is to use information from levels $i-1$ and i in order to build a model at level $i-1$, and (approximately) minimize this model to produce a new step, which is then prolongated into level i.

For the method to be effective, we would like the lower-level model to imitate, somehow, the properties of the higher-level model. In this case, we would like to impose first-order coherence, at least around our current iterate $x_{i,k}$. Considering the framework of Alexandrov and Lewis [1], we will modify the model (5.12) at level $i-1$, obtaining

$$h_{i-1}(x_{i-1,0} + s_{i-1}) = f_i(x_{i,k}) + \langle R_i g_i, s_{i-1}\rangle + \frac{1}{2 f_{i-1}(x_{i-1,0})}\langle s_{i-1}, R_i g_i g_{i-1}^T s_{i-1}\rangle - \frac{f_i(x_{i,k})}{(2 f_{i-1}(x_{i-1,0}))^2}\langle s_{i-1}, g_{i-1} g_{i-1}^T s_{i-1}\rangle. \tag{5.15}$$

Although this model can be very different from the interpolation model we would compute at level i, it is then easy to verify that (5.15) satisfies

$$\begin{aligned} h_{i-1}(x_{i-1,0}) &= f_i(x_{i,k}) \\ \nabla h_{i-1}(x_{i-1,0}) &= R_i g_i. \end{aligned}$$

This implies then that, if we define $s_{i,k} = P_i s_{i-1}$, we have that

$$\langle \nabla h_{i-1}(x_{i-1,0}), s_{i-1}\rangle = \langle R_i g_i, s_{i-1}\rangle = \langle g_i, s_{i,k}\rangle,$$

similarly to what we have obtained in the RMTR case (see Section 4.1). We thus minimize the quadratic model h_{i-1} at level $i-1$, until a critical point $x_{i-1,*}$ is reached at this level, or until we hit the upper-level trust-region boundary. The step $s_{i,k} = P_i s_{i-1} = P_i(x_{i-1,*} - x_{i-1,0})$ is then used as the solution to the trust-region subproblem at level i.

When using this model, in order to decide if we should take a lower-level step, we will check if

$$\|R_i g_i\| \geq \kappa_g \|g_i\|, \tag{5.16}$$

where $\kappa_g \in (0,1)$. If this fails, then we choose to do a regular interpolation step at the current level.

5.2.1.2 *Galerkin* model

Another possibility for the formulation of the lowe-level model is to use the Galerkin strategy described in Chapter 4, where we (approximately) minimize the model given by

$$h_{i-1}(x_{i-1,0} + s_{i-1}) = f_{i-1}(x_{i-1,0}) + \langle R_i g_i - g_{i-1}, s_{i-1}\rangle + \tfrac{1}{2}\langle s_{i-1}, H_{i-1} s_{i-1}\rangle,$$

at level $i-1$, where g_i is the linear component of a (linear) interpolation model computed at level i for f_i, and g_{i-1} and H_{i-1} are determined by building a model of f_{i-1} of the type (5.12) or (5.11) in a set of interpolation points Y_{i-1} defined in $\mathbb{R}^{n_{i-1}}$, centered around $x_{i-1,0} = R_i x_{i,k}$.

If we allow for the interpolation set at level i to include more points than just $n_i + 1$, we can also build a (possibly incomplete) quadratic model at level i and use it in the construction of the final lower-level model, defining

$$h_{i-1}(x_{i-1,0} + s_{i-1}) = f_{i-1}(x_{i-1,0}) + \langle R_i g_i - g_{i-1}, s_{i-1}\rangle + \tfrac{1}{2}\langle s_{i-1}, R_i H_i P_i - H_{i-1} s_{i-1}\rangle.$$

This may seem at first a very expensive strategy, but in practice it can be advantageous in the progress of the algorithm.

Here, we will also use the test (5.16) in order to decide whether to use the lower-level model to compute the step at iteration k at level i.

5.2.1.3 *Upper* model

Since in a DFO algorithm, the most computationally expensive part is not the step computation but the construction of the interpolation model, another strategy can be imagined where we still use information from both levels i and $i-1$ to build the model, but where the step is computed *at level i*. In this case, Algorithm 5.2.1 will be slightly modified in order to suit this model definition. We thus replace steps 1.a.1, 1.b.1 and 1.c.1 by Algorithm 5.2.2.

Here, we can see that

$$\begin{aligned} h_{i,k}(x_{i,k}) &= f_i(x_{i,k}) \\ \nabla h_{i,k}(x_{i,k}) &= g_i, \end{aligned}$$

Algorithm 5.2.2: 1.a': Upper-level model.

- Build a linear interpolation model $q_{i,k}(x_{i,k} + s_i)$ using $Y_{i,k}$ and such that the interpolation conditions (5.10) are satisfied for f_i;
- Set $x_{i-1,0} = R_i x_{i,k}$ and $g_i = R_i g_{i,k}$. Define $Y_{i-1} \in \mathbb{R}^{n_{i-1}}$ containing $x_{i-1,0}$ and build a linear interpolation model

$$q_{i-1}(x_{i-1,0} + s_{i-1}) = f_{i-1}(x_{i-1,0}) + \langle g_{i-1}, s_{i-1}\rangle,$$

  at level $i-1$;
- Prolongate g_{i-1}, and define

$$h_{i,k}(x_{i,k} + s_i) = f_i(x_{i,k}) + \langle g_{i,k}, s_i\rangle + \frac{1}{2f_{i-1}(x_{i-1,0})}\langle g_i(P_i g_{i-1})^T, s_i\rangle \quad (5.17)$$

$$-\frac{f_i(x_{i,k})}{2\left(f_{i-1}(x_{i-1,0})\right)^2}\langle s_i, P_i g_{i-1}(P_i g_{i-1})^T s_i\rangle \quad (5.18)$$

Go to step 1.b.2.

and thus, $h_{i,k}$ can be seen as a low-fidelity model for f_i that uses information from the lower-level function f_{i-1} as well.

In this case, it is not clear which condition must be satisfied in order to decide if the model to be used in the trust-region subproblem is the regular interpolation model or $h_{i,k}$. In our implementation, we have chosen to use a simple V-cycle-type iteration, but this is not necessarily the best choice. Further investigations into this condition might be needed in order to ensure that we use model $h_{i,k}$ only when desired, but the numerical experiments shown in the next section show that even a simple V-cycle strategy can be very useful.

5.3 Numerical Experiments

We have tested this algorithm, with all three choices of models presented in the previous chapter in 3 test problems, described in Section A.2 of the Appendix. Problems `Bratu` and `DN` were tested using three one-dimensional discretization levels of 7, 15 and 31 variables, while problem `Surf` was tested in three two-dimensional discretization grids of 3^2, 7^2 and 15^2 variables each.

In these tables, $\#f$ represents the number of function evaluations needed at each level, and $\#b$ represents the number of basis orthogonalizations needed at each level. Note that $\#b$ might not be an integer, since in some iterations the basis is not complete, and in this case, we add the fraction of the base that has been orthogonalized to this counter. Each table shows the results for the three models presented in the previous section, as well as the Mesh Refinement and 1-level (i.e. usual DFO method) versions of the method.

All runs were performed in Matlab v.7.1.0.183 (R14) Service Pack 3 on a Dell Precision workstation, using the parameters

$$\epsilon_g = 10^{-3}, \qquad \theta = 10^{-12}.$$

	#f			#b		
	level 3	level 2	level 1	level 3	level 2	level 1
1-level	2849	-	-	283.42	-	-
MR	2214	580	186	35.46	161.70	199.97
Coherent	1666	705	402	19.96	22.40	106.72
Galerkin	1671	705	1074	22.65	22.40	125.39
Upper	1663	815	650	22.76	143.95	212.86

Table 5.1: Results for the Multilevel DFO method applied to the `Bratu` problem.

Table 5.1 shows the advantages of these alternative models applied to the `Bratu` problem, as the number of function evaluations needed for convergence in the highest level decreases in every case. It is also worth noting that the number of required basis orthogonalizations (i.e. construction of the Newton Fundamental Polynomials) is smaller for the three models, even if modestly so. It is also worth noting that even if more function evaluations are needed for these models in lower levels than what is needed by the Mesh Refinement technique, since the cost of function evaluations at higher levels might be much larger that the cost of function evaluations at lower levels, this is not a problem.

	#f			#b		
	level 3	level 2	level 1	level 3	level 2	level 1
1-level	1426	-	-	138.29	-	-
MR	1345	388	86	79.04	47.22	21.72
Linear	931	1769	1137	123.46	112.35	72.70
Galerkin	1185	622	1205	24.15	26.66	67.17
Upper	1134	585	296	18.76	21.15	42.17

Table 5.2: Results for the Multilevel DFO method applied to the `DN` problem.

In Table 5.2, we show the results obtained with the algorithm for the `DN` problem. We can again see that the method performs better than the one-level and mesh refinement variants.

In Table 5.3, we show the results obtained by the algorithm for the `Surf` problem. In this case, we have used the techniques described in Section 5.1.4 for the treatment of the underlying

	#f			#b		
	level 3	level 2	level 1	level 3	level 2	level 1
1-level	1184.17	-	-	370.86	-	-
MR	342.82	60.19	93.50	108.83	54.43	361.22
Linear	137.34	310.17	382.75	36.67	113.79	445.36
Galerkin	155.68	331.70	738.75	42.27	107.01	548.93
Upper	204.43	152.72	194.25	61.20	74.37	390.53

Table 5.3: Results for the Multilevel DFO method applied to the `Surf` problem.

sparsity of the problem, since it allows us to solve the problem for a much larger number of variables than previous algorithms. Moreover, the number of function evaluations $\#f$ is no longer integer, since we consider the evaluation of one element function to be a fraction of a complete function evaluation. Again, we can see that the method performs well in practice.

These numerical results are very simple and limited, but they illustrate a possible implementation of this new method. In particular, the model choice is rather flexible and allows for the use of any multilevel strategy, such as the V-cycles reminiscent of multigrid (presented in Chapter 2) or free recursion. One could also imagine other criteria for the use of the lower-level model depending on the characteristics of each problem.

5.4 Conclusions

In this chapter, we have presented a new formulation of the DFO algorithm to be used for problems that possess a multilevel structure. This could allow us to apply a DFO method to these problems, even when the number of variables is large. However, since even the multilevel version of the DFO method shares some of the limitations of this method, due to time constraints is was not possible to test this algorithm in larger problems. More tests are certainly in our perspectives for the future, as is a more efficient implementation of the algorithm.

Conclusions and further research perspectives

In this thesis, we have presented three new developments in nonlinear optimization which are based in the multigrid philosophy. These methods all have the common property of aiming to solve problems that can be described in several levels of accuracy, which is very common while trying to solve optimization problems resulting from the discretization of Partial Differential Equations in a grid. These problems are extremely common in practice, and thus there is a wide range of applications where these methods can be used.

In Chapter 1, we have presented a brief introduction to nonlinear optimization, including a brief review of the main theoretical results for trust-region methods. In Chapter 2, we have presented the basic ideas behind the multigrid method for linear systems of equations. These two chapters have been the main motivation for our work, described in the remaining three chapters.

A Multilevel Algorithm for the Solution of the Trust-Region Subproblem

In Chapter 3, our concern was to solve the trust-region subproblem. We have reviewed two classical methods for the solution of this problem, the Moré-Sorensen method, which solves the trust-region subproblem exactly, and the Truncated Conjugate Gradient method, which solves this problem approximately.

We have then presented a new multilevel strategy for the exact solution of the Euclidean-norm problem, called the Multilevel Moré-Sorensen method. This method is based on the fact that we can interpret the Euclidean-norm trust-region subproblem as a linear system with one parameter (the Lagrange multiplier for the trust-region constraint), and this problem can thus be solved by the use of multigrid techniques when several descriptions of the problem are available. We have shown some results that indicate that this method can be applied to solve the subproblem exactly when the number of variables is very large, in which case the classical Moré-Sorensen technique cannot be applied since the Cholesky factorization of the system matrix cannot be easily computed.

Recursive Multilevel Trust-Region Methods

In Chapter 4, we have presented a brief review of the Recursive Multilevel Trust Region class of methods. These methods have been first introduced by Gratton et al. [25] for uncon-

strained problems, and they combine the trust-region and multigrid philosophies in a fundamental way, such that the use of coarser descriptions of the problem is effective and computationally cheap.

Here, we have presented a ℓ_∞-norm version of the RMTR method, which is also suitable for the solution of bound-constrained problems, and a complete first-order convergence theory that shows that this new class enjoys the same desirable properties as classical trust-region methods, while being able to solve large-scale problems quickly and with a moderate computational cost.

Multilevel Derivative-Free Optimization

Finally, in Chapter 5, we have described a new multilevel strategy for derivative-free optimization problems. Classical derivative-free optimization methods based on the trust-region approach are very efficient, and they have been our motivation for this multilevel extension. We present the basic idea of this algorithm, which is to make use of lower-level descriptions of the problem to build new models around the current iterate, such that the computational cost and number of function evaluations needed for the construction of the model is smaller than that of a full quadratic model, while still being a good approximation for the objective function. We show some results that indicate that this method works and that it is possible to apply multilevel techniques to these problems.

Future Perspectives

The multilevel trust-region methods presented here have shown the possibilities of the application of the multilevel philosophy to nonlinear optimization. Moreover, they are somewhat general, which allows for extensions to different problems in the future. Similarly, with the exception of the Recursive Multilevel Trust-Region method in infinity-norm, the other methods could certainly benefit from a more efficient computational implementation.

The results shown here are only a part of this new field of study, and other works have also focused on this subject. In particular, the theses of Mouffe [46] and Tomanos [65] share a lot of the work done here through collaborations. Furthermore, trust-region methods are certainly not the only application of multigrid techniques to optimization, and although we have focused on this particular subject, it is certainly not the only possibility for large-scale problems.

Finally, these methods are all very recent, but the results obtained so far show that there is still a lot of room for new developments and applications. Possible extensions are the application of Algebraic Multigrid techniques to optimization, as well as the introduction of more complex constraints and the generalization to higher-order models for the trust-region subproblem.

Summary of contributions

Our contributions include new developments in multilevel methods for optimization. We have developed the convergence theory for the infinity-norm version of the Recursive Multilevel Trust-Region method, as well as two new methods for the exact solution of the trust-region subproblem and for the solution of derivative-free optimization problems, both using multilevel techniques. Here, we summarize our contributions.

- The Recursive Multilevel Trust-Region (RMTR) method in infinity norm and its first-order convergence theory have been published in Gratton et al. [24]. This version of the RMTR method is capable of treating bound-constrained problems, and presents excellent numerical results.
- The Multilevel Moré-Sorensen method is obtained by modifying the classical Moré-Sorensen method by considering the trust-region subproblem as a linear system with one parameter. This new method, along with numerical results, has been published in Toint et al. [64].
- The Multilevel Derivative-Free Trust-Region algorithm consists in the use of the underlying multilevel structure of the problem to generate different quadratic models for the trust-region method which are not purely interpolation models. It is presented in this thesis, along with numerical results.

Main notations and abbreviations

General

$\mathbb{R}$	set of real numbers
$\mathbb{R}^n$	real n-dimensional Euclidean space
$e^{[i]}$	i-th coordinate vector
$\|\cdot\|$	absolute value of a scalar
$\|\|\cdot\|\|$	vector norm
$\|\mathcal{S}\|$	cardinality of the set $\mathcal{S}$
$\mathcal{D}$	domain of a function
$\nabla f(x)$	gradient of f
$\nabla^2 f(x)$	Hessian matrix of f
$\text{Proj}(\cdot)$	projection operator
$\rho(A)$	spectral radius of the matrix A
$\lambda_{\min}(A)$	smallest eigenvalue of matrix A
$\lambda_{\max}(A)$	largest eigenvalue of matrix A
$\kappa_p(A)$	p-norm condition number of matrix A
$\mathcal{E}$	set of equality constraints
$\mathcal{I}$	set of inequality constraints
x^*	optimal solution
$\mathcal{F}$	feasible region
$\mathcal{N}$	neighbourhood
$\mathcal{A}(x)$	active set at x
$\mathcal{L}(x, \lambda)$	Lagrangian function

Multigrid Methods

Ω_h	discretized domain with subinterval length h
e^ℓ	error at the current iteration
u^ℓ	approximation to the solution at the current iteration
u_*	exact solution
r^ℓ	residual at the current iteration
P_i	Prolongation operator from level $i-1$ to level i
R_i	Restriction operator from level i to level $i-1$

Algorithms

x_k	kth iterate (vector)
$\mathcal{B}_k$	trust region at iteration k
Δ_k	trust-region radius at iteration k
g_k	gradient of the objective function at x_k
H_k	symmetric approximation to the objective Hessian at x_k
s_k	step at iteration k
x_k^C	(generalized) Cauchy point
$m_k(\cdot)$	model of f at the kth iteration
ρ_k	ratio of actual to predicted decrease

MMS

$\phi(\lambda)$	secular equation at lagrange multiplier λ
λ^L	lower bound for the interval of uncertainty in the Moré-Sorensen algorithm
λ^U	upper bound for the interval of uncertainty in the Moré-Sorensen algorithm
s^M	exact solution to the ℓ_2-norm trust-region subproblem
λ^M	Lagrange multiplier corresponding to the exact solution to the ℓ_2-norm trust-region subproblem
ϵ^Δ	tolerance on the trust-region in the Moré-Sorensen algorithm
$v_{i,k}$	vector at level i, iteration k
M_i	product of restriction operators from level $i+1$ up to level p
Q_i	product of prolongation operators from level $i+1$ up to level p
ϵ^r	tolerance on the residual

RMTR

$x_{i,k}$	iterate at level i, iteration k
$x_{i-1,*}$	solution at level $i-1$
$g_{i,k}$	gradient of the model at level i, iteration k
κ_{g}	bound on the ratio between $\|g_{i,k}\|$ and $\|R_i g_{i,k}\|$
$\Delta_{i,k}$	trust-region radius at level i, iteration k
ϵ_i^Δ	tolerance on the i-th level trust region
ϵ_i^{g}	tolerance on the i-th level gradient
$\chi_{i,k}$	criticality measure at level i, computed at $x_{i,k}$
κ_χ	bound on the ratio between $\chi_{i,0}$ and $\chi_{\pi(i,0)}$
ϵ_i	tolerance on the i-th level criticality measure $\chi_{i,\cdot}$
$\pi(i,k)$	predecessor of iteration k at level i (that is, iteration (i,k) happens inside iteration $\pi(i,k)$)

RMTR

$\mathcal{F}_i$	feasible set at level i
$\mathcal{B}_{i,k}$	"pure" trust region at level i, iteration k
$\mathcal{A}_i$	restricted $(i+1)$-st level trust-region at level i
$\mathcal{W}_{i,k}$	intersection between i-th level trust region at iteration k, restricted $(i+1)$-st level trust region and i-th level feasible set
$\mathcal{L}_i$	intersection between i-th level feasible set and restricted $(i+1)$-st level trust region
$\mathcal{S}_{i,k}$	intersection between i-th level trust region at iteration k and restricted $(i+1)$-st level trust region
$\mathcal{R}(i,k)$	set of all iterations whose predecessor is (i,k)
$\mathcal{T}(i,k)$	set of Taylor iterations inside $\mathcal{R}(i,k)$

Main mathematical notations

BTR	Basic Trust Region
CG	Conjugate Gradient
DFO	Derivative-Free Optimization
KKT	Karush-Kuhn-Tucker
LICQ	Linear Independence Constraint Qualfication
MFCQ	Mangasarian-Fromovitz Constraint Qualfication
RMTR	Recursive Multilevel Trust-Region method
TCG	Truncated Conjugate Gradient method
PTCG	Projected Truncated Conjugate Gradient method
MS	Moré-Sorensen method
MMS	Multilevel Moré-Sorensen method
RQMG	Rayleigh Quotient Minimization Multigrid algorithm
FMG	Full Multigrid
MR	Mesh Refinement
SCM	Sequential Coordinate Minimization

Main abbreviations

`3D-1`	A 3D quadratic problem
`3D-2`	A 3D nonlinear problem
`C-D`	A 3D convection diffusion problem
`3D-BV`	A 3D version of the Moré boundary value problem
`Bratu`	A 1D version of the Bratu problem
`Surf`	A minimal surface problem in 2D
`DN`	A 1D Dirichlet-to-Neumann transfer problem

Multilevel Problems

Appendix

Appendix A

Test Problems Descriptions

A.1 Problems for the Multilevel Moré-Sorensen Method

A.1.1 3D Quadratic Problem 1 (3D-1):

A convex quadratic problem, where we consider the three-dimensional boundary value problem defined by

$$\begin{aligned} -\Delta u(x,y,z) &= f \quad \text{in } S_3 \\ u(x,y,z) &= 0 \quad \text{on } \partial S_3, \end{aligned}$$

where f is chosen so that the analytical solution to this problem is $u(x,y,z) = 8$. In a multilevel formulation, this gives linear systems $A_i x = b_i$ at level i where each A_i is a symmetric positive-definite matrix. This problem is the typical *model problem* for multigrid solvers. Here, we want to find the solution to its variational formulation

$$\min_{x \in \mathbb{R}^{n_p}} \frac{1}{2} x^T A_p x - x^T b_p.$$

A.1.2 3D Nonlinear Problem 2 (3D-2):

Another convex quadratic problem, where we consider the differential equation

$$\begin{aligned} -(1+\sin(3\pi x)^2)\Delta u(x,y,z) &= f \quad \text{in } S_3 \\ u(x,y,z) &= 0 \quad \text{on } \partial S_3, \end{aligned}$$

where f is chosen so that the analytical solution to this problem is

$$u(x,y,z) = x(1-x)y(1-y)z(1-z).$$

In a multilevel formulation, this gives linear systems $A_i x = b_i$ at level i where each A_i is a symmetric positive-definite matrix. This problem is considered in its variational formulation

$$\min_{x \in \mathbb{R}^{n_p}} \frac{1}{2} x^T A_p x - x^T b_p.$$

A.1.3 Convection-Diffusion problem (C-D):

We wan to minimize the variational formulation of the following nonlinear partial differential equation

$$\Delta u - Ru\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} + \frac{\partial u}{\partial z}\right) + f(x,y,z) = 0, \qquad R = 20,$$

where $f(x,y,z) = 2000x(1-x)y(1-y)z(1-z)$, over S_3 with Dirichlet boundary conditions $u = 0$ on ∂S_3.

A.1.4 Boundary Value Problem (3D-BV):

This is a problem inspired by the one dimensional two-point boundary value problem presented in Moré et al. [45] and is defined by

$$-\Delta u(s,t,z) = \tfrac{1}{2}(u(s,t,z) + t + s + z + 1)^3,$$

with

$$\begin{aligned} u(0,t,z) = u(1,t,z) = 0, \quad 0 < t < 1, \\ u(s,0,z) = u(s,1,z) = 0, \quad 0 < s < 1, \\ u(s,t,0) = u(s,t,1) = 0, \quad 0 < z < 1. \end{aligned}$$

Here, we look for the solution of the least squares problem

$$\min_{s,t,z\in[0,1]} \|-\Delta u(s,t,z) - \tfrac{1}{2}(u(s,t,z) + t + s + z + 1)^3\|_2^2.$$

A.2 Problems for the Multilevel Derivative-Free Optimization Method

A.2.1 Bratu - `Bratu`

Here, we are looking for the solution of the 1-dimensional boundary value problem defined in $[0,1]$ by

$$\begin{cases} u'' + R\exp u = 0, \\ u(0) = u(1) = 0. \end{cases}$$

In our case, we will use a finite-element approximation to the least-squares formulation for this problem, namely

$$\begin{cases} \min \displaystyle\int_0^1 \|u(t)'' + R\exp u(t)\|^2 dt, \\ \text{s/t} \quad u(0) = u(1) = 0, \end{cases}$$

where $R = 3.51$ (as suggested in [33]).

A.2.2 Minimal Surface - `Surf`

We wish to find the solution of the minimum surface problem given by

$$\min_{v\in\mathcal{K}} \int_0^1 \int_0^1 (1 + (\partial_x v)^2 + (\partial_y v)^2)^{\frac{1}{2}} dx\, dy,$$

where $\mathcal{K} = \{v \in H^1(S_2) \mid v(x,y) = v_0(x,y) \text{ on } \partial S_2\}$. The boundary condition v_0 is chosen as

$$v_0(x,y) = \begin{cases} f(x), & \text{for} \quad y = 0, \quad 0 \le x \le 1, \\ 0, & \text{for} \quad x = 0, \quad 0 \le y \le 1, \\ f(x), & \text{for} \quad y = 1, \quad 0 \le x \le 1, \\ 0, & \text{for} \quad x = 1, \quad 0 \le y \le 1. \end{cases}$$

where $f(x) = x(1-x)$. To do this, we discretize the problem using a finite element basis, defined by a uniform triangulation of S_2, with same grid spacing h along the 2 coordinate directions. We use the classical P1 functions which are linear on each triangle and take value 0 or 1 at each vertex as basis functions.

A.2.3 Dirichlet-to-Neumann Transfer Problem - `DN`

This problem is taken from Lewis and Nash [41] and is described as follows. Let $\Omega = \{(x,y) \mid 0 \le x \le \pi, 0 \le y \le 1\}$, and $\Gamma = \{(x,0) \mid 0 \le x \le \pi\}$ the lower boundary of Ω. We want to find $a(x)$ that minimizes

$$F(a) = \frac{1}{2} \int_0^\pi \left(\frac{\partial u}{\partial y}(x,0) - \phi(x) \right)^2 dx,$$

where $\phi = \sum_{i=1}^{15} \sin ix + \sin 40x$, and where u is the solution to a finite-differences approximation to the boundary value problem

$$\begin{cases} \Delta u(x,y) &= 0 & \text{in } \Omega \\ u(x,y) &= 0 & \text{in } \partial\Omega \setminus \Gamma \\ u(x,0) &= a(x). & \end{cases}$$

Bibliography

[1] N. M. Alexandrov and R. L. Lewis. An overview of first-order model management for engineering optimization. *Optimization and Engineering*, 2:413–430, 2001.

[2] A. Brandt. Multi-level adaptative solutions to boundary value problems. *Mathematics of Computation*, 31(138):333–390, 1977.

[3] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. SIAM, Philadelphia, USA, 2nd edition, 2000.

[4] T. F. Coleman and J. J. Moré. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM Journal on Numerical Analysis*, 20:187–209, 1983.

[5] B. Colson and Ph. L. Toint. A derivative-free algorithm for sparse unconstrained optimization problems. In A. H. Siddiqi and M. Kočvara, editors, *Trends in Industrial and Applied Mathematics*, pages 131–149, Dordrecht, The Netherlands, 2002. Kluwer Academic Publishers.

[6] B. Colson and Ph. L. Toint. Optimizing partially separable functions without derivatives. *Optimization Methods and Software*, 20(4-5):493–508, 2005.

[7] A. R. Conn and Ph. L. Toint. An algorithm using quadratic interpolation for unconstrained derivative free optimization. In G. Di Pillo and F. Gianessi, editors, *Nonlinear Optimization and Applications*, pages 27–47, New York, 1996. Plenum Publishing. Also available as Report 95/6, Dept of Mathematics, FUNDP, Namur, Belgium.

[8] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis*, 25(182):433–460, 1988. See also same journal 26:764–767, 1989.

[9] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. LANCELOT*: a Fortran package for large-scale nonlinear optimization (Release A)*. Number 17 in Springer Series in Computational Mathematics. Springer Verlag, Heidelberg, Berlin, New York, 1992.

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

[11] A. R. Conn, K. Scheinberg, and Ph. L. Toint. A derivative free optimization algorithm in practice. Technical Report TR98/11, Department of Mathematics, University of Namur, Namur, Belgium, 1998.

[12] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-Region Methods*. Number 01 in MPS-SIAM Series on Optimization. SIAM, Philadelphia, USA, 2000.

[13] A. Curtis, M. J. D. Powell, and J. Reid. On the estimation of sparse Jacobian matrices. *Journal of the Institute of Mathematics and its Applications*, 13:117–119, 1974.

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

[15] S. Dollar, N. Gould, and D. Robinson. On solving trust-region and other regularised subproblems in optimization. Technical Report RAL-TR-2009-003, Rutherford Appleton Laboratories, Oxfordshire OX11 0QX, England, February 2009.

[16] R. P. Fedorenko. A relaxation method for solving elliptic difference equations. *USSR Comp. Math. Math. Phys.*, 1(5):1092–1096, 1962.

[17] M. Fisher. Minimization algorithms for variational data assimilation. In *Recent Developments in Numerical Methods for Atmospheric Modelling*, pages 364–385. ECMWF, 1998.

[18] E. Gelman and J. Mandel. On multilevel iterative methods for optimization problems. *Mathematical Programming*, 48(1):1–17, 1990.

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

[20] S. M. Goldfeldt, R. E. Quandt, and H. F. Trotter. Maximization by quadratic hill-climbing. *Econometrica*, 34:541–551, 1966.

[21] G. H. Golub and C. F. Van Loan. *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)*. The Johns Hopkins University Press, 1996.

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

[23] S. Gratton, A. Sartenaer, and Ph. L. Toint. Second-order convergence properties of trust-region methods using incomplete curvature information, with an application to multigrid optimization. *Journal of Computational and Applied Mathematics*, 24(6):676–692, 2006.

[24] S. Gratton, Mélodie Mouffe, Ph. L. Toint, and M. Weber Mendonça. A recursive ℓ_∞-trust-region method for bound-constrained nonlinear optimization. *IMA Journal of Numerical Analysis*, 28(4):827–861, 2008. doi: 10.1093/imanum/drn034. URL `http://imajna.oxfordjournals.org/cgi/content/abstract/28/4/827`.

[25] S. Gratton, A. Sartenaer, and Ph. L. Toint. Recursive trust-region methods for multiscale nonlinear optimization. *SIAM Journal on Optimization*, 19(1):414–444, 2008. ISSN 1052-6234.

[26] S. Gratton, M. Mouffe, A. Sartenaer, Ph. L. Toint, and D. Tomanos. Numerical experience with a recursive trust-region method for multilevel nonlinear optimization. *Optimization Methods and Software*, to appear, 2009.

[27] A. Griewank. Computational differentiation and optimization. In J. R. Birge and K. G. Murty, editors, *Mathematical Programming: State of the Art 1994*, pages 102–131, Ann Arbor, USA, 1994. The University of Michigan.

[28] A. Griewank and G. Corliss. *Automatic Differentiation of Algorithms: Theory, Implementation and Application*. SIAM, Philadelphia, USA, 1991.

[29] A. Griewank and Ph. L. Toint. On the unconstrained optimization of partially separable functions. In M. J. D. Powell, editor, *Nonlinear Optimization 1981*, pages 301–312, London, 1982. Academic Press.

[30] A. Griewank and Ph. L. Toint. Partitioned variable metric updates for large structured optimization problems. *Numerische Mathematik*, 39:119–137, 1982.

[31] M. D. Hebden. An algorithm for minimization using exact second derivatives. Technical Report T.P. 515, AERE Harwell Laboratory, Harwell, Oxfordshire, England, 1973.

[32] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of the National Bureau of Standards*, 49:409–436, 1952.

[33] Jon Jacobsen and Klaus Schmitt. The liouville-bratu-gelfand problem for radial operators. *Journal of Differential Equations*, 184(1):283–298, 2002. ISSN 0022-0396. doi: DOI: 10.1006/jdeq.2001.4151.

[34] W. Karush. Minima of functions of several variables with inequalities as side conditions. Master's thesis, Department of Mathematics, University of Chicago, Illinois, USA, 1939.

[35] R. Kornhuber. Monotone multigrid methods for elliptic variational inequalities I. *I. Numer. Math*, 69:18–4, 1994.

[36] R. Kornhuber. Monotone multigrid methods for elliptic variational inequalities ii. *Numerische Mathematik*, 72(4):481–499, 1996. ISSN 0945-3245.

[37] R. Kornhuber. Adaptive monotone multigrid methods for some non-smooth optimization problems. In R. Glowinski, J. Périaux, Z. Shi, and O. Widlund, editors, *Domain Decomposition Methods in Sciences and Engineering*, pages 177–191. Wiley, 1997.

[38] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In J. Neyman, editor, *Proceedings of the second Berkeley symposium on mathematical statistics and probability*, pages 481–492, California, USA, 1951. University of Berkeley Press.

[39] K. Levenberg. A method for the solution of certain problems in least squares. *Quarterly Journal on Applied Mathematics*, 2:164–168, 1944.

[40] M. Lewis and S. G. Nash. Practical aspects of multiscale optimization methods for VLSICAD. In Jason Cong and Joseph R. Shinnerl, editors, *Multiscale Optimization and VLSI/CAD*, pages 265–291, Dordrecht, The Netherlands, 2002. Kluwer Academic Publishers.

[41] M. Lewis and S. G. Nash. Model problems for the multigrid optimization of systems governed by differential equations. *SIAM Journal on Scientific Computing*, 26(6):1811–1837, 2005.

[42] J. Mandel and S. McCormick. A multilevel variational method for $Au = \lambda Bu$ on composite grids. *Journal of Computational Physics*, 80(2):442–452, 1989.

[43] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11:431–441, 1963.

[44] J. J. Moré and D. C. Sorensen. On the use of directions of negative curvature in a modified Newton method. *Mathematical Programming*, 16(1):1–20, 1979.

[45] J. J. Moré, B. S. Garbow, and K. E. Hillstrom. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, 7(1):17–41, 1981.

[46] Mélodie Mouffe. *Multilevel Optimization in infinity norm and associated stopping criteria*. PhD in Mathematics, CERFACS, Toulouse, France, 2009.

[47] S. G. Nash. A multigrid approach to discretized optimization problems. *Optimization Methods and Software*, 14:99–116, 2000.

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

[49] Yu. Nesterov. *Introductory Lectures on Convex Optimization*. Applied Optimization. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2004.

[50] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer series in operations research. Springer Verlag, Heidelberg, Berlin, New York, second edition, 2006. ISBN 0-387-30303-0 (hardback).

[51] S. Oh, A. Milstein, Ch. Bouman, and K. Webb. A general framework for nonlinear multigrid inversion. *IEEE Transactions on Image Processing*, 14(1):125–140, 2005.

[52] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, London, 1970.

[53] M. J. D. Powell. On the Lagrange functions of quadratic models defined by interpolation. Technical Report NA10, Department of Applied Mathematics and Theoretical Physics, Cambridge University, Cambridge, England, 2000.

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

[55] M. J. D. Powell. Least Frobenius norm updating of quadratic models that satisfy interpolation conditions. Technical Report NA2002/08, Department of Applied Mathematics and Theoretical Physics, Cambridge University, Cambridge, England, 2002.

[56] M. J. D. Powell. A new algorithm for unconstrained optimization. In J. B. Rosen, O. L. Mangasarian, and K. Ritter, editors, *Nonlinear Programming*, pages 31–65, London, 1970. Academic Press.

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

[58] M. J. D. Powell. A direct search optimization method that models the objective by quadratic interpolation. Presentation at the 5th Stockholm Optimization Days, Stockholm, 1994.

[59] M. J. D. Powell. Trust region methods that employ quadratic interpolation to the objective function. Presentation at the 5th SIAM Conference on Optimization, Victoria, 1996.

[60] M. J. D. Powell and Ph. L. Toint. On the estimation of sparse Hessian matrices. *SIAM Journal on Numerical Analysis*, 16(6):1060–1074, 1979.

[61] Th. Sauer and Y. Xu. On multivariate Lagrange interpolation. *Mathematics of Computation*, 64:1147–1170, 1995.

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

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

[64] Ph. L. Toint, D. Tomanos, and M. Weber-Mendonca. A multilevel algorithm for solving the trust-region subproblem. *Optimization Methods and Software*, 24(2):299–311, 2009. ISSN 1055-6788. doi: http://dx.doi.org/10.1080/10556780802571467.

[65] Dimitri Tomanos. *Algorithms and Softwares for Multilevel Nonlinear Optimization*. PhD in Mathematics, Facultés Universitaires Notre-Dame de la Paix, Namur, Belgium, 2009.

[66] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25, 1997.

[67] U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. Elsevier, Amsterdam, The Netherlands, 2001.

[68] P. Wesseling. *An introduction to Multigrid Methods*. J. Wiley and Sons, Chichester, England, 1992. Corrected Reprint, Edwards, Philadelphia, 2004.