SAPIENZA
UNIVERSITÀ DI ROMA

New gradient methods: spectral properties and steplength selection

Scuola di dottorato in Ricerca Operativa

Dottorato di Ricerca in Ricerca Operativa – XXVI Ciclo

Candidate

Roberta De Asmundis
ID number 1385637

Thesis Advisor

Prof. Gerardo Toraldo

A thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Operational Research

April 26 2014

Thesis defended on May 16 2014
in front of a Board of Examiners composed by:

Prof. Luca Zanni (chairman)

Prof. Andrea Lodi

Dr. Maria Grazia Scutellà

New gradient methods: spectral properties and steplength selection
Ph.D. thesis. Sapienza – University of Rome

© 2013 Roberta De Asmundis. All rights reserved

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: roberta.deasmundis@uniroma1.it

Dedicated to
my "robotic" husband, my beloved parents,
my grandma who is turning 89 thinking she is 62, and my purring cats.

Acknowledgments

Then said a teacher, "Speak to us of Teaching."

And he said:

No man can reveal to you aught but that which already lies half asleep in the dawning of our knowledge.
The teacher who walks in the shadow of the temple, among his followers, gives not of his wisdom but rather of his faith and his lovingness.
If he is indeed wise he does not bid you enter the house of wisdom, but rather leads you to the threshold of your own mind.
The astronomer may speak to you of his understanding of space, but he cannot give you his understanding.
The musician may sing to you of the rhythm which is in all space, but he cannot give you the ear which arrests the rhythm nor the voice that echoes it.
And he who is versed in the science of numbers can tell of the regions of weight and measure, but he cannot conduct you thither.
For the vision of one man lends not its wings to another man.
And even as each one of you stands alone in God's knowledge, so must each one of you be alone in his knowledge of God and in his understanding of the earth.

The Prophet, Kahlil Gibran

I would like to express my whole gratitude to my supervisor, Prof. Gerardo Toraldo, who supported my work from the beginning. His competence, patience and kindness guided me during these years, always caring about me, as a wise teacher. I have to thank him for having introduced me in the world of the nonlinear optimization and numerical analysis, giving me the opportunity to learn new subjects and new methodologies to study, plan, analyse, test, and discuss about my life. During the first year of doctorate he decided to send me to Siena to attend a course on convex optimization, where I met a shy boy, I studied, I analysed, I tested him, and I decided to marry him two years later, changing my life. On the second year of my doctorate, prof. Toraldo told me to go to the University of Florida for a semester, it was a grate life experience, I had to face solitude and the distance from my beloved and my family, a big experiment in which I succeeded, coming back stronger and *happy. I also got to know a beautiful country, a different culture, and I shared my time with some wonderful friends that I will never forget.*

Prof. Toraldo has the merit of having introduced me to special and extraordinary women, above all Prof. Daniela di Serafino, without whom my work would have been enormously more difficult. With her energetic character, her professional behaviour, and her competence, she has been a guide, a lighthouse pointing on how I should work, what I should do, how I should react, how I should look at my future, and I will be forever grateful for all the advices and suggestions she gave me during these three years. Other people were part of my PhD journey, all of them excellent experts, and good fiends, who made this period more enjoyable and interesting.

I also have to thank Prof. William W. Hager, who supervised me during my period at the University of Florida, and Prof. Hongchao Zhang. I will always remember their calm suggestions, their careful considerations, always at the point, during our chats in their room in the hot summertime Floridian afternoons. Their confidence in me is a kind note in my graduate studies.

Last, but not least, I would like to thank my family for letting me make my own choices, always supporting me, encouraging me to move forward, reaching new targets, and my husband Donato, who always stands proud by my side.

Abstract

In the last 25 years the interest in gradient methods has been renewed after the publication of a pioneering paper by Barzilai and Borwein, where a novel strategy for choosing the steplength was proposed. This opened the way to other steplength selection rules, and first order methods for continuous nonlinear optimization returned on the scene, becoming a valid and useful tool for applications. As a matter of fact, gradient methods can be successfully applied to certain ill-posed inverse problems, arising, e.g., in image processing; furthermore, the low computational cost of gradient methods makes them suitable for other large-scale problems, e.g., in machine learning and data mining.

It has become increasingly clear that the critical issue in gradient methods is the choice of the steplength, whereas using the gradient as search direction may lead to effective algorithms. In particular, the Cauchy steplength, henceforth denoted by α_k^{SD}, generally produces very slow convergence, but, on the other hand, it can provide spectral information on the Hessian matrix of the problem, which can be put somehow in an algorithmic framework to construct faster gradient methods.

This work mainly focuses on gradient methods for the unconstrained minimization problem

$$\min_{x \in R^n} \frac{1}{2} x^T A x - b^T x,$$

where $A \in R^{n \times n}$ is symmetric positive-definite and $b \in R^n$. This simple setting not only allows the study of the relevance of the eigenvalues of the Hessian of the objective function to the considered methods, highlighting the effects of a particular steplength instead of another, but it is also of practical importance, since it arises in many applications, e.g., in image processing and compressed sensing. Furthermore, it provides the basis for the development of gradient methods for constrained problems and it paves the way for the the minimization of general non-quadratic functions.

This thesis, starting from the classical Steepest Descent (SD) method by Cauchy, and going through the most recent and competitive gradient methods, investigates how the spectral properties associated with the SD method, if suitably used to build some special steplengths, can influence the performance of the method, dramatically improving its poor original behavior. In this framework, two new methods are presented and analysed, the Steepest Descent method with Alignment (SDA) and the Steepest Descent method with Constant steplength (SDC). Both of them share the idea of fostering a selective elimination of the components of the gradient along the eigenvectors of the Hessian matrix A, thus pushing the search in subspaces of smaller dimensions, and speeding up the convergence of the method. This selec-

tive elimination is based on new steplength selection rules, which alternate, in a cyclic way, sequences of Cauchy steplengths with sequences of constant steplengths containing spectral information on the Hessian.

More in detail, the SDA method alternates a sequence of Cauchy steps with a sequence of constant steps built from the formula

$$\alpha_k^{SDA} = \left(\frac{1}{\alpha_{k-1}^{SD}} + \frac{1}{\alpha_k^{SD}} \right)^{-1},$$

where k refers to the last computed Cauchy step. Under non-restrictive assumptions, it has been proved that the SDA method tends to align the search direction with the eigendirection of A corresponding to the minimum eigenvalue, forcing the algorithm search into a one-dimensional subspace, and hence escaping from the zigzagging behaviour of the classical SD method [26]. Similarly, the SDC methods exploits the Yuan steplength formula [66]

$$\alpha_k^{Y} = 2 \left(\sqrt{ \left(\frac{1}{\alpha_{k-1}^{SD}} - \frac{1}{\alpha_k^{SD}} \right)^2 + 4 \frac{\|g_k\|^2}{\left(\alpha_{k-1}^{SD} \|g_{k-1}\| \right)^2} } + \frac{1}{\alpha_{k-1}^{SD}} + \frac{1}{\alpha_k^{SD}} \right)^{-1},$$

in a cyclic framework, in which a sequence of SD iterates is alternated with a sequence of steps using constant steplengths computed by using that formula, where k refers to the last SD iterate. By studying the asymptotic behaviour of the Yuan steplength, it has been shown that the SDC method has the ability of pushing toward zero the eigencomponents of the gradient in a selective way, proceeding from the component associated with the largest eigenvalue to that associated with the smallest one [24].

Extensive numerical experiments have been performed with the SDA and the SDC methods. They show that both methods are highly competitive with the fastest methods among the gradient ones; in particular, the SDC method clearly appears superior as the Hessian condition number and the accuracy requirement increase.

We note that the behaviour of the two proposed methods is different from that of other efficient gradient methods, where the ability to reduce all the eigencomponents "at the same rate" has been experimentally observed and considered as one of the main reasons for their effectiveness. Furthermore, the "elimination path" followed by the SDC method produces a regularizing effect on certain ill-posed inverse problems [25]. Therefore, in the final part of this dissertation, the application of the SDC method to some ill-posed, inverse problems has been analysed. In particular, it has been observed that the SDC method tends to reconstruct the solution of the selected ill-posed problems giving preference to the “significant” components of the solution, thus showing an implicit filtering effect and allowing to obtain a good reconstructed image with reasonable computational efficiency.

Future work will address the development of projection techniques to be combined with the proposed methods, to solve constrained problems, and the development of strategies aimed at extending the proposed methods to the more general non-quadratic framework. This will be useful, e.g., in image reconstruction, where non-negativity constraints are often considered, and where the functional to minimize is often non-quadratic.

Contents

List of Figures

List of Tables

Chapter 1

From the steepest descent to some faster gradient methods

In this chapter we place the foundation for the development of the subsequent theoretical analysis on gradient methods and step size selection techniques. More in detail, we will consider some among the most popular gradient methods to solve the following quadratic unconstrained minimization problem:

$$\min_{x \in R^n} f(x),$$

where f is a twice continuously differentiable function.
Of course, the state of the art in gradient methods is much wider than what is reported in these pages. Currently, the literature is better focused in the development of methodologies tailored to the applications for which they are intended. In several fields, gradient methods appear to be attractive because of their extreme implementative simplicity and their low memory requirements, and, in addition to these motivations, if required by the the particular problem under consideration, a projection into a feasible region is relatively an inexpensive operation if the constraints are simple. It is also well known that gradient methods may exhibit a very slow convergence if a *good* stepsize selection technique is not applied. In some applications, such as the image deblurring, or the training of support vector machines, the presence of big data has in fact pushed the research of easier and faster gradient-based methods (see, e.g., [5, 12, 62] and the references therein). Unfortunately, the lack of a strong underlying theory has brought researches to use faster gradient method without having the awareness of why this or that method was going better. For this reason we believe in the need of this work, mostly aimed in finding the theoretical motivation for which we can build a fast gradient method using a simple and efficient step size selection rule, making it suitable for the application at hand.

This chapter is organized as follows, we present the classical steepest descent (SD) method and we go through its theoretical properties in Section 1.1; we introduce the Barziali and Borwein method in Section 1.2; then Section 1.3 focuses on some attempts to produce faster gradient methods, escaping from the zigzag path made by the iterates in the SD method. Finally, in Section 1.4 we present one among the most competitive faster gradient method which will be used as main benchmark in the following development of this thesis.

1.1 The steepest descent method

Gradient method was one of the first optimization methods proposed to solve the unconstrained minimization problem,

$$\min_{x \in R^n} f(x),$$

where f is a twice continuously differentiable function whose gradient $\nabla f(x_k)$ in the point x_k will be denoted by g_k. The method is based on the use of the *antigradient* of the function f at the point x_k as search direction, $d_k = -g_k$ which, if normalized according to the euclidean norm, minimizes the directional derivative of f at x_k among all the normalized descent directions, therefore the method is often called the Steepest Descent (SD) method.
A general gradient method is an iterative method which generates a sequence of iterates $\{x_k\}$ by the following rule:

$$x_{k+1} = x_k - \alpha_k g_k,$$

where the steplength α_k is a nonnegative scalar minimizing $f(x_k - \alpha g_k)$, and depends on the particular method under consideration. Since the gradient of f is continuous, $d_k = -g_k$ vanishes if and only if x_k is a stationary point, in this way, by a suitable choice of the steplength α_k (see, e.g., the Armijo [2] or the Wolfe [64] line search procedure), global convergence can be easily ensured.

The theoretical properties of gradient methods mostly derive from the minimization of a convex quadratic function, in this way this dissertation mainly focuses on gradient methods applied to the following convex quadratic problem,

$$\min_x f(x) = \min_x \left(\frac{1}{2} x^T A x - b^T x \right), \tag{1.1}$$

where x and b are vectors in R^n, and A is a symmetric positive definite (SDP) $n \times n$ matrix. This setting can highlight the relevance of the eigenvalues of the Hessian matrix A, emphasizing the effects of a particular steplength instead of another, providing the basis for the development of gradient methods for constrained problems or general non-quadratic minimization problems.
In the rest of this thesis we denote by $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ the eigenvalues of the matrix A and by $\{d_1, d_2, \ldots, d_n\}$ a set of associated orthonormal eigenvectors. Since A is positive definite, all of its eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ are positive. Moreover in this work we consider the following two *assumptions*:

Assumption 1. *The eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ of A are such that:*

$$\lambda_1 > \lambda_2 > \cdots > \lambda_n.$$

Assumption 2. *Any starting point x_0 is such that*

$$g_0^T d_1 \neq 0;$$
$$g_0^T d_n \neq 0.$$

Furthermore, we denote by μ_i^k $i = 1, 2, \ldots, n$, the component of g_k along d_i, i.e.,

$$g_k = \sum_{i=1}^{n} \mu_i^k d_i. \tag{1.2}$$

Finally, for any gradient methods, it follows from the properties of a quadratic function that the gradient can be computed iteratively according to the following formula,

$$g_{k+1} = g_k - \alpha_k A g_k. \tag{1.3}$$

A general scheme for the gradient method to solve Problem (1.1) is reported in Algorithm 1. We have already noticed that the choice of the line search procedure

Algorithm 1 General scheme for the gradient method

```
choose $x_0 \in R^n$
compute $g_0 = Ax_0 - b$
set $k = 0$
while not stop_condition do
    choose $\alpha_k > 0$ with a line search procedure
    update $x_{k+1} = x_k - \alpha_k g_k$
    compute $g_{k+1} = g_k - \alpha_k A g_k$
    update $k = k + 1$
end while
```

(line five of Algorithm 1) is a crucial step in the development of the method. In the SD method proposed by Cauchy[1] in 1847 [13] for the solution of nonlinear systems of equations, α_k is chosen as

$$\alpha_k = \operatorname*{argmin}_{\alpha} f(x_k - \alpha g_k). \tag{1.4}$$

The exact minimization (1.4) leads, in fact, to the so called *Cauchy steplength*:

$$\alpha_k^{SD} = \frac{g_k^T g_k}{g_k^T A g_k}, \tag{1.5}$$

and the updated new iterate is therefore,

$$x_{k+1} = x_k - \frac{g_k^T g_k}{g_k^T A g_k} g_k.$$

The following result states the convergence and the convergence rate of the SD method.

Theorem 1.1.1 (SD convergence). *For any starting point $x_0 \in R^n$ the SD method converges to the unique minimum point x^* of f. Furthermore, the error norm satisfies:*

$$\|x_{k+1} - x^*\|_A^2 \leq \left(\frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n}\right)^2 \|x_k - x^*\|_A^2, \tag{1.6}$$

where the non euclidean A-norm is defined by $\|x\|_A^2 = x^T A x$.

By using the relation $Ax^* = b$ is straightforward to see that

$$\frac{1}{2}\|x - x^*\|_A^2 = f(x) - f(x^*),$$

so the previous theorem states that the function values converge to the minimum function value f^* at a linear rate. Being

$$\rho = \frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n} = \frac{\kappa(A) - 1}{\kappa(A) + 1}, \tag{1.7}$$

the convergence rate of the SD method slows down as the contours of f become more eccentric [48] and so, as the condition number $\kappa(A) = \lambda_1/\lambda_n$ increases, the convergence degrades.

To highlight the relationship among the condition number of the Hessian matrix, the convergence rate of the iterations, and the size of the oscillation in the gradient norm generated by the SD method, a deep analysis was carried out by Akaike in

[1]To simplify notations, the SD method using the Cauchy steplength will be simply denoted by SD.

1959 [1], and then investigated in depth by Nocedal et al. in 2002 [55]. Here we summarize the main aspects and results as follows.

Proposition 1.1.2 (Nocedal et al. 2002). *Under Assumptions 1-2, let $\{x_k\}$ be the sequence of iterates generated by the SD method applied to a strongly convex quadratic function. Then,*

1. *the following limits hold,*

$$\lim_{k\to\infty} \frac{(\mu_i^{2k})^2}{\sum_{j=1}^n (\mu_i^{2k})^2} = \begin{cases} \frac{c^2}{1+c^2}, & \text{if } i=1, \\ 0, & \text{if } i=2,\dots,n-1, \\ \frac{1}{1+c^2} & \text{if } i=n, \end{cases}$$

$$\lim_{k\to\infty} \frac{(\mu_i^{2k+1})^2}{\sum_{j=1}^n (\mu_i^{2k+1})^2} = \begin{cases} \frac{1}{1+c^2}, & \text{if } i=1, \\ 0, & \text{if } i=2,\dots,n-1, \\ \frac{c^2}{1+c^2} & \text{if } i=n, \end{cases}$$

   for some non-zero constant c.

2. *the components μ_1^{2k}, μ_n^{2k}, and $\mu_1^{2k+1}, \mu_n^{2k+1}$ have fixed signs for large k.*

The constant c is also responsible of the rate of convergence in the objective function, being

$$\lim_{k\to\infty} \frac{f(x_{k+1})}{f(x_k)} = \frac{c^2(\lambda_1/\lambda_n - 1)^2}{c^2(1+\lambda_1/\lambda_n)^2 + (c^2-1)^2\lambda_1/\lambda_n}, \tag{1.8}$$

whose right hand side is maximized when $c = 1$, giving the worst rate of convergence. The value of c is connected with the spectrum of the Hessian matrix, and its value depends on the ratio of the components of the gradient, as stated in the following result.

Lemma 1.1.3 (Nocedal et al. 2002). *Under the assumptions of Proposition 1.1.2, the constant c satisfies*

$$c = \lim_{k\to\infty} \frac{\mu_1^{2k}}{\mu_n^{2k}} = -\lim_{k\to\infty} \frac{\mu_n^{2k+1}}{\mu_1^{2k+1}}.$$

Moreover c is uniquely determined by the starting point x_0 and by the eigenvalues and the eigenvectors of A.

Next result is about the asymptotic behaviour of the sequence of steplengths $\{\alpha_k\}$.

Lemma 1.1.4 (Nocedal et al. 2002). *Let $\{x_k\}$ be the sequence of iterates generated by the SD method, and suppose Assumptions 1-2 hold, then,*

$$\begin{aligned}\lim_{k\to\infty} \alpha_{2k}^{SD} &= \frac{1+c^2}{\lambda_n(1+c^2\gamma)},\\ \lim_{k\to\infty} \alpha_{2k+1}^{SD} &= \frac{1+c^2}{\lambda_n(\gamma+c^2),}\end{aligned}$$

where $\gamma = \kappa(A)$ is the spectral condition number of A and c is the same constant as in Proposition 1.1.2.

Equation (1.8), Proposition 1.1.2, and Lemmas 1.1.3, 1.1.4 suggest that, in the SD method eventually

$$g_k \approx \mu_1^k d_1 + \mu_n^k d_n,$$

i.e., the SD method asymptotically reduces its search in the two-dimensional subspace spanned by d_1 and d_n, zigzagging between this two directions without being able to eliminate any of them from the search direction.
When $\lambda_1 = \lambda_n$, the Hessian matrix A is a multiple of the identity matrix and the antigradient direction always points at the solution which is achieved in one iteration.
In Figure 1.1 we show the behaviour of the normalized components of the gradient along the eigendirections d_1 and d_n for the SD method computed at the first 100 iterations, applied to a problem of type (1.1) with $n = 10$ variables, where,

$$A = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n),\ \lambda_i = 11i - 10,\ g_1^{(i)} = \sqrt{1+i},\ \ i = 1, \ldots, 10, \tag{1.9}$$

and b is a vector of zeros.

A more accurate analysis to show the behaviour of the gradient components at the computed solution of (1.9), can be made by mean of the following scalar quantities,

$$\beta_i = \sqrt{\sum_{j=1}^{i} (\mu_j^k)^2}, \quad i = 1, \ldots, n. \tag{1.10}$$

Figure 1.2 shows how the contribute of the components relative to the eigendirections d_i, with $i = 2, \ldots, 9$, is actually negligible, since the values of the correspondent β_i are unchanged.

In addition to these considerations, the recursive Formula (1.3) can be expressed as,

$$g_{k+1} = \alpha_k \left(\frac{1}{\alpha_k} g_k - A g_k \right), \tag{1.11}$$

suggesting that, in order to get faster convergence, a greedy approach like (1.5)

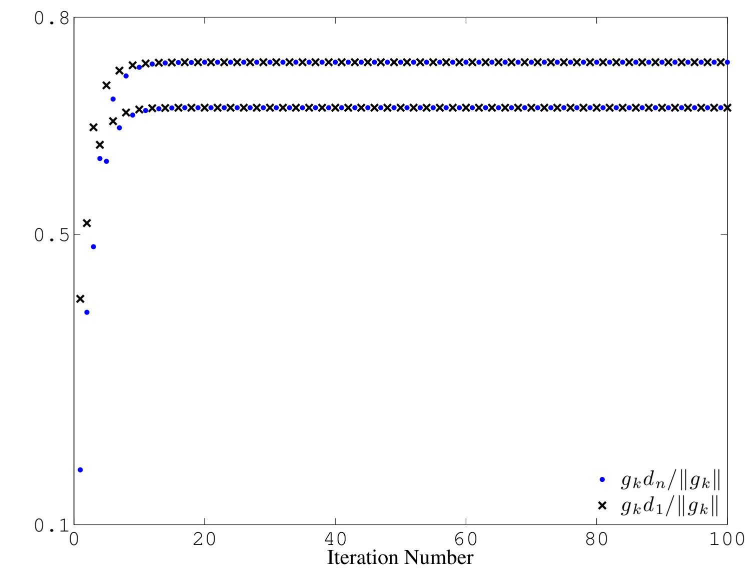


Figure 1.1. Problem 1.9, behaviour of the normalized components of the gradient along the eigendirection d_1 and d_n for the SD method.

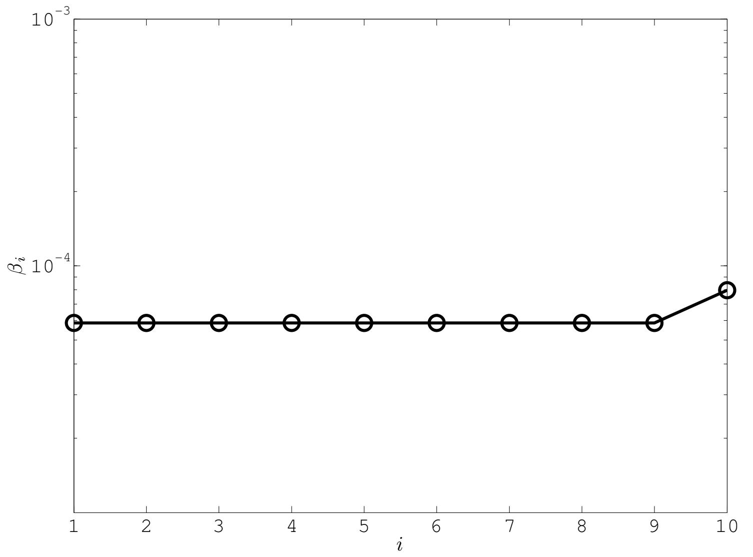


Figure 1.2. Problem 1.9, values of the scalars β_i at the computed solution for the SD method.

might be unsatisfactory, whereas fostering the search direction to align with an eigendirection of A could speed up the convergence of the algorithm [34]. Moreover, Formula (1.3) allows us to express the gradient in terms of the spectrum of the matrix A. From (1.3), in fact, if

$$g_0 = \sum_{i=1}^{n} \mu_i d_i,$$

then

$$g_{k+1} = \prod_{j=0}^{k} (I - \alpha_j A) g_0 = \sum_{i=1}^{n} \mu_i^{k+1} d_i, \tag{1.12}$$

where

$$\mu_i^{k+1} = \mu_i \prod_{j=0}^{k} (1 - \alpha_j \lambda_i). \tag{1.13}$$

If at the k-th iteration, $\mu_i^k = 0$ for some i, it follows from (1.12 - 1.13) that for $h > k$ it will be $\mu_i^h = 0$, and therefore the component of the gradient along d_i will be zero at all subsequent iterations. Furthermore, the condition $\mu_i^k = 0$ holds if and only if $\mu_i = 0$ or $\alpha_j = 1/\lambda_i$ for some $j \le k$.
Finally, next proposition holds.

Proposition 1.1.5. *Under Assumptions 1-2, a gradient method defined by the following iteration,*

$$x_{k+1} = x_k - \frac{1}{\lambda_k} g_k, \quad k = 1, \ldots, n,$$

reachs in at most n iterations the minimum point of Problem 1.1.

Note that the SD method has finite termination if and only if at some iteration the gradient is an eigenvector of A.

Despite its implementative simplicity, because of its oscillatory behaviour, the SD method is too slow to be used in practice. However, its theoretical properties can give a hint on a way to modify the step selection rule, in order to enable faster convergence.
Next section introduces a novel approach for choosing the steplength, originally proposed by Barziai and Borwein in 1988 [4]. The new presented method renewed the interest of the optimization community in gradient methods, opening new questions and suggesting new possible ways to speed up the convergence of the SD method.

1.2 The Barzilai and Borwein method

In 1988 Barzilai and Borwein [4] proposed a gradient method with a nonstandard strategy for choosing the steplength. The Barzilai and Borwein (BB) method follows an iterative *Quasi Newton*[2] scheme,

$$x_{k+1} = x_k - \frac{1}{\sigma_k} g_k, \tag{1.14}$$

where the scalar $\sigma_k > 0$ tends to approximate the Hessian matrix A by

$$B = \sigma_k I,$$

or its inverse matrix by

$$H = \frac{1}{\sigma_k} I = \beta I,$$

where I is the identity $n \times n$ matrix. These approximations can be obtained imposing a *secant condition* to find the value of σ which minimizes the error in

$$\sigma s_k - y_k = 0, \tag{1.15}$$

or the value of β which minimizes the error in

$$s_k - \beta y_k = 0, \tag{1.16}$$

where $s_k = x_k - x_{k-1}$, and $y_k = g_k - g_{k-1}$.
In the first case we get,

$$\min_\sigma s(\sigma) = \min_\sigma \|\sigma s_k - y_k\|^2,$$

which leads to:

$$\sigma_k = \frac{s_k^T y_k}{s_k^T s_k},$$

while in the second case we get,

$$\min h(\beta) = \min_\sigma \|s_k - \beta y_k\|^2,$$

which, remembering that $\beta = \frac{1}{\sigma}$, leads to

$$\sigma_k = \frac{y_k^T y_k}{s_k^T y_k}.$$

[2]For a further discussion on Quasi Newton methods see [27].

Imposing $\alpha_k = \frac{1}{\sigma_k}$, according to our previous notation, we have the two following steplengths:

$$\alpha_k^{BB1} = \frac{s_k^T s_k}{s_k^T y_k}, \tag{1.17}$$

$$\alpha_k^{BB2} = \frac{s_k^T y_k}{y_k^T y_k}. \tag{1.18}$$

It is strait forward to see that the steplength α_k^{BB1} is equal to α_{k-1}^{SD}, i.e., the Cauchy steplength at the previous iteration, while α_k^{BB2} is equal to α_{k-1}^{SDg}, where

$$\alpha_k^{SDg} = \frac{g_k^T A g_k}{g_k^T A^2 g_k} = \operatorname*{argmin}_\alpha \|\nabla f(x_k - \alpha g_k)\| = \operatorname*{argmin}_\alpha \|(I - \alpha A) g_k\|, \tag{1.19}$$

therefore, both the BB steplengths (1.17) and (1.18) can be seen as steplengths with one-step delay [24].
Since α_k^{BB1}, which is the steplength used in the BB method to determine the point x_{k+1}, is the Cauchy steplength used in the SD method at iteration k, the simplicity of the BB method lays in the fact that at each iteration, the method computes the step that minimizes the quadratic objective function along the antigradient direction but, instead of using this step at the k-th iteration, saves the step to be used in the next iteration [11].
Using the Mean-Value Theorem of integral calculus, we have,

$$y_k = \left[\int_0^1 \nabla^2 f(x_k + t s_k) dt\right] s_k,$$

Therefore, (1.17, 1.18) define the inverse of a Rayleigh quotient relative to the average Hessian matrix $\int_0^1 \nabla^2 f(x_k + t s_k) dt$. These coefficient are between the reciprocal of the minimum and the maximum eigenvalue of the average Hessian, which motivates the denomination of spectral method [11]. In fact, specifically for quadratic problems, it can be seen that, being $y_k = A s_k$, then

$$\alpha_k^{BB1} = \frac{s_k^T s_k}{s_k^T A s_k}; \quad \alpha_k^{BB2} = \frac{s_k^T A s_k}{s_k^T A^2 s_k},$$

in this way α_k^{BB1} can be seen as the inverse of the Rayleigh quotient of A at the vector s_k, while α_k^{BB2} as the inverse of the Rayleigh quotient of A at the vector $\sqrt{A} s_k$.
Finally, $\alpha_k^{BB2} \leq \alpha_k^{BB1}$, $\forall k$ (see, e.g., lemma 2.1 in [60]), therefore,

$$0 < \frac{1}{\lambda_1} < \alpha_k^{BB2} \leq \alpha_k^{BB1} < \frac{1}{\lambda_n}, \quad \forall k.$$

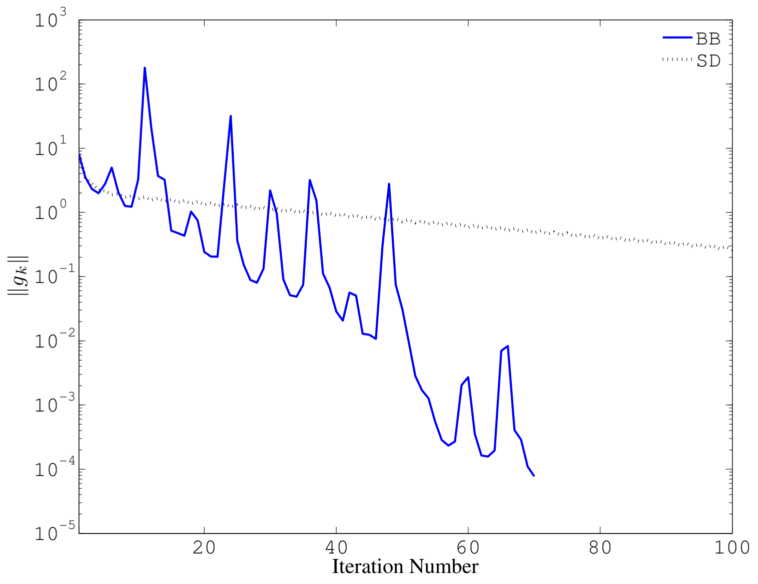


Figure 1.3. Problem 1.9, behavior of the sequence $\{\|g_k\|\}$ for the BB, and SD methods.

A consequence is that, under Assumptions 1-2, assuming that the first steplength is different from λ_1 or λ_n, it follows that the BB method does not have the property of finite termination.

Convergence of the Barzilai and Borwein method applied to Problem 1.1 has been deeply investigated. When in their original paper, Barzilai and Borwein showed that for two-dimensional strictly convex quadratic functions the BB method converges R-superlinearly, the optimization community tried to prove a similar result for the general n-dimensional case. Chronologically, this expectation was first disappointed by Fletcher in 1990 [31], who conjectured that only R-linear convergence should be expected. This hypotesis was confirmed by Raydan in 1993 [59], and Dai and Liao in 2002 [19], the former proving global convergence for the strictly convex quadratic case with any number of variables, the latter, the conjectured R-linear convergence result. Raydan also showed that the BB method is much more efficient than the SD method in solving large scale positive definite linear systems of equations.

Figure 1.3 shows the behaviour of the sequence $\{\|g_k\|\}$ for the BB and the SD method applied to Problem 1.9. Here we used the following condition,

$$\|g_k\| \leq 10^{-5}\|g_0\|, \tag{1.20}$$

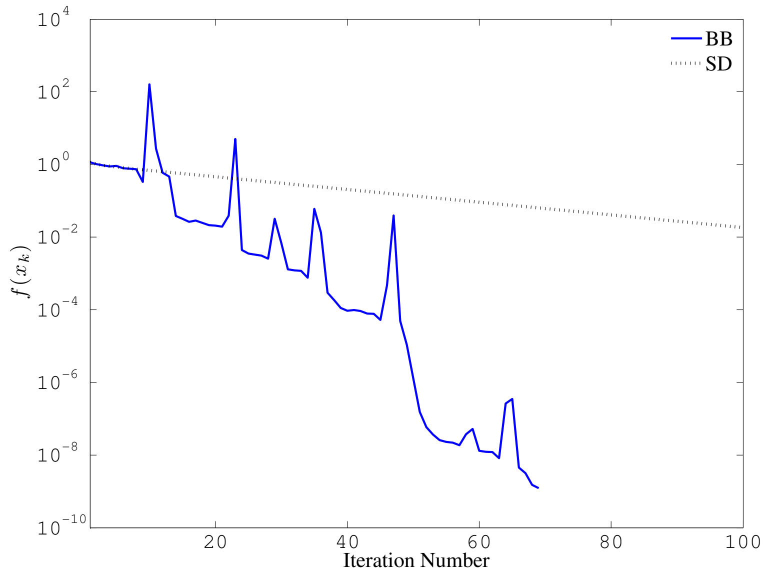


Figure 1.4. Problem 1.9, behavior of the sequence $\{f(x_k)\}$ for the BB, and SD methods.

as stopping criterion. We tested both the SD method and the BB method, the former converges in 70 iterations, while the latter takes more than 500 iterations, but only the first 100 iterations are displayed to emphasize the behaviour of the BB method.

The performance of the BB method is also examined in terms of function values in Figure 1.4, where the nonomotonicity of the sequence $\{f(x_k)\}$ is clearly visible.

The best efficiency of the BB method in solving large scale quadratic problems, opened the way to other gradient methods with retards (see, e.g., [35, 60, 22, 15, 18] and references therein) and for years, the credit for this improved efficiency has been given to the delay in step, rather than to other properties of the method, as the spectral properties of the Hessian matrix, which have been however object of study and investigation [32], but not yet embedded in an algorithmic framework to leverage their potential.

1.3 First attempts to escape from the zigzag path of the iterates

In 2002, Raydan and Svaiter [60] studied the positive effects of using relaxed Cauchy steplengths ending up with the Relaxed Steepest Descent (RSD) method. A general scheme of this method is given in Algorithm 2.

Algorithm 2 Relaxed Steepest Descent (RSD)

```
choose $x_0 \in R^n$
compute $g_0 = Ax_0 - b$
set $k = 0$
while not stop_condition do
    choose $\alpha_k > 0$ randomly in $[0, 2\alpha_k^{SD}]$
    update $x_{k+1} = x_k - \alpha_k g_k$
    compute $g_{k+1} = g_k - \alpha_k A g_k$
    update $k = k + 1$
end while
```

The new iterate x_{k+1} is computed along the antigradient direction, as in the SD method, but with a random choice of the steplength in the range of $\left[0, 2\alpha_k^{SD}\right]$, i.e.,

$$x_{k+1} = x_k - \rho_k \alpha_k^{SD} g_k, \quad \rho_k \in [0, 2] . \tag{1.21}$$

Note that, for $\rho_k = 1$, $\forall k$, Formula (1.21) reduces to the SD method iteration, while for $\rho_k = 2$, $\forall k$, no function reduction is produced since $f(x_k) = f(x_k - 2\alpha_k g_k)$. In the main time, $f(x_{k+1}) < f(x_k)$, $\forall \rho_k \in (0, 2)$. In the following, $\alpha_k = 2\alpha_k^{SD}$ will be called the *double Cauchy step.*
The RSD method is monotone, being $\alpha_k \leq 2\alpha_k^{SD}$, and its convergence can be proved under very mild assumptions on ρ_k, as stated in the next theorem.

Theorem 1.3.1 (Raydan and Svaiter 2002)**.** *If the sequence $\{\rho_k\}$ has an accumulation point $\bar{\rho}_k \in (0, 2)$, then the sequence of iterates $\{x_k\}$ converges to x^*, the optimal solution of Problem 1.1.*

The Cauchy steplength is still the best possible choice when the search direction is an eigenvector of the Hessian matrix, but in practice, avoiding this special and rare case, numerical experiments show that the RSD method largely outperforms the SD method, becoming a valid tool to accelerate the convergence of the classical SD method.
To have a better view of the behaviour of the RSD method, we compared it with the BB and the SD methods. Figure 1.5 shows the performances of the RSD, BB, and SD methods tested on Problem 1.9. As we can see, the BB method is still the fastest and most effective, followed by the RSD method with 129 iterations.

Other problems under consideration were randomly generated with dimensions $n = 10^3$. The Hessian matrix A was obtained by running the MATLAB function `sprandsym` with density 0.8, kind 1, and condition number $\kappa(A) = 10^2, 10^3, 10^4, 10^5$. For each instance of A, x^* was generated by `rand` with entries in $[-10, 10]$, and $b = Ax^*$ was used in the linear term. Furthermore, for each problem, five starting

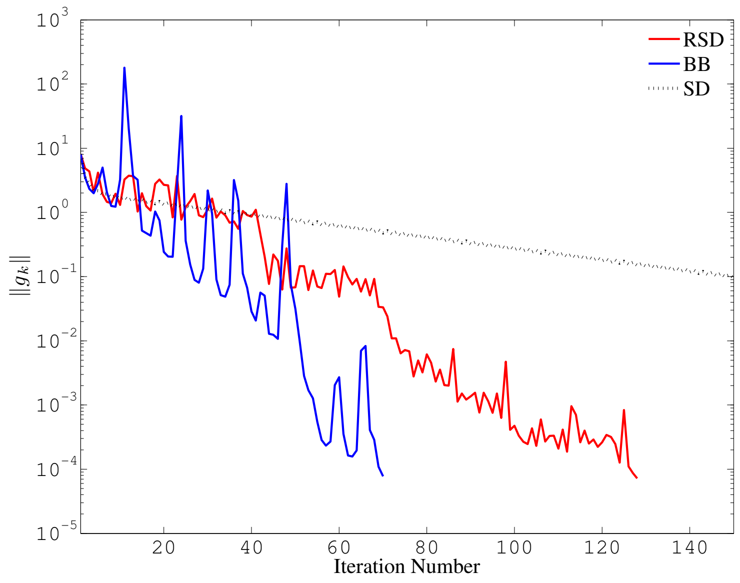


Figure 1.5. Problem 1.9, behaviour of the sequence $\{\|g_k\|\}$ for the RSD, BB, and SD methods.

points were generated by `rand` with entries in $[-10, 10]$. Finally, condition (2.32) was used as stopping criterion.
The results of the comparison are in Table 1.1, we can see that the BB method is the fastest in terms of iteration number, but the improvement made by the RSD method, with respect to the SD method, is impressive and not negligible.

$\kappa(A)$	SD	BB	RSD
10^2	316	65	107
10^3	2162	163	333
10^4	$> 10^4$	401	769
10^5	$> 10^4$	631	1222

Table 1.1. Iteration number for the random generated problems with $n = 10^3$ and condition number ranging from 10^2 to 10^5.

To investigate on gradient methods whose steplengths do not guarantee descent in the objective function, a hybrid nonmonotone method, called the Cauchy-Barzilai-Borwein (CBB) method, was presented [60]. The CBB method is, as suggested by its name, a combination of steepest descent and Barzilai-Borwein iterations. Observing that for quadratics, the BB steplength is equal to the Cauchy

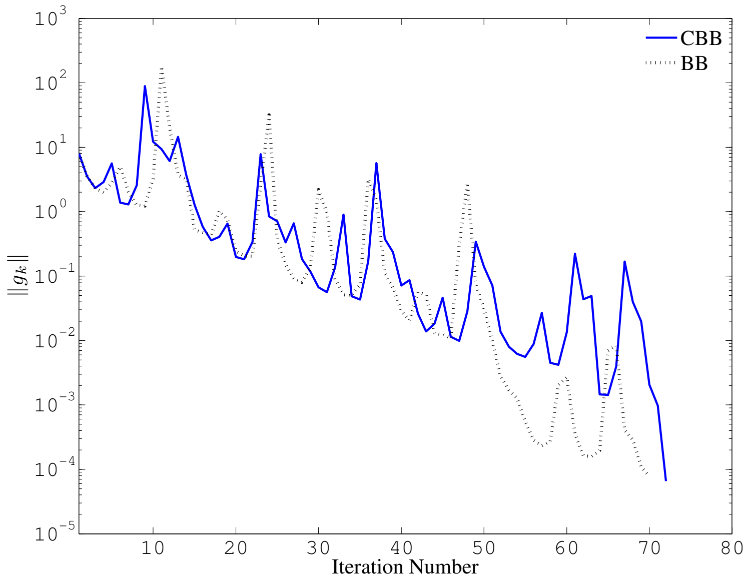


Figure 1.6. Problem 1.9, behaviour of the sequence $\{\|g_k\|\}$ for the CBB, and BB methods.

steplength taken at the previous iteration, the CBB method computes the Cauchy steplength once and uses it twice, in this way the computational cost of two consecutive CBB iterations is almost comparable with one steepest descent iteration[3]. From their numerical experiments, the authors observed the tendency of the CBB method to force gradient directions to approximate eigenvectors of the Hessian matrix A, as then observed in [32] for the BB method. Figures 1.6-1.7 show the behaviour of the gradient norm and the function values for the CBB and the BB methods applied to Problem 1.9, while, to verify the tendency to approximate some of the eigenvectors of A, we seek for the number of times in which g_k *is almost parallel to* Ag_k, in the same way as in [60], i.e., by using the cosine of the angle between g_k and Ag_k,

$$\cos(g_k, Ag_k) = \frac{g_k^T A g_k}{\|g_k\|\|Ag_k\|} > 1 - \epsilon, \tag{1.22}$$

where the variable ϵ was set to be equal to 5×10^{-4}. The results obtained by running the CBB, RSD and BB methods are shown in Table 1.2.

A bigger class of nonmonotone gradient methods with retards was previously introduced by Friedlander, Martinez, Molina and Raydan in 1999 [35] . Given a

[3]The CBB method performs one more vector sum and one more inner product than the SD method.

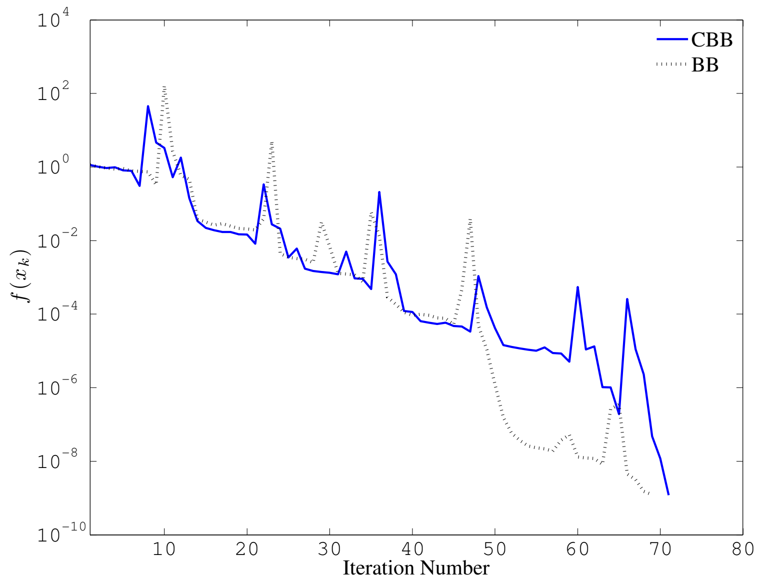


Figure 1.7. Problem 1.9, behaviour of the sequence $\{f(x_k)\}$ for the CBB, and BB methods.

Tol	CBB	RSD	BB
10^{-5}	1	0	3
10^{-8}	5	0	7
10^{-12}	5	1	7

Table 1.2. Problem 1.9, number of times in which the gradient is almost an eigenvector of the Hessian matrix for the CBB, RSD and BB methods tested with different values of the tolerance in the stopping criterion.

positive integer m, the new iteration is defined by

$$x_{k+1} = x_k - \alpha_{\nu_k} g_k, \tag{1.23}$$

where α_{ν_k} is the Cauchy step taken at a previous iteration $\nu_k \in \left\{k, k-1, \dots, \bar{k}\right\}$, with $\bar{k} = \max\{0, k-m\}$. This can be considered as a generalization of the SD and the BB methods, including the classical Cauchy steplength and the BB steplengths as special cases, moreover, the following result holds.

Theorem 1.3.2 (Friedlander et al. 1999)**.** *Let* $\{s_k\} = \{(x_k - x_{k-1})\}$ *be the sequence generated by the following iterative scheme:*

$$x_{k+1} = x_k - \alpha_k g_k,$$
$$\alpha_k = \frac{s_{\nu_k}^T s_{\nu_k}}{s_{\nu_k}^T A s_{\nu_k}}.$$

Assume that the sequence $\frac{s_k}{\|s_k\|}$ is convergent to a normalized vector $s \in R^n$, then

$$\lim_{k\to\infty} \frac{1}{\alpha_k} = s^T A s, \tag{1.24}$$

s is an eigenvector of A with eigenvalue $s^T A s$ and the convergence of $\{x_k\}$ is Q-superlinear.

It is worth noting that Formula (1.24) relates the inverse of the steplength to an eigenvalue of the Hessian matrix.

To conclude this section, we can observe that a first attempt to escape from the zigzag path of the iterates, aligning the search direction with an eigendirection of the Hessian matrix, seems to speed up the convergence of the classical SD method. Next section completes the chapter, introducing other gradient methods whose surprising behaviour is due to the alternation of some fixed number of Cauchy steplengths, and other steps computed through Formula (1.5).

1.4 Alternate steplengths gradient methods

A large contribute to the problem of finding a steplength selection rule which enables fast convergence, preserving the monotonicity of the method, was given by Dai and Yuan (see, e.g., [23] and references therein).
They began their research studying the properties of the Optimal Steplength (OPT1) method [28]:

$$\alpha^{OPT1} = \frac{2}{\lambda_1 + \lambda_n}, \tag{1.25}$$

which comes up from the minimization of the quantity $\|I - \alpha A\|$.
Although α^{OPT1} has the best convergence result, it is practical useless, since λ_1 and λ_n are normally unknown to the user, unless they can be somehow estimated beforehand, for this reason the performance of the OPT1 method are unknown. This fact gave a strong motivation in the design a new gradient method whose steplength could have a connection with α^{OPT1}, thus the following steplength was proposed [20, 21]:

$$\alpha_k = \frac{\|g_k\|}{\|A g_k\|}. \tag{1.26}$$

Formula (1.26) tends to (1.25), for $k \to \infty$, so, as expected, the proposed method (in the following simply denoted by OPT2) makes it possible to approximate the behaviour of the OPT1 method. Although Formula (1.26) produces better results compared to those obtained using the SD method, it still has the drawback that the gradient g_k tends to zero along two directions alternatively, being enable to escape

from the zigzag path of the iterates, as for the SD method.
In order to accelerate the convergence, an hybrid gradient method, called the Alternate Step (AS) method, which alternates α_k^{SD} and α_k^{BB} steps, was suggested [15],

$$\alpha_k^{AS} = \begin{cases} \alpha_k^{SD} & if \ \text{k} \ is \ odd, \\ \alpha_k^{BB} & if \ \text{k} \ is \ even. \end{cases} \tag{1.27}$$

Numerical experiments showed that the AS method is a promising alternative to the BB method, being Q-superlinear convergent for two dimensional strictly convex quadratics, and R-linearly convergent in any dimension. Its numerical superiority to the SD method is addressed to its nonmonotonic behaviour, but this aspect is in contrast with the fact that, if the dimension of the problem is very large, a monotone algorithm should be preferred to a nonmonotone one, since the object is to minimize the function, as pointed out in [22]. The advantage of retaining monotonicity has been further highlighted by several authors (see, e.g., [33]), especially when dealing with non-quadratic problems.
To probe the performance of a possible monotone gradient method faster than the SD method, the Alternate Minimization (AM) method was proposed [22]. The main idea of this method is to alternately impose the minimization of the the gradient norm and of the function value along the line. This leads to,

$$\alpha_k^{AM} = \begin{cases} \frac{g_k^T A g_k}{g_k^T A^2 g_k} & if \ \text{k} \ is \ odd, \\ \alpha_k^{SD} & if \ \text{k} \ is \ even. \end{cases} \tag{1.28}$$

Furthermore, being

$$\frac{g^T A g}{g^T A^2 g} \leq \frac{g^T g}{g^T A g}, \ \forall g \in R^n - \{0\},$$

the AM method is monotone.

Several works have been recently devoted to design faster gradient methods [23, 67, 33, 26, 24], many of which are monotone, whose common basic idea is to combine Cauchy steps with other steplengths. Yuan supports this approach through a theoretical and computational analysis leading to the conclusion that *a good gradient method would use at least one exact line search (the Cauchy step) in every few iterations* [67]. Through all his work, particular interest must be given to the following steplength formula[4] [66],

$$\alpha_k^Y = \frac{2}{\sqrt{\left(\frac{1}{\alpha_{k-1}^{SD}} - \frac{1}{\alpha_k^{SD}}\right)^2 + \frac{4\|g_k\|^2}{\|s_{k-1}\|^2}} + \frac{1}{\alpha_{k-1}^{SD}} + \frac{1}{\alpha_k^{SD}}}, \tag{1.29}$$

[4]Remember $s_{k-1} = x_k - x_{k-1}$.

which has the important property that for two dimensional quadratic functions if $\alpha_1 = \alpha_1^{SD}$, $\alpha_2 = \alpha_2^{Y}$, and $\alpha_3 = \alpha_3^{SD}$ then x_4 gives the minimum in exact arithmetic [23]. Two monotone gradient methods were therefore proposed [66], based on the following two different steplength selection rules ((1.30) and (1.31)).

$$\alpha_k^{Y1} = \begin{cases} \alpha_k^{SD} & if \ \ k \ \ is \ \ odd, \\ \alpha_k^{Y} & \text{if}\, k \ \ is \ \ even \end{cases} \tag{1.30}$$

$$\alpha_k^{Y2} = \begin{cases} \alpha_k^{SD} & if \quad \text{mod}\ (k,3) \neq 0, \\ \alpha_k^{Y} & \text{if} \quad \text{mod}\ (k,3) = 0. \end{cases} \tag{1.31}$$

Numerical experiments in [66] shows that this two methods are comparable with the BB method for large scale quadratic problems, and better than the BB method for small scale quadratic problems. Surprisingly, even if the two methods look like each other, (1.30) behaves much worse than (1.31). A possible explanation of this better behaviour is given in [23], where an experimental analysis is carried out to prove that in (1.31), the gradient components with respect to the eigenvectors of the Hessian matrix, are decreasing together. This property was called the *decreasing together* property. A similar behaviour had been observed in [32, 16] for the BB method and other cyclical variant of the SD method.

In order to confirm this particular behaviour, we consider Problem 1.9 and we look for the size s of the following set,

$$\left\{ i : 1 \leq i \leq n \ \middle| \ \frac{|g_{\bar{k}}^i|}{\|g_{\bar{k}}\|} > n\epsilon \right\}, \tag{1.32}$$

where $\bar{k}$ is the iteration in which the absolute tolerance ϵ on the stopping criterion in reached. This analysis is also reported in [23], but with a slightly different definition of the set which leads, however, to similar considerations. We know that asymptotically in the SD, RSD and OPT2 methods, the gradient belongs to the subspace spanned by its two dominant eigencomponents, in this way we expect s to be equals to 2, since there will be two gradient components relatively bigger than the others. Table 1.3 reports the values of s for the SD, RSD, OPT2, AS, AM, Y1 and Y2 methods for different values of ϵ. As expected, the value of s for the SD, RSD and OPT2 methods is exactly 2 for all the values of ϵ, while the value of s for the other methods increases as ϵ becomes smaller. The BB and the Y2 methods show a similar behaviour, and numerical experiments confirm that this two methods give the best results in terms of iteration number. The RSD method, despite its poor value of s behaves much better than the SD and the OPT2 methods, suggesting that the *decreasing together* property is a first-level consideration to understand the

ϵ	SD	RSD	OPT2	AS	Y1	Y2	BB
10^{-3}	2	2	2	4	4	6	6
10^{-6}	2	2	2	6	9	10	9
10^{-9}	2	2	2	8	9	10	10
10^{-12}	2	2	2	9	10	10	10

Table 1.3. Problem 1.9, values of s for different gradient methods

difference among gradient methods, but, as we will see in the next chapter, it is not the only responsible for the improved convergence speed.

More efficient gradient methods were presented in [23]. In this case the following version of Formula (1.29) was given:

$$\alpha_k^Y = \frac{2}{\sqrt{\left(\frac{1}{\alpha_{k-1}^{SD}} - \frac{1}{\alpha_k^{SD}}\right)^2 + \frac{4\|g_k\|^2}{\left(\alpha_{k-1}^{SD}\|g_{k-1}\|\right)^2}} + \frac{1}{\alpha_{k-1}^{SD}} + \frac{1}{\alpha_k^{SD}}}. \qquad (1.33)$$

Formula (1.33) is equivalent to the former if $s_{k-1} = -\alpha_k^{SD} g_{k-1}$, while if $\alpha_{k-1} \neq \alpha_{k-1}^{SD}$ the two formulas might be different.
The idea was to consider a larger class of gradient methods in which the steplength (1.29) is computed for a fixed number of times after a fixed number of exact line search iterations, following the scheme:

$$\alpha_k^{DY} = \begin{cases} \alpha_k^{SD} & \text{if } \mathrm{mod}(k, h_1) < h, \\ \alpha_k^Y & \text{otherwise,} \end{cases} \qquad (1.34)$$

in which α_k^{YD} is recomputed at each iteration. The case of the choice $h = 2$ and $h_1 = 4$ (i.e., $m = 2$), is suggested in [23] as the most efficient one. With this choice the method becomes:

$$\alpha_k^{DY} = \begin{cases} \alpha_k^{SD} & if \quad \mathrm{mod}\ (k, 4) = 0, 1 \\ \alpha_k^Y & \quad otherwise. \end{cases} \qquad (1.35)$$

The above method, which will be henceforth simply denoted by DY, computes the first steplength (1.33) after 2 exact line search iterations. Furthermore, it is a monotone method, and despite the similarity among all the previous methods, it is significantly faster than the other analysed gradient methods, being a strong opponent of the BB method, therefore it will be used, together with the BB method, in the following development of this dissertation as benchmark.

Chapter 2

Spectral properties of gradient methods

In Chapter 1 we saw that the choice of the steplength is a critical issue in the definition of a gradient method, while the use of the gradient as search direction leads to very effective algorithms. In particular, we saw how, by mean of the analysis carried out by Akaike and Nocedal, the zigzag path made by the iterates in the steepest descent method is the main responsible for its slow convergence, forcing the gradient in the two dimensional subspace spanned by the two dominant eigenvectors.
Many modified steplengths are possible, producing faster gradient methods whose surprising behaviour has only been partially explained by an experimental analysis, conjecturing that they randomly cross the spectrum of the Hessian matrix [32], or they have a sort of *decreasing together* property [23]. As a consequence, with a lack of strong theoretical justifications, the convergence is speeded up, but no guarantee of monotonicity is usually given.

In this chapter we show how, moving from the properties of the SD method, highlighted in [1, 55], we can derive a deeper theoretical analysis to support the development of new gradient methods, whose computational results are superior of those of the most competitive methods in literature[1], while preserving monotonicity.

[1]see the methods reviewed in Chapter 1

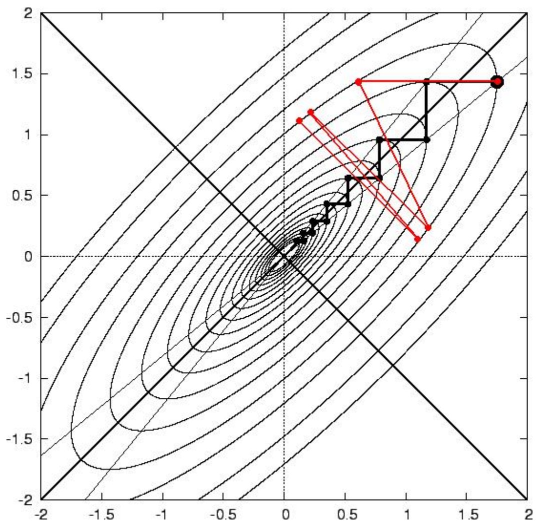


Figure 2.1. Behaviour of the sequence x_k generated by a gradient method using a double Cauchy step at each iteration. After few iterations an evident alignment of the search direction with one of the axis is produced.

2.1 A modified form of relaxation

We already noticed that Formula 1.11, reported here for sake of clarity,

$$g_{k+1} = \alpha_k \left(\frac{1}{\alpha_k} g_k - A g_k \right), \tag{2.1}$$

suggests that, in order to get faster convergence, fostering the search direction to align with an eigendirection of A, could speed up the convergence of the classical SD method [34]. An intuitive approach aimed at improving the SD method is shown in Figure 2.1, where the path that a gradient method would do adopting a double Cauchy step at each iteration is represented in red. This simple illustration shows how, after few double Cauchy steps[2], a significant alignment of the search direction with one of the two dominant eigendirections can be reached. In this way, a double Cauchy step, although useless in reducing the objective function of the problem, could reveal meaningful information about the spectrum of A, as confirmed by the next proposition.

Proposition 2.1.1. *Let us consider the sequences $\{x_k\}$ and $\{g_k\}$ generated by a gradient method using the double Cauchy steplength*

$$\alpha_k = 2\alpha_k^{SD}, \tag{2.2}$$

[2] $\alpha_k = 2\alpha_k^{SD}$.

at each iteration, then

$$\lim_k \frac{g_k}{\prod_{j=1}^k (1-\alpha_j\lambda_1)} = \mu_1 d_1, \tag{2.3}$$

$$\lim_k \alpha_k = \frac{2}{\lambda_1}, \tag{2.4}$$

$$\lim_k \nabla f\left(x_k - \alpha_k^{SD} g_k\right) = 0. \tag{2.5}$$

Proof. It is

$$g_k = \mu_1 \left(\prod_{j=1}^k (1-\alpha_j\lambda_1)\right) d_1 + \sum_{i=2}^n \mu_i \left(\prod_{j=1}^k (1-\alpha_j\lambda_i)\right) d_i$$

and hence

$$\frac{g_k}{\prod_{j=1}^k (1-\alpha_j\lambda_1)} = \mu_1 d_1 + \sum_{i=2}^n \mu_i \prod_{j=1}^k \frac{(1-\alpha_j\lambda_i)}{(1-\alpha_j\lambda_1)} d_i. \tag{2.6}$$

Furthermore,

$$\frac{\lambda_n}{2} \leq \frac{1}{\alpha_j} \leq \frac{\lambda_1}{2} \tag{2.7}$$

and then

$$1-\alpha_j\lambda_n \geq -1, \quad 1-\alpha_j\lambda_1 \leq -1. \tag{2.8}$$

If we set $\theta = \lambda_1 - \lambda_2$, then $\lambda_1 \geq \lambda_i + \theta$, and it follows that

$$1-\alpha_j\lambda_i \geq 1-\alpha_j\left(\lambda_1-\theta\right)$$

and hence, by (2.8),

$$\frac{1-\alpha_j\lambda_i}{1-\alpha_j\lambda_1} \leq 1 + \frac{\theta\alpha_j}{1-\alpha_j\lambda_1}. \tag{2.9}$$

By using (2.7) we get

$$\frac{\theta\alpha_j}{1-\alpha_j\lambda_1} = \frac{\theta}{\frac{1}{\alpha_j}-\lambda_1} \leq \frac{\theta}{\frac{\lambda_n}{2}-\lambda_1}$$

and thus

$$\frac{1-\alpha_j\lambda_i}{1-\alpha_j\lambda_1} \leq 1-\rho, \tag{2.10}$$

with

$$\rho = \frac{2\theta}{2\lambda_1-\lambda_n}. \tag{2.11}$$

Since

$$\frac{1-\alpha_j\lambda_i}{1-\alpha_j\lambda_1} = -1 + \frac{2-\alpha_j(\lambda_1+\lambda_i)}{1-\alpha_j\lambda_1} \tag{2.12}$$

and, by (2.7),

$$\begin{aligned}&\frac{2-\alpha_j(\lambda_1+\lambda_i)}{1-\alpha_j\lambda_1} \geq \frac{2-\frac{2}{\lambda_n}(\lambda_1+\lambda_i)}{1-\alpha_j\lambda_1} = \\ &\frac{2\lambda_n-2\lambda_1-2\lambda_i}{\lambda_n(1-\alpha_j\lambda_1)} = \frac{2\lambda_1-2\lambda_n+2\lambda_i}{\alpha_j\lambda_1\lambda_n-\lambda_n} \geq \\ &\frac{2\theta+2\lambda_i}{2\lambda_1-\lambda_n} \geq \rho,\end{aligned}$$

we get

$$-1+\rho \leq \frac{1-\alpha_j\lambda_i}{1-\alpha_j\lambda_1} \leq 1-\rho.$$

Therefore, by (2.6), we have (2.3).

Because of (2.3)

$$\lim_k \alpha_k = 2\frac{\mu_1^2 d_1^T d_1}{\mu_1^2 d_1^T A d_1},$$

and, since $Ad_1 = \lambda_1 d_1$, we have

$$\lim_k \alpha_k = 2\frac{\mu_1^2 d_1^T d_1}{\mu_1^2 \lambda_1 d_1^T d_1} = \frac{2}{\lambda_1}.$$

Thus (2.4) holds.

Finally, in order to prove (2.5) we first note that the sequence $\{\|g_k\|\}$ is bounded above, and so is $\{\prod_{j=1}^k (1-\alpha_j\lambda_1)\}$ because of (2.3). Then

$$\begin{aligned}&\lim_k \nabla f\left(x_k - \frac{\alpha_k}{2} g_k\right) = \lim_k \left(g_{k-1} - \frac{\alpha_k}{2} A g_{k-1}\right) = \\ &\lim_k \prod_{j=1}^{k-1}(1-\alpha_j\lambda_1)\left(\frac{g_{k-1}}{\prod_{j=1}^{k-1}(1-\alpha_j\lambda_1)} - \frac{\alpha_k}{2} A \frac{g_{k-1}}{\prod_{j=1}^{k-1}(1-\alpha_j\lambda_1)}\right) = \\ &\lim_k \prod_{j=1}^{k-1}(1-\alpha_j\lambda_1)\left(\mu_1 d_1 - \frac{1}{\lambda_1}\mu_1 A d_1\right) = \lim_k \prod_{j=1}^{k-1}(1-\alpha_j\lambda_1)\left(\mu_1 d_1 - \mu_1 d_1\right) = 0,\end{aligned}$$

hence (2.5) holds and the proof is complete. □

Proposition 2.1.1 gives strength to the intuition that a double Cauchy step has a significant impact in terms of alignment of the gradient with the eigenvector d_1. To verify the described advantage we roughly consider a *modified* version of the SD method, named the SDM method, in which 5 consecutive double Cauchy steps are performed every 10 Cauchy steps. We apply the SDCM method to a simple, illustrative problem, to show how, this rough modification of the SD method, produces a not negligible effect in decreasing the overall number of iterations. For this reason,

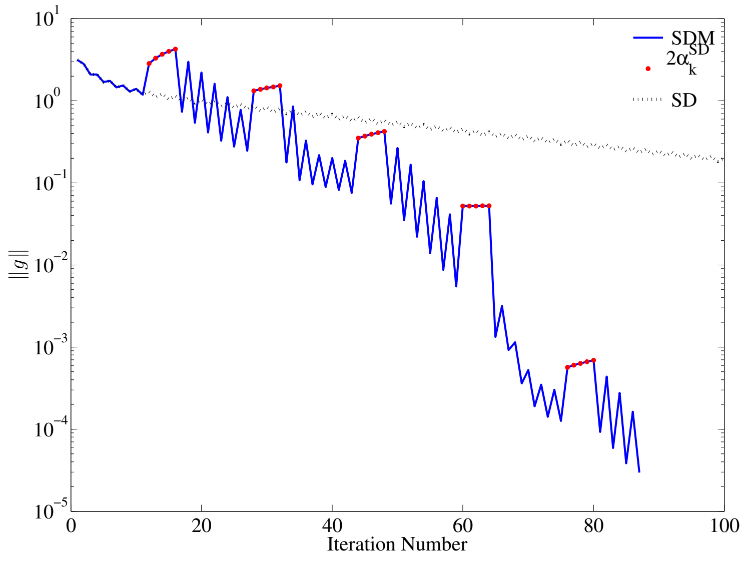


Figure 2.2. Problem 2.13, convergence history for the SD and SDM methods, red dots represent the values of the gradient norm when a double Cauchy step is adopted.

we consider a problem of $n = 10$ variables, where,

$$\kappa(A) = 10^2,\ A = \mathrm{diag}(\lambda_n, \dots, \lambda_1),\ b = (1, \dots, 1)^T,\ x_0 = (0, \dots, 0)^T, \tag{2.13}$$

being $\lambda_n = 1,\ \lambda_n = 100$, and λ_i randomly generated in $[\lambda_n, \lambda_1]$ for $i = n-1, \dots, 2$. Finally, we use $\|g_k\| \leq 10^{-5}\|g_0\|$ as stop condition.

The effect of the SDM method is shown in Figure 2.2. As we can see, on one hand, the norm of the gradient remains almost constant during the five iterations in which the double Cauchy step is adopted, on the other hand, however, a substantial reduction, produced at the end of these *special iterations*, increases the decay of the gradient norm, thus speeding up the convergence of the method, allowing us to save some iterations.

Starting from these considerations, Proposition 2.1.1 also suggests how to modify the RSD method [60], considering an over-relaxation rather than an random relaxation of the Cauchy step in $[0, 2]$. To this end, we consider a steplength of the form

$$\alpha_k \in [0.8\alpha_k^{SD}, 2\alpha_k^{SD}]. \tag{2.14}$$

This modification gives rise to a different version of the RSD method, called the Relaxed Steepest Descent with Alignment (RSDA), summed up in Algorithm 3.

The RSDA method is tested on Problem 2.13, and compared with the SD and

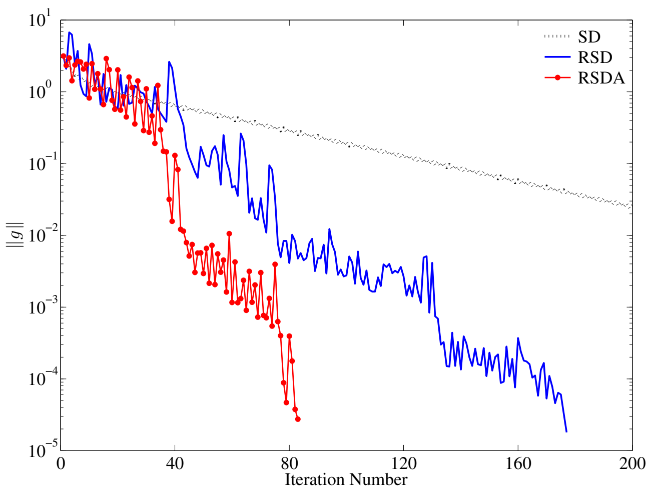


Figure 2.3. Problem 2.13, convergence history for the SD and RSD, and RSDA methods.

```
Algorithm 3 Relaxed Steepest Descent with Alignment (RSDA)
  choose $x_0 \in R^n$
  compute $g_0 = Ax_0 - b$
  set $k = 0$
  while not stop_condition do
    choose $\alpha_k > 0$ randomly in $[0.8\alpha_k^{SD}, 2\alpha_k^{SD}]$
    update $x_{k+1} = x_k - \alpha_k g_k$
    compute $g_{k+1} = g_k - \alpha_k A g_k$
    update $k = k + 1$
  end while
```

the RSD methods. The results are shown in Figure 2.3, where we can clearly see that the RSDA method outperform the RSD method.
Due to the randomness inherent in the two methods, a careful and deeper analysis is needed in order to evaluate the effectiveness of the RSDA method, especially to check the validity of our claim about the advantage in using the RSDA method rather than the RSD method. Extensive numerical tests will be considered in in the Numerical Experiments section to get a clear picture of the numerical behaviour of this algorithmic approach.

2.2 A new cyclical framework

We now suggest another way of modifying the SD method to force the gradients into a one dimensional subspace as the iterations progress. We first show that the sequence of steplengths $\{\alpha_k^{SD}\}$ in the SD method gives asymptotically some meaningful information about the spectrum of the Hessian matrix.

Proposition 2.2.1. *Let us consider the sequence $\{x_k\}$ generated by the SD method applied to Problem (1.1), and suppose that Assumptions 1-2 hold. Then, the sequences $\{\alpha_{2k}^{SD}\}$ and $\{\alpha_{2k+1}^{SD}\}$ are converging and*

$$\lim_k \left(\frac{1}{\alpha_{2k}^{SD}} + \frac{1}{\alpha_{2k+1}^{SD}} \right) = \lambda_1 + \lambda_n \,. \tag{2.15}$$

Proof. By Lemma 3.3 in [55], it is

$$\begin{aligned} \lim_k \alpha_{2k}^{SD} &= \frac{1+c^2}{\lambda_n(1+c^2\gamma)}, \\ \lim_k \alpha_{2k+1}^{SD} &= \frac{1+c^2}{\lambda_n(\gamma+c^2)}, \end{aligned}$$

where c is the same constant as in Proposition 1.1.2 and $\gamma = \kappa(A)$; then (3.12) trivially follows. □

Proposition 2.2.2. *Under Assumptions 1-2, the sequence $\{x_k\}$ generated by a gradient method using at each step a constant step length*

$$\widehat{\alpha} = \frac{1}{\lambda_1 + \lambda_n}, \tag{2.16}$$

converges to x^*. Moreover,

$$\lim_k \frac{\mu_h^k}{\mu_n^k} = \frac{\mu_h}{\mu_n} \lim_k \left(\frac{\lambda_n}{\lambda_1} + \frac{\lambda_1 - \lambda_h}{\lambda_1} \right)^k = 0 \qquad h = 1, 2, ..., n-1, \tag{2.17}$$

where μ_i^k $(i = 1, 2, ..., n)$ is defined in (1.13).

Proof. Since $\alpha_k^{SD} \geq 1/\lambda_1$ for any k, then $\alpha_k^{SD} \geq \widehat{\alpha}$; therefore, Proposition 1.3.1 applies and $\lim_k x_k = x^*$. From (1.13) we have that

$$\mu_h^k = \mu_h \left(\frac{\lambda_1 + \lambda_n - \lambda_h}{\lambda_n + \lambda_1} \right)^k, \quad \mu_n^k = \mu_n \left(\frac{\lambda_1}{\lambda_n + \lambda_1} \right)^k$$

and (2.17) clearly holds. □

Relation (2.17) indicates that, if the hypotheses of Proposition 2.2.2 hold, then the sequences $\{\mu_h^k\}$, for $h < n$, go to zero faster than $\{\mu_n^k\}$. Thus, a gradient method with steplength (2.16) tends to align the search direction with the eigendirection of A corresponding to the minimum eigenvalue λ_n.

We note that the constant steplength (2.16) is half of the theoretically optimal constant steplength (1.25) in [28], shown here again for reasons of clarity,

$$\alpha^{OPT1} = \frac{2}{\lambda_1 + \lambda_n}.$$

In Chapter 1 we saw that the steplength

$$\alpha_k^{OPT2} = \frac{\|g_k\|}{\|Ag_k\|}, \tag{2.18}$$

originally proposed in [21], converges to α^{OPT1}, allowing to approximate the extreme eigenvalues of A. However, despite its nice theoretical features, we saw that a gradient method using (2.18) as steplength at each iterations, only leads to a slight reduction in the number of iterations with respect to the SD method. Propositions 1.1.2 and 2.2.1 suggest an approach different from that in [21], aimed to speed up the convergence of the SD method by forcing the algorithm search directions in the one-dimensional subspace spanned by the eigendirection d_n. Of course, computing the exact value of (2.16) is unrealistic, but Proposition 2.2.1 suggests that, for k sufficiently large,

$$\widetilde{\alpha}_k = \left(\frac{1}{\alpha_k^{SD}} + \frac{1}{\alpha_{k+1}^{SD}} \right)^{-1} \tag{2.19}$$

can be used as an approximate value for (2.16).

Besides the steplength (2.19), the Yuan steplength,

$$\alpha_k^{Y} = \frac{2}{\sqrt{\left(\frac{1}{\alpha_{k-1}^{SD}} - \frac{1}{\alpha_k^{SD}} \right)^2 + \frac{4\|g_k\|^2}{\left(\alpha_{k-1}^{SD}\|g_{k-1}\|\right)^2}} + \frac{1}{\alpha_{k-1}^{SD}} + \frac{1}{\alpha_k^{SD}}}, \tag{2.20}$$

has also a favorable alignment property, as we can see in the following analysis.
In order to find a relationship between the asymptotic behaviour of α_k^Y and the eigenvalues of the matrix A, we provide a different expression of α_k^Y, which also highlights some connection with $\widetilde{\alpha}_k$.

Lemma 2.2.3. *The Yuan steplength (2.20) can be written as*

$$\alpha_k^Y = \frac{2}{\widetilde{\alpha}_k^{-1} + \sqrt{\widetilde{\alpha}_k^{-2} - 4\rho_k}} = \frac{2\widetilde{\alpha}_k}{1 + \sqrt{1 - 4\rho_k \widetilde{\alpha}_k^2}}, \tag{2.21}$$

where $\widetilde{\alpha}_k$ is defined in (2.19) and

$$\rho_k = \frac{1}{\alpha_{k-1}^{SD}\alpha_k^{SD}} - \frac{\|g_k\|^2}{\|g_{k-1}\|^2} \frac{1}{\left(\alpha_{k-1}^{SD}\right)^2}. \tag{2.22}$$

Proof. We first observe that

$$\alpha_k^Y = 2\left(\sqrt{\left(\frac{1}{\alpha_{k-1}^{SD}} - \frac{1}{\alpha_k^{SD}}\right)^2 + 4\frac{\|g_k\|^2}{\left(\alpha_{k-1}^{SD}\|g_{k-1}\|\right)^2}} + \frac{1}{\alpha_{k-1}^{SD}} + \frac{1}{\alpha_k^{SD}}\right)^{-1} =$$

$$2\left(\sqrt{\left(\frac{1}{\alpha_{k-1}^{SD}} + \frac{1}{\alpha_k^{SD}}\right)^2 - 4\frac{1}{\alpha_{k-1}^{SD}\alpha_k^{SD}} + 4\frac{\|g_k\|^2}{\left(\alpha_{k-1}^{SD}\|g_{k-1}\|\right)^2}} + \frac{1}{\alpha_{k-1}^{SD}} + \frac{1}{\alpha_k^{SD}}\right)^{-1}.$$

Then, the thesis trivially follows from the definition of $\widetilde{\alpha}_k$. □

We are now ready to analyse the asymptotic behaviour of α_k^Y.

Proposition 2.2.4. *Let $\{x_k\}$ be the sequence generated by the SD method applied to problem (1.1), starting from any point x_0, and suppose that Assumptions 1-2 hold. Then,*

$$\lim_k \alpha_k^Y = \frac{1}{\lambda_1}, \tag{2.23}$$

$$\lim_k \rho_k = \lambda_1\lambda_n, \tag{2.24}$$

where α_k^Y and ρ_k are defined in (2.20) and (2.22), respectively.

Proof. By Propositions 1.1.3 -1.1.4 we have

$$\lim_k \frac{1}{\alpha_k^{SD}\alpha_{k-1}^{SD}} = \frac{\lambda_n^2(\gamma + c^2)(1 + c^2\gamma)}{(1 + c^2)^2},$$

$$\lim_k \frac{\|g_k\|^2}{(\alpha_{k-1}^{SD}\|g_{k-1}\|)^2} = \frac{\lambda_n^2 c^2(\gamma - 1)^2}{(1 + c^2)^2},$$

where $\gamma = \kappa(A)$, and therefore

$$\lim_k \rho_k = \frac{\lambda_n^2}{(1 + c^2)^2}\left[(\gamma + c^2)(1 + c^2\gamma) - c^2(\gamma - 1)^2\right] = \tag{2.25}$$

$$\frac{\lambda_n^2}{(1 + c^2)^2}\left(\gamma + c^4\gamma + 2c^2\gamma\right) = \lambda_1\lambda_n. \tag{2.26}$$

By Lemma 2.2.3 and Proposition 1.1.4, we get

$$\lim_k \alpha_k^Y = \frac{2}{\lambda_1 + \lambda_n + \sqrt{(\lambda_1 + \lambda_n)^2 - 4\lambda_1\lambda_n}} = \frac{1}{\lambda_1}.$$

□

It trivially follows from Proposition 2.2.4 that, under Assumptions 1-2, the largest and the smallest eigenvalues of A can be approximated through α_k^Y, $\widetilde{\alpha}_k$ and ρ_k. More precisely,

$$\lim_k \frac{1}{\alpha_k^Y} = \lambda_1, \qquad \lim_k \rho_k \alpha_k^Y = \lim_k \frac{1}{\widetilde{\alpha}_k} - \frac{1}{\alpha_k^Y} = \lambda_n. \tag{2.27}$$

In other words, Propositions 2.2.2 and 2.2.4 show that the SD method asymptotically reveals some second order information, which can be conveniently exploited to speed up the convergence of the classical SD method. In fact, as observed in the Chapther 1, for any gradient method if at the k-th iteration $\alpha_k = 1/\lambda_i$ for some i, then the component of the gradient along the eigenvector d_i of A will be zero at all subsequent iterations. In particular, if we take the steplength $\alpha_k = 1/\lambda_1$, the component of the gradient along the corresponding eigenvector of A will be eliminated. Furthermore, a decrease in the objective function is guaranteed by this choice of α_k, since

$$\frac{1}{\lambda_1} \leq \alpha_k^{SD}.$$

Again, computing the exact value of λ_1 is unrealistic, but we have a way to approximate it, as a matter of fact Formula (2.27) suggests how to get an estimation of it.

Relying on this theoretical analysis, we are now able to suggest a new algorithmic strategy which can be adopted to use the speeding effect provided by the two steplengths $\widetilde{\alpha}_k$ and α_k^Y.
Our approach uses a finite sequence of Cauchy steps followed by a finite sequence of constant steps computed through Formulas (2.19), or (2.20). It is worth noting that the proposed strategy is in line with the consideration that *a good gradient method would use at least one exact line search (the Cauchy step) in every few iterations* by Yuan, [67]. Since Proposition 1.1.2 shows that in the SD method

$$g_k = \mu_1^k d_1 + \mu_n^k d_n + \zeta_k, \tag{2.28}$$

with ζ_k going to zero faster than $\mu_1^k d_1 + \mu_n^k d_n$, our approach is based on the idea of using sequences of Cauchy steps to force the search in a two dimensional space and, at the same time, supply a suitable approximation of one of the two steplengths $\widetilde{\alpha}_k$, and α_k^Y, to be used in aligning the search direction with d_n. The use of a finite sequence of Cauchy steps has therefore a twofold goal: forcing the search in a two-dimensional space and getting a suitable approximation of $1/(\lambda_1 + \lambda_n)$ through $\widetilde{\alpha}_k$, or of $1/\lambda_1$ through α_k^Y. Once this *good* approximation is obtained, the steplength providing such approximation is used, with the aim of driving toward

zero the component μ_1^k of the gradient along d_1. Note that the steplength $\widetilde{\alpha}_k$ approximates the quantity $(\lambda_1 + \lambda_n)^{-1}$, which gets closer and closer to $1/\lambda_1$ as the condition number of A increases, therefore we expect that the two methods will behave similarly as the condition number of the Hessian increases. Of course, μ_1^k cannot generally vanish, but it follows from (1.13) that if the approximation of $1/\lambda_1$ is accurate enough, then taking the same value of $\widetilde{\alpha}_k$ or α_k^Y for multiple steps can significantly reduce μ_1^k.
We also note that, in the ideal case where the component along d_1 is completely removed, the quadratic problem reduces to a $(n-1)$-dimensional problem and a new sequence of Cauchy steps followed by some steps with a constant value of $\widetilde{\alpha}_k$, or α_k^Y, can drive toward zero the component along the eigenvector d_2. For these reasons, a cyclic alternation of SD steplengths and constant steplengths can be performed with the aim of eliminating the components of the gradient, according to the decreasing order of the eigenvalues of A, producing the desired alignment with d_n. In other words, our strategy is aimed at reducing the search in subspaces of smaller and smaller dimensions, and forcing the gradient method to deal with problems with better and better condition numbers. The use of SD steplengths should also help in reducing the components of the gradient that are not addressed by the current constant steplength.
The above strategy for the choice of the steplength can be formalized giving birth to two new gradient methods, the Steepest Descent with Alignment (SDA) method, and the Steepest Descent with Constant steplengths (SDC) method, as follows:

$$\alpha_k^{SDA} = \begin{cases} \alpha_k^{SD} & \text{if } \mathrm{mod}(k, h_1) < h, \\ \widetilde{\alpha}_s & \text{otherwise, with } s = \max\{i \le k \,:\, \mathrm{mod}(i, h_1) = h\}; \end{cases} \tag{2.29}$$

and

$$\alpha_k^{SDC} = \begin{cases} \alpha_k^{SD} & \text{if } \mathrm{mod}(k, h_1) < h, \\ \alpha_s^{Y} & \text{otherwise, with } s = \max\{i \le k \,:\, \mathrm{mod}(i, h_1) = h\}, \end{cases} \tag{2.30}$$

where $h_1 > h \geq 2$. In both of the methods, SD steplengths are alternated with constant ones, computed through formulas (2.19), and (2.20), respectively. In other words, we make h consecutive exact line searches and then, using the last two SD steplengths, we compute the steplength (2.19) or the Yuan steplength, to be applied in $m = h_1 - h$ consecutive gradient iterations. It is clear that the two parameters h and m play complementary roles. Large values of h, which provide more accurate approximations of $1/(\lambda_1 + \lambda_n)$ and $1/\lambda_1$, are likely to work well with small values of m. Conversely, rough approximations of $1/(\lambda_1 + \lambda_n)$ and $1/\lambda_1$, due to small values of h, should be balanced by large values of m.

It is worth observing that the steplength (2.30) generally differs from the one proposed in [23], since in the latter case the Yuan steplength (2.20) is recomputed at each iteration. In particular, when $h \geq 2$, only the first of the m consecutive Yuan steplengths is computed after two exact line searches, and hence these m steplengths cannot be regarded as terms of a sequence for which Proposition 2.2.4 holds. This is the case of the choice $h = 2$ and $h_1 = 4$ (i.e., $m = 2$), of the DY method (1.35), suggested in [23] as the most efficient one.

The main drawback of a gradient method with steplengths (2.29-2.30), henceforth called SDA and SDC, is the non-monotonicity, which is more likely to show up when small values of h and/or large values of m are adopted. Therefore, some strategy aimed to ensure convergence has to be applied, e.g., we may impose that the steplength does not exceed $2\alpha_k^{SD}$. The resulting methods, called SDA and SDC with Monotonicity (SDAM - SDCM), use a steplength obtained from (2.29-2.30) by substituting $\widetilde{\alpha}_s$ and α_s^Y with

$$\min\left\{\widetilde{\alpha}_s,\, 2\alpha_k^{SD}\right\}, \text{ or } \min\left\{\alpha_s^Y,\, 2\alpha_k^{SD}\right\}.$$

Note that the SDAM and SDCM methods can be seen as a special case of the relaxed steepest descent method in [60].

To illustrate the behaviour of the SDA and SDC methods, and of their monotone versions, SDAM and SDCM, we use a test problem with 1000 variables and A diagonal, defined as follows:

$$A_{ii} = \frac{1}{i\sqrt{i}}, \quad b_i = 0. \tag{2.31}$$

The starting point x_0 is such that

$$Ax_0 = e, \quad e = (1, 1, \ldots, 1)^T,$$

and the following stopping condition is used:

$$\|g_k\| < tol\, \|g_0\|, \tag{2.32}$$

where $tol = 10^{-3}, 10^{-6}, 10^{-9}, 10^{-12}$.
The experiments are performed by using MATLAB. We notice that the SD method takes 5954 iterations to satisfy condition (2.32) with $tol = 10^{-3}$.

	SDA (h,m)									DY
tol	(2,2)	(2,4)	(2,6)	(8,2)	(8,4)	(8,6)	(16,2)	(16,4)	(16,6)	
10^{-3}	823 (8)	609 (39)	665 (82)	1852	825 (7)	841 (18)	951	881	831	824
10^{-6}	1907 (22)	1293 (80)	1227 (142)	3399	1658 (13)	1373 (32)	1675	1543	1409 (3)	1999
10^{-9}	2867 (27)	1970 (124)	1619 (188)	5896	2185 (15)	2073 (54)	2195	2097	2050 (3)	2541
10^{-12}	3945 (33)	2728 (185)	2249 (226)	7922	2788 (22)	2457 (59)	3029	2803	2443 (3)	3565

Table 2.1. Problem 2.31, number of iterations of the SDC1 and DY methods. Number of nonmonotone SDA steps reported in brackets.

	SDAM (h,m)								
tol	(2,2)	(2,4)	(2,6)	(8,2)	(8,4)	(8,6)	(16,2)	(16,4)	(16,6)
10^{-3}	1035	1165	633	1852	769	872	951	881	831
10^{-6}	2403	1862	1207	3399	1265	1333	1675	1543	1549
10^{-9}	3431	2589	2084	5896	1833	1750	2195	2097	2476
10^{-12}	4886	3257	2882	7922	2440	2321	3029	2803	2773

Table 2.2. Problem 2.31, number of iterations of the SDAM method.

	SDC (h,m)									DY
tol	(2,2)	(2,4)	(2,6)	(8,2)	(8,4)	(8,6)	(16,2)	(16,4)	(16,6)	
10^{-3}	764 (11)	544 (53)	500 (102)	880	629 (6)	584 (2)	1155	823 (2)	809 (5)	824
10^{-6}	1518 (23)	1131 (98)	899 (162)	1472	1090 (12)	1248 (20)	1782	1353 (2)	1036 (9)	1999
10^{-9}	1854 (32)	1600 (152)	1346 (220)	2527	1514 (16)	1767 (36)	2394	1762 (3)	1541 (13)	2541
10^{-12}	2440 (39)	1997 (180)	1644 (264)	2870	2092 (21)	2049 (40)	2880	2109 (3)	2100 (20)	3565

Table 2.3. Problem 2.31, number of iterations of the SDC and DY methods. Number of nonmonotone SDC steps reported in brackets.

	SDCM (h,m)								
tol	(2,2)	(2,4)	(2,6)	(8,2)	(8,4)	(8,6)	(16,2)	(16,4)	(16,6)
10^{-3}	1040	592	580	880	634	506	1155	852	685
10^{-6}	1276	1080	1054	1472	1150	1026	1782	1250	1250
10^{-9}	1952	1754	1468	2527	1690	1452	2394	1782	1632
10^{-12}	2402	2180	1962	2870	2146	1970	2880	2230	2224

Table 2.4. Problem 2.31, number of iterations of the SDCM method.

In Figure 2.4 we plot, on a log scale, the values of the sequences $\left\{\left|\widetilde{\alpha}_k - \frac{1}{(\lambda_1+\lambda_n)}\right|\right\}$, $\left\{\left|\alpha_k^Y - \frac{1}{\lambda_1}\right|\right\}$, and $\{\|g_k\|\}$, for $k = 1, \ldots, 100$, against the number of iterations, computed by using the Cauchy steplengths resulting from the application of the SD

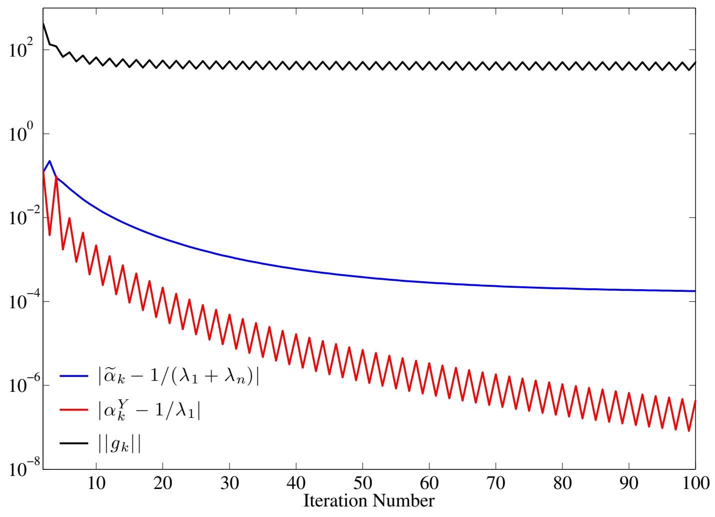


Figure 2.4. Problem 2.31, behaviour of the sequences $\left\{\left|\widetilde{\alpha}_k - \frac{1}{\lambda_1+\lambda_n}\right|\right\}$, $\left\{\left|\alpha_k^Y - \frac{1}{\lambda_1}\right|\right\}$, and $\{\|g_k\|\}$ for for the first 100 iterations of the SD method.

method, where the stop condition $\|g_k\| < 10^{-5}\|g_0\|$ has been considered. Although the SD performs very poorly, a quite accurate approximation of $1/(\lambda_1+\lambda_n)$ and $1/\lambda_1$ is achieved after few iterations. In Tables 2.1-2.4 we report the results obtained by running the SDA and SDC methods, and their monotone versions SDAM and SDCM with 9 possible choices for the pair (h, m), obtained by varying h and m in $\{2, 8, 16\}$ and $\{2, 4, 6\}$, respectively. These tests are aimed at understanding whether and how h and m affect the methods, in terms of number of iterations and spectral properties. For each run we show the overall number of iterations and, for SDA and SDC, the number of nonmonotone steps, i.e., the steps where the objective function increases. For comparison purposes, in Table 2.1 and Table 2.3 we also report the number of iterations required by the Dai-Yuan (DY) method using the scheme reported in (1.35). We see that the SDA and the SDC methods never fails, despite no strategy is adopted to ensure global convergence. Although the role of h and m is not negligible in the numerical behaviour of the methods, both of them appear very competitive with the DY method, regardless of the choice of the parameters. For the smallest value of h, the performance strongly improves as m increases, while this tendency is less evident for larger values of h. Making several consecutive exact linesearches (for instance, $h = 16$) fosters monotonicity, but the overall number of iterations might tend to increase, due to the slow convergence of the SD method. Conversely, the monotonicity of the method deteriorates as m grows, especially when few SD steps are performed.

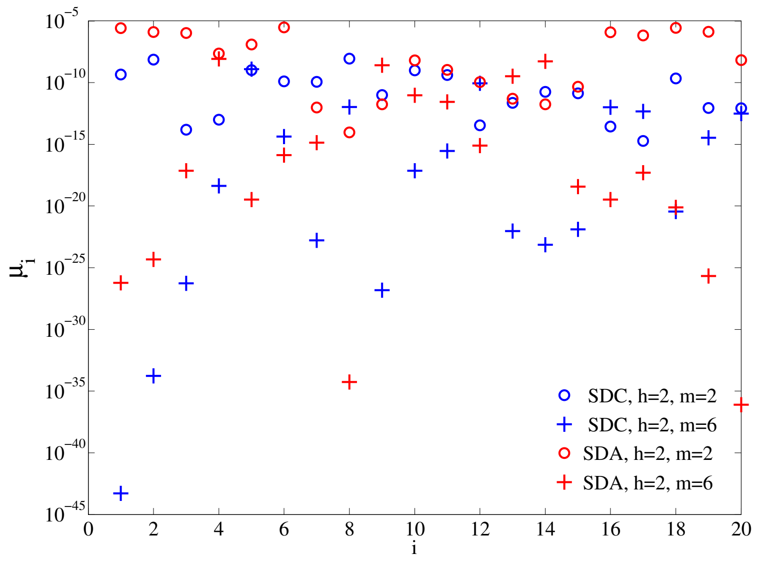
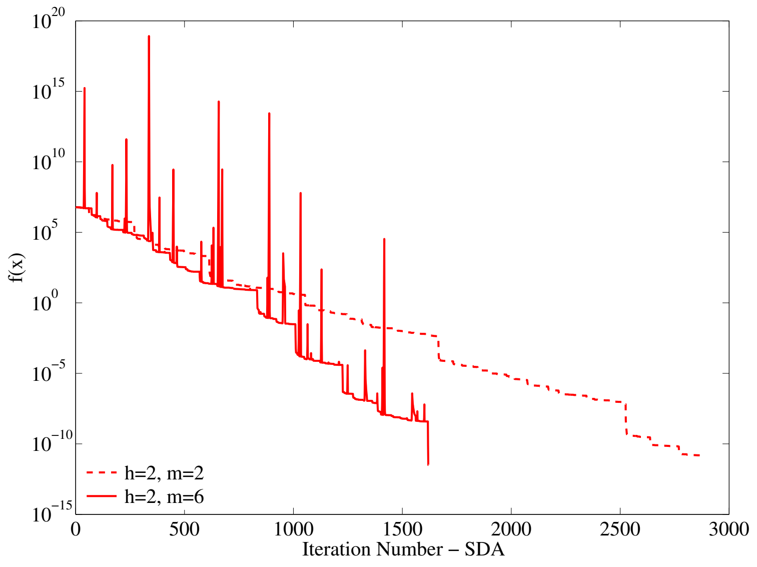


Figure 2.5. Problem 2.31, values of the eigencomponents μ_i^k $(i = 1, \dots, 20)$ of the gradient at the solution computed by the SDA and SDC methods, for $h = 2$ and $m = 2$ and for $h = 2$ and $m = 6$.

To gain a further insight into the behaviour of the SDC method, we also analyse how the eigencomponents of the gradient are affected by m. In Figure 2.5, we plot the values of the first 20 eigencomponents of the gradient at the solution computed by the SDA and SDC methods, with $tol = 10^{-9}$, for $h = 2$ and $m = 2, 6$ (the smallest value of h is considered to better highlight the effects produced by different values of m). The two methods behave similarly, but the better approximation of $1/\lambda_1$ obtained with the SDC method is clearly visible in terms of the eigencomponents reduction. We can also notice that the order of magnitude of the eigencomponents is smaller for $m = 6$, validating the role played by multiple constant steplengths in driving toward zero the eigencomponents corresponding to the largest eigenvalues.

Finally, in Figures 2.6-2.7 we compare the convergence histories of the objective function in the SDA and SDC methods and their monotone versions, with $tol = 10^{-9}$, for $h = 2$ and $m = 2$ and for $h = 2$ and $m = 6$.
The first consideration which can be done is that the oscillating behaviour of the SDA and SDC methods for $m = 6$ clearly appears, as well as their faster convergence with respect to the SDAM and SDCM methods for both values of m. In addition, it is confirmed that the SDC (and SDCM) method behaves better than SDA (and SDAM) method, the advantage of saving around 500 iterations is in fact due to the better approximation of $1/\lambda_i$ cyclically reached as the SDC (and SDCM) method goes on, as already observed in the above considerations. The figures also



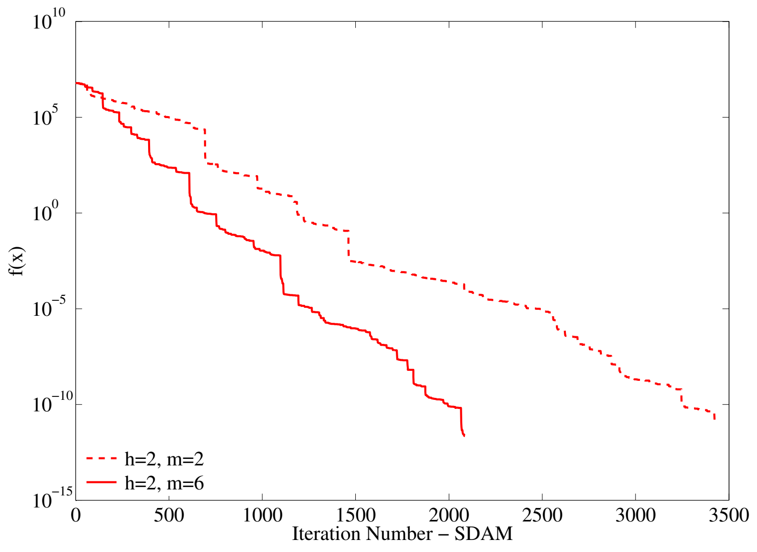


Figure 2.6. Problem 2.31, convergence history of $\{f(x_k)\}$ in the SDA and SDAM methods (top and bottom, respectively), for $h = 2$ and $m = 2$ and for $h = 2$ and $m = 6$.

confirms that imposing monotonicity does not yield a severe deterioration of convergence. This is an important point which will be further exploited in the Numerical experiments section.

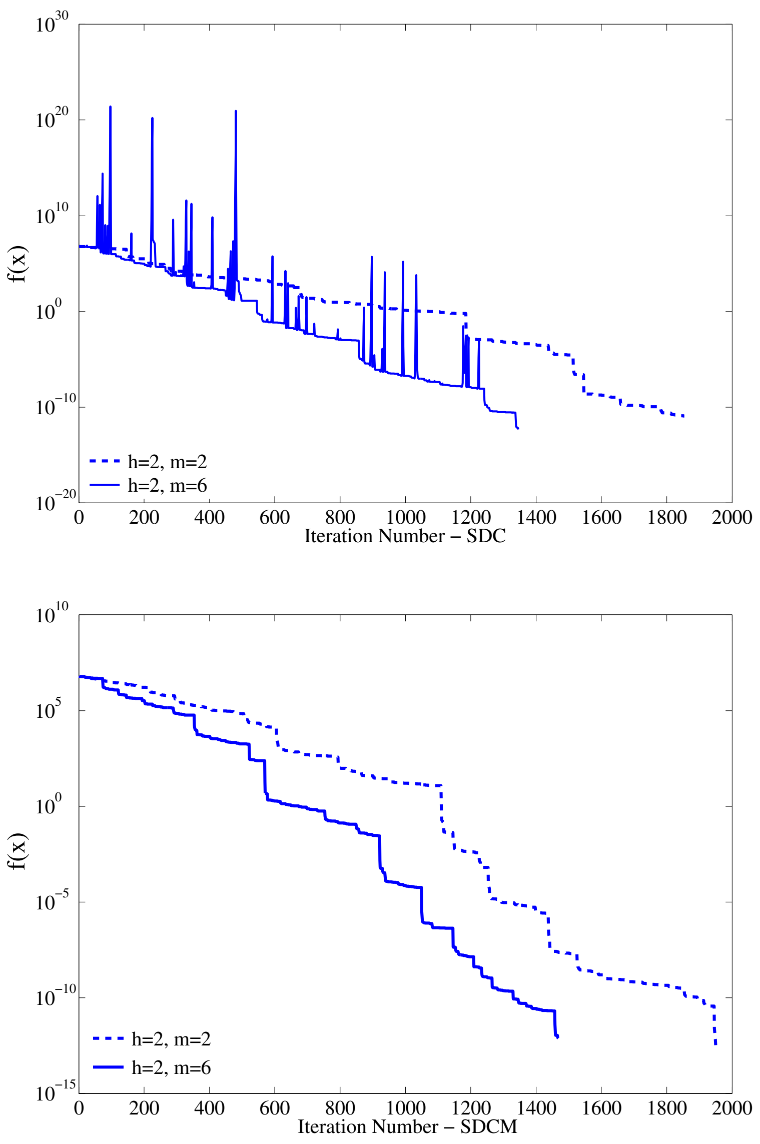


Figure 2.7. Problem 2.31, convergence history of $\{f(x_k)\}$ in the SDC and SDCM methods (top and bottom, respectively), for $h = 2$ and $m = 2$ and for $h = 2$ and $m = 6$.

2.3 A dynamic adaptive technique

In [26], De Asmundis et al. proposed a novel adaptive strategy for the SDA method, henceforth called SDA_ADP. As we saw in Section 2.2, Proposition 2.2.1 suggests that, for k sufficiently large,

$$\widetilde{\alpha}_k = \left(\frac{1}{\alpha_k^{SD}} + \frac{1}{\alpha_{k+1}^{SD}} \right)^{-1} \tag{2.33}$$

can be used as an approximate value for

$$\widehat{\alpha} = \frac{1}{\lambda_1 + \lambda_n}. \tag{2.34}$$

Moreover, we know that we can get a suitable approximation of (2.34) by mean of sequences of Cauchy steps α_k^{SD}. The idea of the *dynamic adaptive step selection technique* is to avoid fixing the number of SD iterations (h in the SDA scheme 2.29), but to dynamically determine it using a so-called *switch condition.* More precisely, the *switch condition* monitors when the sequence $\{\widetilde{\alpha}_k\}$ settles down, checking if the absolute value of the differences between two consecutive computed $\widetilde{\alpha}_k$ is less than an *a priori* fixed tolerance ε. SDA_ADP thus performs m consecutive iterations using the last computed $\widetilde{\alpha}_k$ as steplength, if it produces a decrease in the objective function (otherwise, SDA_ADP adopts a double Cauchy step). In this way the *switch condition* decides if the method can switch from SD iterates to constant iterates.
The main difference between the SDA and the SDA_ADP methods is the monotone behavior. The SDA_ADP method is, in fact, forced to be monotone, taking a double Cauchy step in absence of a decrease in the objective function. When a double Cauchy step is adopted, the method restarts with a sequence of SD iterates until the *switch condition* is again satisfied. Despite their differences, both of the methods share the idea of using a finite sequence of Cauchy steps followed by a finite sequence of constant steps computed through Formula (2.33). Again a sequence of constant *special* steps can emphasise the alignment effect, thus speeding up the convergence.

To play a fair game, we also consider the adaptive version of the SDC method, named henceforth the SDC_ADP method. Following the same scheme of the SDA_ADP method, we use the *switch condition* to dynamically skip from SD iterates to a fixed number of constant iterates computed through the Yuan formula (2.20). We restart the method each time a double Cauchy step is adopted as a consequence of a nonmonotone behaviour.

In Tables 2.5-2.6 we report the results obtained by running the SDA_ADP and SDC_ADP methods on Problem 2.31, with 10 possible choices for the parameter m, from 1 to 10. For each run we show the overall number of iterations, while the number of times a double Cauchy step is adopted is reported in brackets. For comparison purposes, we also report the number of iterations required by the DY method using the scheme reported in (1.35). Here again we want to understand how the choice of m can affect the performances of the method, in terms of number

tol	SDA_ADP (m)										DY
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	
10^{-3}	1466	2002	1332	1067 (2)	754 (1)	813 (4)	826 (5)	892 (7)	586 (5)	633 (4)	824
10^{-6}	2689	3685	2638 (1)	1947 (5)	1615 (2)	1603 (9)	1492 (7)	1265 (8)	1122 (8)	1137 (7)	1999
10^{-9}	3359	6156	3825 (1)	2454 (6)	2030 (3)	2374 (14)	1871 (10)	1902 (14)	1684 (11)	1982 (16)	2541
10^{-12}	4298	9027	4555 (1)	3119 (6)	2730 (5)	2857 (17)	2452 (14)	2358 (18)	2003 (13)	2560 (18)	3565

Table 2.5. Problem 2.31, number of iterations of the SDA_ADP method. Number of double Cauchy steps reported in brackets.

tol	SDC_ADP (m)										DY
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	
10^{-3}	1293	1199	661	757	657 (1)	516 (1)	600 (1)	490 (1)	588 (2)	541 (1)	824
10^{-6}	3564	1957	1217 (1)	1330 (1)	1255 (4)	930 (2)	1102 (4)	954 (2)	1016 (5)	909 (3)	1999
10^{-9}	5082	2972	1660 (2)	2169 (2)	1572 (4)	1380 (4)	1740 (6)	1439 (4)	1389 (7)	1227 (4)	2541
10^{-12}	6328	4278	2147 (2)	2566 (2)	2071 (6)	1866 (6)	2187 (7)	1899 (9)	1774 (9)	1586 (6)	3565

Table 2.6. Problem 2.31, number of iterations of the SDC_ADP method. Number of double Cauchy steps reported in brackets.

of iterations and spectral properties. The two methods appear to behave quite differently, being the SDC_ADP faster and smoother as the parameter m increases. Furthermore, the number of times in which a double Cauchy step is adopted is appreciably lower in the SDC_ADP method. The role of m is not negligible in the numerical behaviour of both of the methods, but in the SDA_ADP method it appears to be a much more critical issue. While the SDC_ADP method becomes more competitive with DY as the value of m increases, the SDA_ADAP gives the worst performances for $m = 2$, then its performance gets better, but still we have some oscillation, e.g. going from $m = 9$ to $m = 10$. However, for both of the methods, making several constant steps, allowing bigger values of m, increases the number of iterations in which a double Cauchy step is adopted, decreasing the overall number of iterations. In [26] the authors use $m = 5$, this value is a good trade-off between speed of convergence, in terms of iteration number, and number of iterations in which the double Cauchy step is adopted.
Other tests on SDA_ADP and SDC_ADP will be carried on in the Numerical experiments section.

2.4 Numerical Experiments

In this section we compare the performances of the methods seen in the Sections 2.1, 2.2, and 2.3. More precisely, we first analyse the numerical behaviour of the RSDA method, comparing it to its natural opponent, the RSD method [60], and to the BB method [4], to have an additional valid comparison. At this point, we move our attention through the SDA and SDC methods, and their monotone and adaptive versions, comparing them all with their most competitive opponent, the DY method [23], as already partially done during their theoretical descriptions in the previous sections of this chapter.

We first considered two sets of test problems of type (1.1), with A diagonal, $b = 0$ and dimension $n = 10^4$. For each set we defined

$$A_{11} = \xi, \quad A_{nn} = 1,$$

with $\xi = 10^4, 10^5, 10^6$, and built the remaining diagonal entries of A so that $\kappa(A) = \xi$. In the first set of problems, referred to as RAND, the diagonal matrix entries A_{jj}, with $j = 2, \ldots, n-1$, were randomly generated in $[A_{11},\ A_{nn}]$, while in the second set, referred to as NONRAND, they were set as follows:

$$A_{jj} = 10^{\frac{ncond}{n-1}(n-j)},$$

with $ncond = \log_{10}\xi = \log_{10}\kappa(A)$. For each problem, 10 starting points were randomly generated with entries in $[-5, 5]$.
We note that the RAND problems, as well as the starting points, are those used in [16] to compare different gradient methods (actually, larger condition numbers are considered here). However, we also decided to use the NONRAND problems in order to test our methods on a class of quadratic problems on which the SD method exhibits very slow convergence (by running the SD method on an instance of the RAND and NONRAND problems with condition number $\kappa(A) = 10^6$, we got, for $tol = 10^{-6}$, 3321 iterations in the first case, and more than 100,000 iterations in the second case). The stopping criterion (2.32) was used by all the methods, with $tol = 10^{-6}, 10^{-9}, 10^{-12}$; a maximum number of 25,000 iterations was also set, but it was never reached in our experiments.
All the methods were implemented in MATLAB (v. 7.12.0.635 - R2011a). The random diagonal entries of the matrix A in the first set of problems, as well as the starting points, were generated by using the MATLAB `rand` function.

In Table 2.7 we report the number of iterations on each RAND and NONRAND problem, averaged over the 10 runs with different starting points[3], obtained by running the RSDA, RSD and BB methods. The last row is obtained by adding

[3]Because of the randomness in the methods, each problem was run 10 times.

up the iterations performed on all the problems. Although the superiority of the RSDA method over the RSD method is clearly visible, the performance of the two algorithms deteriorates as the tolerance on the stop condition increases, while the condition number doesn't seems to be crucial. The BB method gives the overall best results, but we don't feel confident in saying that the RSDA method fails, in fact it is impressive how a small theoretical modification can be translated in such a practical gain.

(a) RAND problems

Problem		Methods		
$\kappa(A)$	*tol*	BB	RSD	RSDA
10^4	1e-06	259	451	313
10^4	1e-09	882	1707	1091
10^4	1e-12	1323	2723	1803
10^5	1e-06	244	486	335
10^5	1e-09	2238	3928	2691
10^5	1e-12	4256	7634	5014
10^6	1e-06	222	413	304
10^6	1e-09	4574	7156	5364
10^6	1e-12	10768	19566	12848
Total iters		24766	44064	29763

(b) NONRAND problems

Problem		Methods		
$\kappa(A)$	*tol*	BB	RSD	RSDA
10^4	1e-06	622	1136	765
10^4	1e-09	1221	2172	1481
10^4	1e-12	1664	3407	2179
10^5	1e-06	1509	2694	1789
10^5	1e-09	3519	6274	4213
10^5	1e-12	5684	9915	6373
10^6	1e-06	2712	4426	3045
10^6	1e-09	10228	17493	11079
10^6	1e-12	17378	28105	18789
Total iters		44537	75622	49713

Table 2.7. RAND and NONRAND problems, number of iterations of the BB, RSD and RSDA methods.

In Tables 2.8, 2.9, 2.10, and 2.11 we report the number of iterations performed by the SDA, SDC and DY methods on each RAND and NONRAND problem, averaged over the 10 runs with different starting points. We decided not to run other gradient methods, since their performances are usually not superior than the one of the DY method [23, 26, 24]. Here again, the last row in each table is obtained by adding up the iterations performed on all the problems. The SDA, SDC, SDAM, and SDCM methods were run with different values of h and m, i.e., all the combinations of $h = 10, 20, 30, 40, 50$ and $m = 2, 4, 8$. Here we neglected smaller values of h to avoid too strong non-monotonicity, but this issue will be discussed in the following pages.

As for the problem considered in Section 2.2, the SDA and SDC method never fail; actually, they exhibit a quite surprising monotonic behaviour for all the selected combinations of h and m (i.e., the SDA and SDC methods coincide with their monotone versions, SDAM and SDCM, respectively), except $(10, 8)$. In this latter case, the number of SDAM and SDCM iterations, reported in brackets, is

comparable with the number of SDA or SDC iterations. As expected, the RAND problems are easier to solve than the NONRAND ones. However, the SDA and SDC methods do not perform as bad as the SD method on the NONRAND problems, and actually the number of total iterations increases by at most a factor of 2.2 when moving from the RAND to the NONRAND problems. For the SDA and SDC methods, the overall worst performance occurs for $h = 10$, which is likely to produce a steplength $\widetilde{\alpha}_k$, or α_k^Y, providing an approximation of $1/(\lambda_1+\lambda_n)$, or $1/\lambda_1$, not enough accurate. In this case the choice of m appears crucial, as we can see by comparing the results for $m = 2, 4$ with those for $m = 8$. The largest value of m is able to make up for the effects of using a small value of h, though producing non-monotonicity. For the other values of h, $m = 8$ is often less favorable than the other values of m, especially if high accuracy in the solution is required. This is in line with the previous observation that a large value of m is unnecessary, or even counterproductive, when a sufficiently large value of h is selected. For the SDC method, our conjecture is that once a Yuan steplength has been able to eliminate a gradient eigencomponent, a further use of such steplength (which is likely to be smaller than the values of $1/\lambda_i$ corresponding to the remaining eigencomponents) can only slow down the method. For $h \geq 20$, setting $m = 2, 4$ leads to comparable results, with differences of less than 10% in the total number of iterations; using $m = 8$ generally does not pay, especially for the NONRAND problems. The largest value of h generally becomes more effective on the most ill-conditioned problems when high accuracy is required in the solution. On the other hand, when the condition number is not too large and the accuracy requirement is not too high, smaller values of h seem to be preferable. We believe that, when the problems are *easy* (i.e., they just require a few hundred iterations), a large value of h is unnecessary to get a suitable steplength, and actually it tends to increase the overall number of iterations, doing too many SD iterates.
In order to support this argument, we define a *sweep* as the sequence of h SD steps followed by m gradient steps with constant steplengths, and analyse the number of sweeps required by the SDA and SDC methods with different values of h. For instance, on the RAND problem with $\kappa(A) = 10^6$ and $\texttt{tol} = 10^{-6}$, SDA performs, on average, 4.7 sweeps for $h = 50$ and $m = 4$, and 7.1 sweeps for $h = 30$ and $m = 4$, while SDC performs, on average, 5.8 sweeps for $h = 50$ and $m = 4$, and 7.3 sweeps for $h = 30$ and $m = 4$. However, the number of SDA and SDC iterations is larger in the first case (255 and 315 vs 243 and 249). In other words, increasing h increases the number of iterations despite the reduction of the number of sweeps.

Concerning the comparison between the DY and the SDA and SDC methods, we note that SDA and SDC performs worse than DY when the lowest accuracy is required. In the remaining cases, the SDA and SDC methods tend to outperform the DY method for $h \geq 20$, with a saving in the number of iterations which increases significantly with the condition number and the accuracy requirement. This is a remarkable result, since the literature shows that, among the gradient methods, the DY method exhibits the best overall numerical performance [16, 26]. It is a fact, though, that in our experiments reported in Table 2.7, both in the RAND and NONRAND problems with $\kappa(A) = 10^6$ and $\texttt{tol} = 10^{-12}$, even the BB and the RSDA methods perform better that the DY method, saving some iterations, while in all the other cases, the DY method is superior, as it is supposed to be.

Since the effectiveness of the BB methods has been related to their nonmonotone behaviour [32], we wondered if non-monotonicity also plays an important role in determining the nice behaviour of the SDA and SDC methods. Considering, therefore, values of m bigger than h, next tables show the numerical results taking into consideration the couples (h, m) in which $m = 6, 18, 30$. Here we allowed smaller values of h (i.e. $h = 2, 4$) to underline how the strong non-monotonicity can influence the performance of the methods. For comparison purposes, and to make the tables easy to be read, the DY method was again considered.
In Tables 2.12 and 2.13 we report the number of iterations performed by the SDA, SDC and DY methods on each RAND and NONRAND problem imposing $h = 2, 4$ and $m = 6$. The first thing to note is precisely the nonmonotone behaviour of the SDA and SDC methods, and the fact that this nonmonotonicity works, even comparing these results with the ones in the previous tables, especially when the accuracy requirement is not too high. The corresponding results for the SDAM and SDCM methods are not competitive, with few exceptions, e.g., for $h = 4$ and $m = 6$ imposing monotonicity brings a gain in terms of number of iterations when the accuracy requirement is 10^{-12} for both the SDA and SDC methods. In Table 2.13 we can see that, even though the improvement brought by the nonmonotonicity is less visible, still the performances deteriorate by running the SDAM and SDCM methods.

Going on with our experimental analysis, we now try to increase the value of m, considering $m = 18$ and $m = 30$. The first set of tables, 2.14, 2.15 for the RANDOM problems, and 2.16, 2.17 for the NONRAND problems, are relative to $m = 18$.
At a first sight, it can be noted how the SDA-SDAM and SDC-SDCM methods behave different if applied to the RAND or NONRAND problems. This is also the trend in Tables 2.18, 2.19 and Tables 2.20, 2.21 where we tested the methods using $m = 30$. Especially for the RAND problems, we can see how the combination of

Problem		SDA (h,m)															DY
$\kappa(A)$	*tol*	(10,2)	(10,4)	(10,8)	(20,2)	(20,4)	(20,8)	(30,2)	(30,4)	(30,8)	(40,2)	(40,4)	(40,8)	(50,2)	(50,4)	(50,8)	
10^4	1e-06	217	239	219	238	240	228	259	248	247	266	269	257	300	269	276	222
10^4	1e-09	757	857	709 (720)	729	722	812	719	737	731	757	753	757	773	772	759	683
10^4	1e-12	1244	1387	1114 (1189)	1149	1193	1196	1190	1123	1135	1126	1095	1172	1150	1152	1118	1140
10^5	1e-06	232	256	235	248	231	234	256	244	251	265	279	247	298	282	276	228
10^5	1e-09	2285	2552	1860 (1967)	1596	1579	1682	1711	1486	1534	1654	1539	1718	1613	1578	1672	1839
10^5	1e-12	4833	4810	3459 (3421)	3019	3012	3442	3200	2672	2920	2687	2752	2975	2774	2509	2928	3469
10^6	1e-06	217	231	212 (216)	223	217	221	235	243	239	267	244	240	274	255	258	204
10^6	1e-09	6033	6865	4400 (3469)	4015	2975	4222	3153	3101	3640	2443	2612	3383	2947	2399	2862	4283
10^6	1e-12	15212	19735	9884 (8717)	8665	7518	10275	7260	6548	8751	5164	5414	7605	5852	5370	6513	13231
Total iters		31030	36932	22092 (20153)	19882	17687	22312	17983	16402	19448	14629	14957	18354	15981	14586	16662	25299

Table 2.8. RAND problems: mean number of iterations of the SDA and DY methods. SDAM iterations in brackets if different from the SDA one.

Problem		SDC (h,m)															DY
$\kappa(A)$	*tol*	(10,2)	(10,4)	(10,8)	(20,2)	(20,4)	(20,8)	(30,2)	(30,4)	(30,8)	(40,2)	(40,4)	(40,8)	(50,2)	(50,4)	(50,8)	
10^4	10^{-6}	242	229	235	238	234	232	260	272	253	270	309	268	280	341	291	222
10^4	10^{-9}	840	882	678 (698)	772	658	707	740	690	719	779	734	726	894	779	855	683
10^4	10^{-12}	1324	1407	1186 (1206)	1181	1105	1092	1147	1069	1135	1197	1141	1164	1224	1122	1256	1140
10^5	10^{-6}	255	255	225	246	243	229	297	262	253	267	284	253	300	335	290	228
10^5	10^{-9}	2209	2742	1872 (1767)	1742	1666	1619	1555	1448	1775	1576	1408	1665	1507	1398	1606	1839
10^5	10^{-12}	4421	5168	3279 (3279)	2861	2739	3050	2816	2563	3025	2319	2744	2846	2607	2633	2774	3469
10^6	10^{-6}	227	221	204	224	230	221	245	249	227	239	281	255	244	315	277	204
10^6	10^{-9}	4999	6845	3587 (3309)	2576	2415	5118	2477	2897	3583	2614	2914	3401	2461	2520	4079	4283
10^6	10^{-12}	13274	18219	8099 (7735)	5869	7620	10866	5668	7168	8774	5690	6615	8671	5345	4944	7165	13231
Total iters		27791	35968	19365 (18658)	15709	16910	23134	15205	16618	19744	14951	16430	19249	14862	14387	18593	25299

Table 2.9. RAND problems: mean number of iterations of the SDC and DY methods. SDCM iterations in brackets if different from the SDC one.

Problem		SDA (h,m)															DY
$\kappa(A)$	*tol*	(10,2)	(10,4)	(10,8)	(20,2)	(20,4)	(20,8)	(30,2)	(30,4)	(30,8)	(40,2)	(40,4)	(40,8)	(50,2)	(50,4)	(50,8)	
10^4	1e-06	632	591	532 (538)	537	516	542	541	526	566	591	592	596	601	586	603	525
10^4	1e-09	1158	1138	1038 (1041)	1006	985	1011	1000	979	1022	1022	1015	1014	1060	1047	1023	975
10^4	1e-12	1653	1649	1417 (1462)	1454	1369	1436	1454	1369	1421	1490	1423	1440	1545	1460	1437	1453
10^5	1e-06	1473	1527	1281 (1263)	1213	1177	1318	1273	1160	1209	1236	1234	1279	1291	1211	1227	1216
10^5	1e-09	3988	3752	2992 (2879)	2746	2870	3122	2822	2683	2716	2798	2690	2843	2722	2795	2817	3052
10^5	1e-12	6043	6087	4495 (4425)	4225	4039	4924	4095	4024	4183	4205	3978	4299	4227	4043	4132	4686
10^6	1e-06	2743	3017	2203 (2113)	1916	2017	2281	1981	2025	2073	2003	2033	2001	1986	2085	2025	2297
10^6	1e-09	12516	16206	8457 (8090)	7661	7903	8643	7670	6986	8047	7571	7256	7512	7189	7286	7121	10075
10^6	1e-12	22619	28626	14692 (13278)	11758	12642	14255	12835	11315	13086	12152	11874	12850	11604	11656	11480	18824
Total iters		52825	62593	37107 (35089)	32516	33518	37532	33671	31067	34323	33068	32095	33834	32225	32169	31865	43103

Table 2.10. NONRAND problems: mean number of iterations of the SDA and DY methods. SDAM iterations in brackets if different from the SDA one.

Problem		SDC (h,m)															DY
$\kappa(A)$	*tol*	(10,2)	(10,4)	(10,8)	(20,2)	(20,4)	(20,8)	(30,2)	(30,4)	(30,8)	(40,2)	(40,4)	(40,8)	(50,2)	(50,4)	(50,8)	
10^4	10^{-6}	597	559	555 (553)	579	555	515	586	558	548	625	557	551	657	584	580	525
10^4	10^{-9}	1117	1176	1030 (1012)	995	1039	974	1066	999	1030	1086	1008	1001	1122	1051	1063	975
10^4	10^{-12}	1652	1707	1487 (1457)	1409	1413	1468	1443	1410	1441	1561	1451	1464	1545	1448	1495	1453
10^5	10^{-6}	1377	1421	1154 (1171)	1241	1245	1134	1227	1178	1251	1234	1256	1247	1290	1219	1166	1216
10^5	10^{-9}	3686	3795	2965 (3022)	2827	2866	3077	2791	2672	2756	2858	2805	2789	2840	2759	2705	3052
10^5	10^{-12}	5395	6136	4660 (4559)	4326	4303	4935	4128	4116	4131	4226	4193	4316	4186	4117	4149	4686
10^6	10^{-6}	2880	2470	2084 (1961)	2130	2047	1985	2049	1917	2031	2173	1914	1958	2038	2019	2024	2297
10^6	10^{-9}	12224	13754	7972 (8262)	7148	7737	8146	7348	7262	7934	7388	7406	7836	7420	7140	6989	10075
10^6	10^{-12}	21544	24038	13110 (14045)	11945	11873	14356	11656	12212	13938	11786	12148	12149	11482	11991	12224	18824
Total iters		50472	55056	35017 (36042)	32600	33078	36590	32294	32324	35060	32937	32738	33311	32580	32328	32395	43103

Table 2.11. NONRAND problems: mean number of iterations of the SDC and DY methods. SDCM iterations in brackets if different from the SDC one.

Problem		SDA (h, m)		SDC (h, m)		DY
$\kappa(A)$	*tol*	(2,6)	(4,6)	(2,6)	(4,6)	
10^4	1e-06	239 (223)	217	222 (223)	208 (212)	222
10^4	1e-09	766 (813)	653 (680)	647 (787)	682 (691)	683
10^4	1e-12	1176 (1293)	1116 (1068)	1033 (1272)	1113 (1110)	1140
10^5	1e-06	228 (231)	231 (221)	211 (217)	222 (227)	228
10^5	1e-09	1687 (2185)	1706 (1742)	1636 (2031)	1556 (1553)	1839
10^5	1e-12	2945 (3798)	3275 (3375)	2690 (3930)	2731 (2856)	3469
10^6	1e-06	234 (204)	205 (211)	210 (213)	212 (208)	204
10^6	1e-09	4187 (5074)	3726 (3388)	3205 (4657)	3313 (3597)	4283
10^6	1e-12	8489 (12611)	10205 (8594)	6677 (12151)	9195 (7915)	13231
Total iters		19951 (26432)	21334 (19496)	16531 (25481)	19232 (18369)	25299

Table 2.12. RAND problems: mean number of iterations of the SDA, SDC and DY methods. SDAM and SDCM iterations in brackets if different from the SDA and SDC one.

Problem		SDA (h, m)		SDC (h, m)		DY
$\kappa(A)$	*tol*	(2,6)	(4,6)	(2,6)	(4,6)	
10^4	1e-06	513 (521)	483 (502)	493 (513)	495 (483)	525
10^4	1e-09	984 (1062)	925 (912)	932 (984)	945 (925)	975
10^4	1e-12	1391 (1510)	1372 (1367)	1331 (1391)	1325 (1372)	1453
10^5	1e-06	1290 (1275)	1202 (1167)	1115 (1290)	1149 (1202)	1216
10^5	1e-09	3233 (3601)	2758 (2824)	2730 (3233)	2458 (2758)	3052
10^5	1e-12	4817 (5474)	4365 (4288)	4349 (4817)	3951 (4365)	4686
10^6	1e-06	2296 (2399)	2040 (1930)	2025 (2296)	2001 (2040)	2297
10^6	1e-09	8897 (11059)	8478 (8365)	7396 (8897)	7912 (8478)	10075
10^6	1e-12	15771 (19013)	13419 (13247)	12448 (15771)	12748 (13419)	18824
Total iters		39192 (45914)	35042 (34602)	32819 (39192)	32984 (35042)	43103

Table 2.13. NONRAND problems: mean number of iterations of the SDA, SDC and DY methods. SDAM and SDCM iterations in brackets if different from the SDA and SDC one.

small values of h with big values of m seems to work well, even considering the monotone versions of the methods, SDAM and SDCM. This confirms our idea that larger values of m are able to make up for the effects of using a small value of h, paying the price of having an inaccurate approximation of $1/(\lambda_1 + \lambda_n)$ or $1/\lambda_1$, thus producing nonmonotonicity. Looking for the reasons of the good behaviour of the monotone versions, SDAM and SDCM, we monitored the number of times in which the SDAM and SDCM methods adopt a double Cauchy step to avoid nonmonotonicity. In this way, for example, for the RAND problems, assuming $\kappa(A) = 10^6$, tolerance 10^{-12}, considering $h = 4$ (and $m = 18$), we have an average of 1601 double Cauchy steps for the SDAM method and of 1753 double Cauchy steps for the SDCM method over a total average of 7503 and 8086 iterations, respectively. This means that, in this case, the SDAM and SDCM methods spent almost 1/5 of the total iterations to avoid nonmonotonicity, and this correction leads to an increase in the final number of iterations with respect to the corresponding SDA and SDC methods. For $h = 10$ we don't have a big difference among the values of SDA-SDAM and SDC-SDCM, here, always considering $\kappa(A) = 10^6$ and tolerance 10^{-12}, we have an average of 460 double Cauchy steps for the SDAM method, and of 511 double Cauchy steps for the SDCM method over a total average of 6199 and 6078 iterations, respectively. In this way the incidence of the double Cauchy step seems to play a fundamental role in the global performance of the methods. Our conjecture is that the massive use of the double Cauchy step, with its alignment property described in Proposition 2.1.1, can bring an improvement in the overall decrease of the gradient eigencomponents. This effect is less visible for the NONRAND problems, where the SDAM and SDCM methods are comparable with the SDA and SDC methods, respectively.

Problem		SDA (h, m)				DY
$\kappa(A)$	*tol*	(2,18)	(4,18)	(8,18)	(10,18)	
10^4	1e-06	398 (250)	310 (240)	251 (243)	247 (249)	222
10^4	1e-09	1041 (702)	804 (729)	683 (645)	664 (671)	683
10^4	1e-12	1416 (1079)	1230 (1154)	1078 (1043)	1027 (1063)	1140
10^5	1e-06	398 (255)	318 (249)	261 (252)	257 (250)	228
10^5	1e-09	1574 (1426)	1427 (1591)	1405 (1336)	1440 (1502)	1839
10^5	1e-12	2343 (2570)	2556 (2712)	2566 (2433)	2498 (2677)	3469
10^6	1e-06	388 (232)	298 (222)	259 (238)	232 (230)	204
10^6	1e-09	2764 (2382)	2337 (2833)	2635 (2672)	2993 (2966)	4283
10^6	1e-12	4729 (8006)	5102 (7503)	5467 (6229)	6012 (6199)	13231
Total iters		15051 (17802)	14382 (17233)	14605 (15091)	15370 (15807)	25299

Table 2.14. RAND problems: mean number of iterations of the SDA and DY methods. SDAM iterations in brackets if different from the SDA one.

Problem		SDC (h, m)				DY
$\kappa(A)$	*tol*	(2,18)	(4,18)	(8,18)	(10,18)	
10^4	1e-06	289 (251)	297 (242)	249 (241)	248 (237)	222
10^4	1e-09	781 (658)	690 (683)	700 (650)	710 (665)	683
10^4	1e-12	1302 (1037)	1107 (1038)	1061 (1024)	1074 (1035)	1140
10^5	1e-06	404 (252)	291 (262)	243 (248)	233 (234)	228
10^5	1e-09	1529 (1412)	1350 (1731)	1407 (1699)	1407 (1347)	1839
10^5	1e-12	2100 (2708)	2224 (2915)	2302 (2662)	2463 (2399)	3469
10^6	1e-06	312 (244)	339 (236)	227 (222)	235 (228)	204
10^6	1e-09	2357 (3565)	2220 (2903)	2676 (2490)	1981 (2978)	4283
10^6	1e-12	4107 (8293)	4781 (8086)	5234 (5536)	4865 (6078)	13231
Total iters		13181 (18420)	13299 (18096)	14099 (14772)	13216 (15201)	25299

Table 2.15. RAND problems: mean number of iterations of the SDC and DY methods. SDCM iterations in brackets if different from the SDC one.

Problem		SDA (h, m)				DY
$\kappa(A)$	*tol*	(2,18)	(4,18)	(8,18)	(10,18)	
10^4	1e-06	757 (524)	615 (518)	534 (505)	515 (495)	525
10^4	1e-09	1337 (930)	1054 (959)	961 (925)	970 (933)	975
10^4	1e-12	1861 (1357)	1564 (1378)	1364 (1360)	1346 (1363)	1453
10^5	1e-06	1530 (1097)	1307 (1124)	1169 (1162)	1141 (1158)	1216
10^5	1e-09	3391 (2663)	2920 (2656)	2672 (2724)	2560 (2584)	3052
10^5	1e-12	4865 (4155)	4229 (4001)	4018 (4005)	3991 (3936)	4686
10^6	1e-06	2505 (1895)	2104 (1957)	1876 (1983)	1919 (1830)	2297
10^6	1e-09	8152 (7070)	7466 (7359)	7348 (7356)	7467 (6939)	10075
10^6	1e-12	13532 (12355)	12053 (13008)	11833 (12201)	11985 (11675)	18824
Total iters		37930 (32046)	33312 (32960)	31775 (32221)	31894 (30913)	43103

Table 2.16. NONRAND problems: mean number of iterations of the SDA and DY methods. SDAM iterations in brackets if different from the SDA one.

Problem		SDC (h, m)				DY
$\kappa(A)$	*tol*	(2,18)	(4,18)	(8,18)	(10,18)	
10^4	1e-06	726 (505)	593 (508)	511 (527)	522 (512)	525
10^4	1e-09	1158 (921)	1004 (947)	934 (953)	943 (923)	975
10^4	1e-12	1607 (1321)	1416 (1354)	1324 (1338)	1329 (1338)	1453
10^5	1e-06	1608 (1120)	1157 (1200)	1188 (1138)	1085 (1139)	1216
10^5	1e-09	2861 (2633)	2566 (2719)	2474 (2544)	2578 (2476)	3052
10^5	1e-12	4225 (4103)	3903 (4090)	3859 (3679)	3885 (4007)	4686
10^6	1e-06	2174 (1897)	1946 (2072)	1881 (2015)	2023 (1975)	2297
10^6	1e-09	7334 (7514)	6777 (7343)	7072 (7341)	6972 (6885)	10075
10^6	1e-12	11237 (11823)	11158 (11782)	12173 (11529)	11497 (11666)	18824
Total iters		32930 (31837)	30520 (32015)	31416 (31064)	30834 (30921)	43103

Table 2.17. NONRAND problems: mean number of iterations of the SDC and DY methods. SDCM iterations in brackets if different from the SDC one.

Finally, we considered another set of test problems, consisting of the Laplace1(a) and Laplace1(b) problems described in [32], which arise from a uniform 7-point finite-difference discretization of the 3D Poisson equation on a box, with homogeneous Dirichlet boundary conditions. These problems have 10^6 variables and a highly sparse Hessian matrix with condition number $10^{3.61}$. For each problem 5 starting points were generated by `rand` with entries in $[0, 1]$; the iteration was terminated when the stopping criterion (2.32) was satisfied, using $tol = 10^{-2}, 10^{-4}, 10^{-6}$, to check the effects of different accuracy requirements.
We decided to use this set of problems to test the RSDA method and adaptive versions of SDA and SDC using $m = 5$, as suggested in [26]. We did not test the SDA_ADP and SDC_ADP on the RAND and NONRAND problems since this analysis was partially implicitly done, being the switch condition usually satisfied for small value of h, always less than 20. The algorithms were also compared with the CG method implemented in the MATLAB `pcg` function, the BB, the DY and

Problem		SDA (h, m)					DY
$\kappa(A)$	*tol*	(2,30)	(4,30)	(8,30)	(10,30)	(20,30)	
10^4	1e-06	503 (274)	400 (282)	329 (267)	284 (260)	269 (286)	222
10^4	1e-09	1437 (799)	948 (703)	773 (713)	818 (727)	708 (729)	683
10^4	1e-12	2049 (1140)	1348 (1108)	1164 (1076)	1088 (1053)	1089 (1086)	1140
10^5	1e-06	455 (289)	377 (284)	355 (288)	294 (275)	287 (274)	228
10^5	1e-09	2390 (1603)	2067 (1463)	1365 (1498)	1434 (1407)	1290 (1536)	1839
10^5	1e-12	2876 (2697)	2522 (2360)	2238 (2630)	2530 (2381)	2244 (2651)	3469
10^6	1e-06	473 (280)	382 (256)	300 (268)	286 (255)	271 (258)	204
10^6	1e-09	3568 (2915)	2583 (2910)	2299 (2516)	2560 (2826)	2891 (2343)	4283
10^6	1e-12	4712	3883 (6099)	4533 (6837)	5072 (6362)	6169 (5448)	13231
Total iters		18463 (16096)	14510 (16203)	13356 (14874)	14366 (15546)	15218 (14611)	25299

Table 2.18. RAND problems: mean number of iterations of the SDA and DY methods. SDAM iterations in brackets if different from the SDA one.

Problem		SDC (h, m)					DY
$\kappa(A)$	*tol*	(2,30)	(4,30)	(8,30)	(10,30)	(20,30)	
10^4	1e-06	394 (305)	353 (268)	273 (255)	287 (263)	288 (268)	222
10^4	1e-09	1238 (777)	972 (679)	717 (680)	730 (653)	703 (714)	683
10^4	1e-12	1483 (1135)	1253 (1064)	1142 (1037)	1054 (1056)	1066 (1087)	1140
10^5	1e-06	428 (310)	376 (264)	302 (268)	322 (282)	289 (267)	228
10^5	1e-09	1349 (1427)	1738 (1329)	1363 (1546)	1322 (1229)	1269 (1553)	1839
10^5	1e-12	2079 (2526)	2610 (2553)	2078 (2569)	2173 (2282)	2269 (2581)	3469
10^6	1e-06	397 (290)	354 (248)	317 (265)	309 (279)	265 (257)	204
10^6	1e-09	2580 (2563)	2701 (2966)	2003 (3075)	2405 (2626)	2699 (2617)	4283
10^6	1e-12	3557 (4950)	3920 (6120)	3944 (6572)	5120 (6402)	6330 (5240)	13231
Total iters		13505 (14283)	14277 (15491)	12139 (16267)	13722 (15072)	15470 (14584)	25299

Table 2.19. RAND problems: mean number of iterations of the SDC and DY methods. SDCM iterations in brackets if different from the SDC one.

the RSD methods, in our implementations. In Tables 2.22-2.24, for each problem, we report the average number of iterations for the six algorithms, as in the previous set of test problems, for each starting point, RSD and RSDA were run 10 times varying the seed in the `rand` function used in the choice of the steplength.

The results in Table 2.24 show that CG outperforms the other methods when high accuracy is required. In this case, the RSDA and RSD methods achieve the poorest results, with the RSDA method showing a significant improvement over the RSD method; the BB, DY, SDA_ADP and SDC_ADP methods take a smaller number of iterations than the RSDA and RSD methods, but are still much slower than the CG method. Very interesting are the results in Tables 2.22 and 2.23, which suggest that, for low accuracy requirements, gradient methods, especially the DY, SDA_ADP and SCD_ADP methods, provide reasonable alternatives to the CG method, for instance in the computational contexts outlined in [32] and [46]. The

Problem		SDA (h, m)					DY
$\kappa(A)$	*tol*	(2,30)	(4,30)	(8,30)	(10,30)	(20,30)	
10^4	1e-06	1402 (575)	863 (528)	646 (550)	627 (548)	564 (555)	525
10^4	1e-09	1868 (1001)	1512 (970)	1124 (955)	1084 (987)	959 (937)	975
10^4	1e-12	2485 (1401)	1998 (1353)	1565 (1395)	1521 (1400)	1355 (1381)	1453
10^5	1e-06	2056 (1207)	1607 (1184)	1301 (1114)	1241 (1153)	1166 (1174)	1216
10^5	1e-09	4752 (2648)	3620 (2641)	2959 (2575)	2704 (2732)	2568 (2638)	3052
10^5	1e-12	6764 (3932)	5191 (4002)	4356 (3986)	4149 (3970)	4054 (4045)	4686
10^6	1e-06	3912 (1953)	2682 (2032)	2159 (1985)	2043 (1889)	1918 (1829)	2297
10^6	1e-09	10556 (7245)	8279 (6913)	7458 (7678)	7298 (7429)	6922 (6839)	10075
10^6	1e-12	17572 (12278)	12882 (10946)	12332 (11887)	11902 (12687)	11421 (11282)	18824
Total iters		51367 (32240)	38634 (30569)	33900 (32125)	32569 (32795)	30927 (30680)	43103

Table 2.20. NONRAND problems: mean number of iterations of the SDA and DY methods. SDAM iterations in brackets if different from the SDA one.

Problem		SDC (h, m)					DY
$\kappa(A)$	*tol*	(2,30)	(4,30)	(8,30)	(10,30)	(20,30)	
10^4	1e-06	1402 (556)	918 (558)	614 (541)	568 (571)	579 (538)	525
10^4	1e-09	1645 (997)	1246 (984)	1066 (948)	990 (984)	989 (992)	975
10^4	1e-12	2296 (1447)	1689 (1386)	1442 (1395)	1404 (1397)	1353 (1399)	1453
10^5	1e-06	2044 (1144)	1542 (1140)	1196 (1174)	1165 (1169)	1125 (1157)	1216
10^5	1e-09	4550 (2659)	2985 (2584)	2634 (2735)	2712 (2615)	2709 (2605)	3052
10^5	1e-12	6411 (4117)	4235 (3983)	3966 (4120)	3982 (3992)	3939 (4069)	4686
10^6	1e-06	4159 (2056)	2556 (1875)	2044 (2110)	1911 (1984)	1949 (1820)	2297
10^6	1e-09	9027 (7232)	7108 (6879)	6846 (7075)	6522 (6986)	6755 (6805)	10075
10^6	1e-12	14585 (12139)	11641 (11572)	10928 (11717)	10937 (11510)	11952 (11758)	18824
Total iters		46119 (32347)	33920 (30961)	30736 (31815)	30191 (31208)	31350 (31143)	43103

Table 2.21. NONRAND problems: mean number of iterations of the SDC and DY methods. SDCM iterations in brackets if different from the SDC one.

results in Table 2.24 show that the performance of the gradient algorithms with respect to the CG method seriously deteriorates as the stopping condition becomes stronger. About the SDA_ADP and SDC_ADP methods, we note that their behaviour depends on the SD ability to provide a good approximation of $1/(\lambda_1 + \lambda_n)$ through $\widetilde{\alpha}_k$ (Proposition 2.2.1), and of $1/\lambda_1$ through (2.20) (Proposition 2.2.4). A rather inaccurate approximation (which, in our experience, is usually achieved very soon using the SDA_ADP and SDC_ADP methods) can be sufficient to get a low-accuracy solution in few iterations. Conversely, getting high accuracy in the solution requires a good approximation of $1/(\lambda_1+\lambda_n)$ and $1/\lambda_1$, and therefore many SD iterates, which can be counterproductive increasing the total iterations of the methods.

We conclude this chapter by observing that the ability of the SDA and SDC

Problem	CG	BB	DY	RSDA	RSD	SDA_ADP	SDC_ADP
Laplace1(a)	16	14	12	14	18	17	14
Laplace1(b)	16	14	12	14	18	17	14

Table 2.22. Iterations for the Laplace problems, with stop condition $\|g_k\| < 10^{-2}\|g_0\|$.

Problem	CG	BB	DY	RSDA	RSD	SDA_ADP	SDC_ADP
Laplace1(a)	135	225	185	269	406	186	181
Laplace1(b)	135	205	196	282	397	184	184

Table 2.23. Iterations for the Laplace problems, with stop condition $\|g_k\| < 10^{-4}\|g_0\|$.

Problem	CG	BB	DY	RSDA	RSD	SDA_ADP	SDC_ADP
Laplace1(a)	181	484	389	596	900	392	417
Laplace1(b)	181	495	397	593	913	416	393

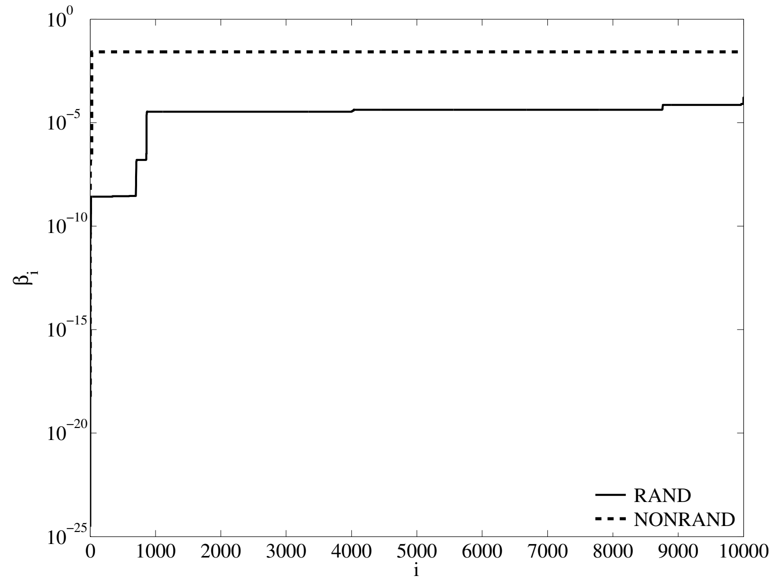
Table 2.24. Iterations for the Laplace problems, with stop condition $\|g_k\| < 10^{-6}\|g_0\|$.

methods to eliminate the eigencomponents corresponding to the largest eigenvalues, already pointed out in Section 2.2, is confirmed by the experiments on the RAND and NONRAND problems. A clear picture of this feature is provided in Figure 2.8, where the scalars

$$\beta_i^k = \sqrt{\sum_{j=1}^{i} \left(\mu_j^k\right)^2}, \quad i = 1, \dots, n,$$

are plotted at the last iteration of the SDA and SDC methods, using $h = 30$ and $m = 8$, applied to specific instances of the RAND and NONRAND problems with $\kappa(A) = 10^6$ and $tol = 10^{-12}$. Such ability is especially apparent for the RAND problem, for which the size of the gradient at the last iteration ($\|g_k\| \simeq 10^{-5}$) is mainly determined by the eigencomponents μ_i^k associated with few smallest eigenvalues. Even though this phenomenon is visible in both pictures of Figure 2.8, it clearly emerges that the initial values of β_i^k are smaller for the SDC method; this is due to the most efficient approximation of $1/\lambda_i$ provided by the use of the Yuan steplengths.

The asymptotic properties of the SDA and SDC methods are also confirmed by the fact that they own the *decreasing together* property [23], as the fast gradient methods presented in Chapter 1. To highlight how the gradient components with respect to the eigenvectors of the Hessian matrix are decreasing together, we again



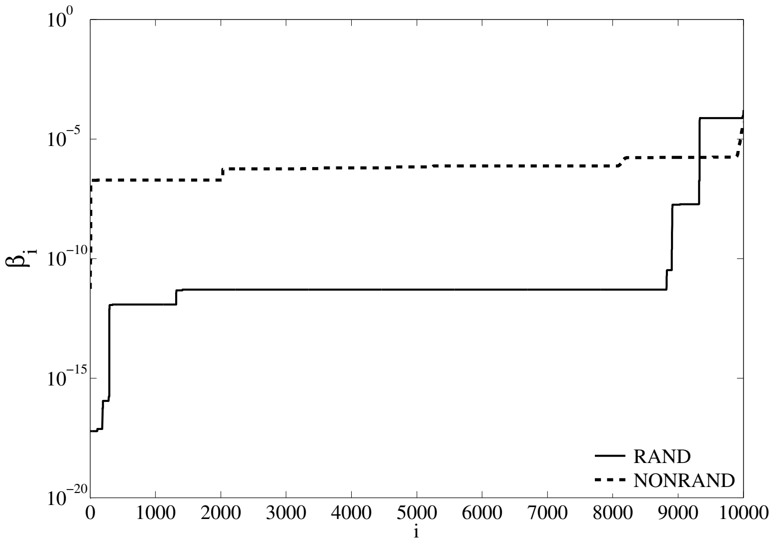


Figure 2.8. RAND and NONRAND problems with $\kappa(A) = 10^6$: values of the scalars β_i, $i = 1, \dots, n$, at the last SDA (top) and SDC (bottom) iterations ($tol = 10^{-12}$).

show the values of s (see Formula (1.32) in Chapter 1) for Problem 2.31 in Table 2.25. We set $h = 2$ and $m = 2, 4, 6$. As expected, s increases almost always as ϵ becomes smaller, reaching its maximum value in the SDC method for $\epsilon = 10^{-12}$ and $m = 4$. The behaviour of the SDA and the SDC methods for $m = 2$ confirms the fact that is the number of constant steps which plays the main role in the methods, the improvement brought by $m = 4$ and $m = 6$ is in fact not negligible. Conversely, the DY method, even if it exhibits a good behaviour, demonstrating that it has the

ϵ	SDA (h,m)			SDC (h,m)			DY
	(2, 2)	(2, 4)	(2,6)	(2, 2)	(2, 4)	(2,6)	
10^{-3}	292	510	209	178	188	183	192
10^{-6}	473	793	883	521	508	619	473
10^{-9}	481	860	980	387	907	907	754
10^{-12}	367	854	984	754	987	975	761

Table 2.25. Problem 2.31, values of s for the SDA, the SDC and the DY methods.

decreasing property, does not succeed in increasing the size of s, restraining as ϵ becomes smaller.

Chapter 3

An application to discrete ill-posed problems

In this chapter we consider discrete linear problems of the form

$$\mathbf{b} = A\mathbf{x} + \mathbf{n}, \tag{3.1}$$

where $A \in R^{m\times n}$ and $\mathbf{b} \in R^m$ $(m \geq n)$ are known data, $\mathbf{n} \in R^m$ is unknown and represents perturbations in the data, and $\mathbf{x} \in R^n$ represents an object to be recovered. Such problems arise in many application fields, such as signal and image processing, statistical inference, geophisics, and astronomy (see, e.g., [29]).
For example, in image restoration the matrix A models the blurring effect produced by the image acquisition process, $\mathbf{b}$ is the observed blurred and noisy image, $\mathbf{n}$ is some additive noise, and $\mathbf{x}$ is the original image. We assume that A is ill-conditioned, with singular values decaying to zero, as it is usually the case in the above-mentioned applications; we also assume that A is full rank.

Because of the ill-conditioning of A, computing the solution of the least squares problem

$$\min_{\mathbf{x}\in R^n} \frac{1}{2}\|\mathbf{b} - A\mathbf{x}\|^2 \tag{3.2}$$

does not provide a meaningful solution of (3.1), since it amplifies the noise contained in the data. Therefore, a regularization method is applied to compute a reasonable approximation to the exact solution. Roughly speaking, a regularization method replaces the original problem with a family of *close* better-conditioned (regularized) problems, depending on a parameter, such that, for an appropriate choice of the parameter, the solution of the corresponding regularized problem converges to the exact solution when the noise tends to zero [29]. The regularized problems can be obtained by adding to the objective function in (3.2) a penalty term based on some norm or seminorm of the solution, such as in the Tikhonov and l_1 regularizations, or by exploting the truncated SVD and GSVD decompositions, or by applying iterative methods (for more details see, e.g., [7, 29, 30, 45] and the references therein).

As observed in [7], iterative regularization methods for the solution of (3.2) are very flexible (e.g., they can be efficiently applied to both spatially variant and invariant blurs), allow easy integration of other regularization techniques, and easy treatment of constraints such as nonnegativity. These methods generally show a semiconvergence behaviour of the relative error, i.e., this error decreases in the early iterations and then begins to increase, due to the fact that the method eventually converges to the solution of (3.2), therefore a suitable early stop of the iterations is needed to obtain a good approximation to the solution. The choice of the iteration index where the method has to be stopped plays a fundamental role and it is based on further information on the problem. For example, the Morozov's discrepancy principle [51] requires terminating the iterations as soon as

$$|A\mathbf{x}_k - \mathbf{b}| \leq \tau\delta, \tag{3.3}$$

where δ is the so-called noise norm and $\tau > 1$.

The regularizing properties of the classical Landweber, steepest descent (SD) and conjugate gradient (CG) methods have been widely investigated (see, e.g., [29, 43, 52]). In particular, it is well known that the Landweber and SD methods generally exhibit very slow convergence, requiring a relatively large number of iterations before the discrepancy principle is satisfied, and thus they are rarely used in practice. Conversely, CG methods, such as CGLS and LSQR, rapidly compute a good approximation to the solution; however, they show a fast transition from convergence to divergence, hence being sensitive to the accuracy of the noise norm estimation. On the other hand, as we saw in Chapter 1, starting from the innovative Barzilai-Borwein approach [4], several new gradient methods have been developed that use suitable steplengths to achieve a significant speedup over SD [15, 22, 66, 23, 16, 33, 34, 35, 26, 24, 67]. This has motivated the interest toward their possible use as regularization methods, and recent work has been devoted to understand the behaviour of some of them in the solution of discrete inverse problems [3, 14].

In this chapter we analyse the regularization properties of the SDA and SDC methods, which have shown to be highly competitive with the above-mentioned fast gradient methods in terms of speed[1]. Both the SDA and SDC methods share the idea of fostering a selective elimination of the components of the gradient along the

[1]For further details see Section 2.4 in Chapter 2.

eigenvectors of the Hessian matrix, thus pushing the search in subspaces of smaller dimensions and speeding up the convergence of the method.

Following [52], we first perform a filter factor analysis of the two methods, showing that, when they are applied to the least squares problem (3.2), they tend to approximate first the components of the solution corresponding to large singular values of the matrix A, thus producing a useful filtering effect.

The rest of the chapter is organised as follows. In Section 3.1 we apply the SDA and SDC methods to Problem 3.2, highlighting the change in their main features. In Section 3.2 we analyse the regularizing properties of the two methods, by studying the associated filter factors. In Section 3.3 we present the results of numerical experiments concerning the application of the SDA and SDC methods to some well known test problems, confirming the previous findings and showing that the new gradient methods can provide an effective alternative to the CG methods.

3.1 Behaviour of the SDA and SDC methods

Gradient methods can be applied to Problem 3.2 generating a sequence of iterates $\{\mathbf{x}_k\}$ by using

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k, \tag{3.4}$$

where

$$\mathbf{g}_k = A^T(A\mathbf{x}_k - \mathbf{b}) \tag{3.5}$$

is the gradient at $\mathbf{x}_k$ of the objective function in (3.2) and $\alpha_k > 0$ is, as usual, a steplength computed by applying a suitable rule.

In order to analyse the behaviour of the SDA and the SDC methods, we consider the singular value decomposition of A,

$$A = U\Sigma V^T, \tag{3.6}$$

where $U = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_m] \in R^{m\times m}$, $V = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n] \in R^{n\times n}$, and $\Sigma = (\sigma_1, \sigma_2, \dots, \sigma_n) \in R^{m\times n}$. Note that the squares of the singular values σ_i are the eigenvalues of the Hessian matrix of the objective function in (3.2), and the right singular vectors $\mathbf{v}_i$ are a set of associated orthogonal eigenvectors. According to this, the Assumption 1 in Chapter 1 becomes,

$$\sigma_1 > \sigma_2 > \cdots > \sigma_n. \tag{3.7}$$

By using (3.4) and (3.5), it is easy to verify that if

$$\mathbf{g}_0 = \sum_{i=1}^{n} \mu_i^0 \mathbf{v}_i,$$

then

$$\mathbf{g}_k = \sum_{i=1}^{n} \mu_i^k \mathbf{v}_i, \qquad \mu_i^k = \mu_i^0 \prod_{j=0}^{k} (1 - \alpha_j \sigma_i^2), \tag{3.8}$$

which are Formulas (1.12)-(1.13) in Chapter 1 referred here to Problem 3.2.
In this way it follows again that, if at the k-th iteration $\mu_i^k = 0$ for some i, then for $l > k$ it will be $\mu_i^l = 0$, i.e., the component of the gradient along $\mathbf{v}_i$ will be zero at all subsequent iterations. Moreover, the condition $\mu_i^k = 0$ holds if and only if $\mu_i^0 = 0$ or $\alpha_j = 1/\sigma_i^2$ for some $j \leq k$.
In Chapter 2 we introduced the SDA and the SDC methods as follows:

$$\alpha_k^{SDA} = \begin{cases} \alpha_k^{SD} & \text{if } \operatorname{mod}(k, h_1) < h, \\ \widetilde{\alpha}_s & \text{otherwise, with } \; s \; = \max\{i \leq k \; : \; \operatorname{mod}(i, h_1) = h\}; \end{cases} \tag{3.9}$$

and

$$\alpha_k^{SDC} = \begin{cases} \alpha_k^{SD} & \text{if } \operatorname{mod}(k, h_1) < h, \\ \alpha_s^{Y} & \text{otherwise, with } \; s \; = \max\{i \leq k \; : \; \operatorname{mod}(i, h_1) = h\}, \end{cases} \tag{3.10}$$

where $h_1 > h \geq 2$.

The proposed steplength

$$\widetilde{\alpha}_s = \left(\frac{1}{\alpha_{s-1}^{SD}} + \frac{1}{\alpha_s^{SD}} \right)^{-1} \tag{3.11}$$

is related to the largest and smallest singular values of A being[2]

$$\lim_k \widetilde{\alpha}_k = \frac{1}{\sigma_1^2 + \sigma_n^2}\,. \tag{3.12}$$

As discussed in Chapter 2, this characteristic and the properties of the SD method suggest that the SDA method combines the tendency of the SD method to choose its search direction in the two-dimensional space spanned by $\mathbf{v}_1$ and $\mathbf{v}_n$ with the tendency of a gradient method with constant steplength $1/(\sigma_1^2 + \sigma_n^2)$ to align the search direction with $\mathbf{v}_n$. This yields a significant improvement of convergence speed over the SD method, as shown by the numerical experiments reported at the end of the chapter.

[2]See Proposition 2.2.1 in Chapther 2 for further details.

The SDC method, proposed in [24], uses as *special* constant step the Yuan steplength α_s^Y, where[3]

$$\lim_k \alpha_k^Y = \frac{1}{\sigma_1^2}. \tag{3.13}$$

By using this result and (3.8), we can conclude that the better the approximation of $1/\sigma_1^2$ provided by α_k^{SD}, the smaller the component along $\mathbf{v}_n$ of the gradient computed by using that steplength. According to (3.8), a multiple application of the step α_h^Y can drive toward zero the component of the gradient along $\mathbf{v}_1$, i.e., μ_1^k. In the ideal case where the component along $\mathbf{v}_1$ is completely removed, Problem 3.2 reduces to a $(n-1)$-dimensional problem, and a new sequence of Cauchy steps followed by some steps with a fixed value of α_h^Y can drive toward zero the component along $\mathbf{v}_2$. This procedure can be repeated with the aim of eliminating the components of the gradient according to the decreasing order of the singular values of A. The effectiveness of this approach is confirmed by the numerical experiments reported in the previous chapter and in [24].
Finally, we observe that if Problem 3.2 is ill conditioned, then

$$\frac{1}{\sigma_1^2 + \sigma_n^2} \approx \frac{1}{\sigma_1^2};$$

therefore, the SDA method tends to eliminate the components of the gradient along $\mathbf{v}_1$ similarly to the SDC method. More generally, it fosters the elimination of the components corresponding to the singular values $\sigma_i >> \sigma_n$.

3.2 Filtering properties of the SDA and SDC methods

We can express the solution of the least squares problem (3.2) by using an SVD decomposition (3.6) of A:

$$\mathbf{x}^\dagger = A^\dagger \mathbf{b} = \sum_{i=1}^{n} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i = \mathbf{x}_{true} + \sum_{i=1}^{n} \frac{\mathbf{u}_i^T \mathbf{n}}{\sigma_i} \mathbf{v}_i, \tag{3.14}$$

where $\mathbf{x}_{true}$ is the *true* solution of (3.1). Since the singular values of A decay to zero, the division by small singular values amplifies the corresponding noise components, and the solution $\mathbf{x}^\dagger$ results useless.

A regularized solution can be obtained by modifying the least squares solution (3.14) as

$$\mathbf{x}_{reg} = \sum_{i=1}^{n} \phi_i \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i, \tag{3.15}$$

[3]See Proposition 2.2.4 in Chapter 2 for further details.

where the scalars ϕ_i, called *filter factors*, are such that the components of the solution corresponding to large singular values are preserved ($\phi_i \approx 1$) and those corresponding to small singular values are filtered out ($\phi_i \approx 0$) [45]. Their choice is peculiar to discriminate among the existing filtering techniques. Well known methods are the Tikhonov and the truncated singular values decomposition methods, where the filter factors are respectively written as,

$$\phi_i = \frac{\sigma_i^2}{\sigma_i^2 + \epsilon}, \quad \text{and} \quad \phi_i = \begin{cases} 1 & \text{if } i \leq r, \\ 0 & \text{if } i > r \end{cases},$$

where ϵ is a nonnegative constant, and $r \in \{1, \ldots, n\}$ is called the regularization parameter.

As observed in [10, 8, 63, 7], iterative methods have many advantages over simple filtering techniques, having a cost depending on the amount of computation needed per iteration, as well as on the number of iterations needed to reach a good restoration of the image, and their convergence, often too slow, is usually accelerated using preconditioning. Preconditioning can be however a dangerous choice if not done carefully, it can indeed lead to erratic convergence behavior that results in fast convergence to a poor approximate solution. Therefore the need of fast gradient methods, with the aim of reaching a good approximate solution with a *reasonable* cost, without using any preconditioning thecnique.
To analyse the behaviour of our two gradient methods, the SDA and the SDC methods, we start with the following proposition[4], which gives the expression of the filter factors associated to a general gradient method.

Proposition 3.2.1. *Let $\{\mathbf{x}_k\}$ be the sequence of iterates produced by a general gradient method, assuming $\mathbf{x}_0 = \mathbf{0}$ to be the starting iterate, and using Formulas (3.4), (3.5); suppose to have a singular value decomposition of A (3.6), then we get the following expression of the $(k+1)$-th iterate:*

$$\mathbf{x}_{k+1} = \sum_{i=1}^{n} \left(1 - \prod_{l=0}^{k} \left(1 - \alpha_l \sigma_i^2 \right) \right) \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i,$$

where the scalars

$$\phi_i^{k+1} = 1 - \prod_{l=0}^{k} \left(1 - \alpha_l \sigma_i^2 \right), \quad i = 1, \ldots, n, \tag{3.16}$$

[4]Its proof is already known and used in [14, 58], we report it here to familiarize with the terminology which will be used in the following development of our analysis.

play the role of the filter factors associated with the $(k+1)$-th iterate.

Proof. We can write

$$\begin{aligned}\mathbf{x}_1 = \mathbf{x}_0 - \alpha_0 \mathbf{g}_0 = \mathbf{x}_0 - \alpha_0 A^T(A\mathbf{x_0} - \mathbf{b}) =\\ = \alpha_0 A^T \mathbf{b} = \alpha_0 A^T \sum_{i=1}^{n} (\mathbf{u}_i^T \mathbf{b})\mathbf{u}_i =\\ = \sum_{i=1}^{n} \alpha_0 \sigma_i^2 \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i}\mathbf{v}_i =\\ = \sum_{i=1}^{n} \{1 - (1 - \alpha_0\sigma_i^2)\} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i}\mathbf{v}_i,\end{aligned}$$

in this way,

$$\mathbf{x}_1 = \sum_{i=1}^{n} \{1 - \prod_{l=0}^{} (1 - \alpha_l \sigma_i^2)\} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i}\mathbf{v}_i = \sum_{i=1}^{n} \phi_i \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i}\mathbf{v}_i, \tag{3.17}$$

and we can use (3.17) as base case.
We now show that if the thesis is true for $\mathbf{x}_k$, so it is for $\mathbf{x}_{k+1}$.
Being

$$\mathbf{x}_k = \sum_{i=1}^{n} \phi_i \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i}\mathbf{v}_i, \quad \text{with} \quad \phi_i = 1 - \prod_{l=0}^{k-1} (1 - \alpha_l \sigma_i^2),$$

we get,

$$\mathbf{x}_{k+1} = \tag{3.18}$$

$$\begin{aligned}= (I - \alpha_k A^T A)\mathbf{x}_k + \alpha_k A^T \mathbf{b} =\\ = (I - \alpha_k A^T A) \sum_{i=1}^{n} \{1 - \prod_{l=0}^{k-1} (1 - \alpha_l \sigma_i^2)\} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i}\mathbf{v}_i + \alpha_k \sum_{i=1}^{n} (\mathbf{u}_i^T \mathbf{b}) A^T \mathbf{u}_i =\\ = \sum_{i=1}^{n} \{1 - \prod_{l=0}^{k-1} (1 - \alpha_l \sigma_i^2)\} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i}(1 - \alpha_k \sigma_i^2)\mathbf{v}_i + \alpha_k \sum_{i=1}^{n} (\mathbf{u}_i^T \mathbf{b}) A^T \mathbf{u}_i =\\ = \sum_{i=1}^{n} \{1 - \prod_{l=0}^{k-1} (1 - \alpha_l \sigma_i^2)\} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i}(1 - \alpha_k \sigma_i^2)\mathbf{v}_i + \alpha_k \sum_{i=1}^{n} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \sigma_i^2 \mathbf{v}_i =\end{aligned}$$

$$= \sum_{i=1}^{n} \left\{1 - \prod_{l=0}^{k} (1 - \alpha_l \sigma_i^2)\right\} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i}\mathbf{v}_i = \sum_{i=1}^{n} \phi_i \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i}\mathbf{v}_i. \tag{3.19}$$

Putting together (3.18) and (3.19) the proof is complete. □

From (3.16) it follows that if $\alpha_m = 1/\sigma_i^2$ for some $m \leq k$, then $\phi_i^{k+1} = 1$ at the

$(k+1)$-th iteration and all subsequent ones. In fact we would have

$$\phi_i^{k+1} = 1 - \prod_{l=0}^{k} \left(1 - \alpha_l \sigma_i^2\right) = 1 - (1-\alpha_0\sigma_i^2)(1-\alpha_1\sigma_i^2)\cdots(1-\alpha_m\sigma_i^2)\cdots(1-\alpha_k\sigma_i^2) = 1,$$

and of course that filter factor would be one at all subsequent iterations. More generally, the better α_m approximates $1/\sigma_i^2$ the closer ϕ_i^{k+1} will be to 1; furthermore, multiple values of α_m close to $1/\sigma_i^2$ push ϕ_i^{k+1} to go toward 1 quickly. We also note that $1/\alpha_m >> \sigma_i^2$ implies $\phi_i^k \approx 0$, therefore, the SDA and SDC methods, thanks to the use of the respective constant steplengths, are expected to (approximately) reconstruct the components of the pseudoinverse solution (3.14) according to the decreasing order of the associated singular values, thus producing a regularization effect. In order to illustrate this behaviour, in Figure 3.1 we plot the filter factors of SDA and SDC at the k-th iteration, with $k = 5, 10, 20, 40$, for the test problem `heat` from Hansen's *Regularization Tools* [44], using $n = 64$ and kappa $= 1$. For comparison purposes, we also plot the filter factors of SD and CG at the same iterations; more precisely, we consider the CGLS method and compute the corresponding filter factors as in [52]. In SDA and SDC we set $h = 4$ and $m = 2$. As shown in Section 2.4 of Chapter 2, the SDA and SDC methods work well for small values of m, such as $m = 2, 4$, when the constant steplengths provide fairly good approximations of the inverses of the squared singular values, i.e., when h is sufficiently large. Note that this leads in practice to the monotonicity of the methods, which can be lost as m increases with respect to h. On the other hand, too large values of h slow down the method because of the low efficiency of the Cauchy steps. Furthermore, the choice of h is also related to the size of the problem and to the accuracy requirement, and small values of h are effective when the size and the accuracy are modest, as in this case.

The plots show that the filter factors of the SDA and SDC methods behave as expected. Furthermore, as k increases the two methods produce a larger number of filter factors close to 1 than the SD method, i.e., they are faster in providing good approximations of the components of the solution corresponding to large singular values (note also that the SDC method is slightly faster than the SDA method). As well known, the filter factors of the CG method become soon oscillating. We observe that we get similar results with $h = 8$ and $h = 10$, but the SDA and SDC methods tend to reconstruct a slightly smaller number of solution components than with $h = 4$, according to the fact that larger values of h do not necessarily imply convergence acceleration. We are in fact able to give a theoretical characterization of the approximate solution computed through the SDA and SDC methods.

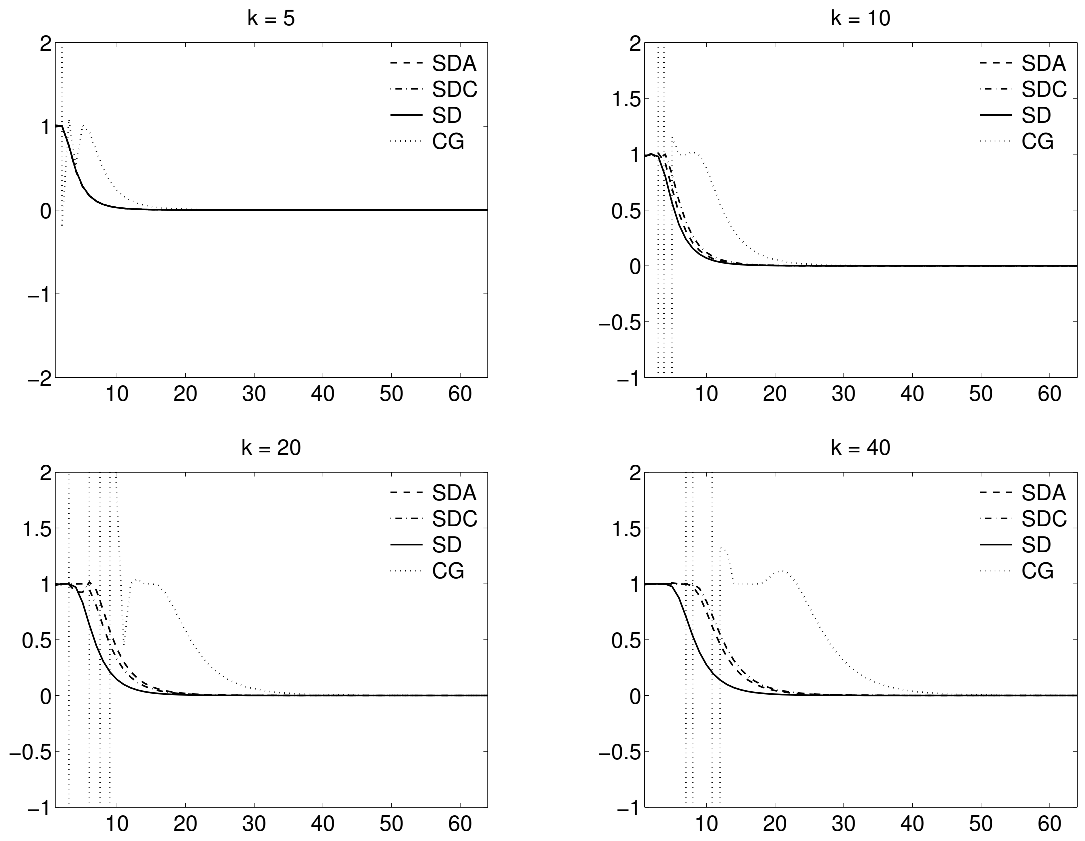


Figure 3.1. Filter factors of the SDA, SDC, SD and CG methods applied to problem `heat`, at iterations 5, 10, 20 and 40.

Being $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k$, using (3.4) and (3.8), we have that

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \sum_{i=1}^{n} \mu_i^0 \prod_{j=0}^{k-1} \left(1 - \alpha_j \sigma_i^2\right) \mathbf{v}_i. \tag{3.20}$$

Since the asymptotic behavior of the SDC methods, if $\alpha_m = 1/\sigma_i^2$ for some $m \leq k$, it can drive toward zero the component of the gradient along $\mathbf{v}_i$, fostering $(1 - \alpha_m \sigma_i^2)$ to go to zero. Without loss of generality, in order to simplify our notation, we can suppose $i = 1$. In the ideal case in which α_m is exactly $1/\sigma_1^2$, then (3.20) becomes,

$$x_{k+1} = x_k - \alpha_k \sum_{i=2}^{n} \mu_i^0 \prod_{l=0}^{k-1} \left(1 - \alpha_l \sigma_i^2\right) \mathbf{v}_i. \tag{3.21}$$

Iterating the algorithm, the gradient components related to the eigenvectors $\mathbf{v}_i$ associated to the largest eigenvalues will be neatly vanished, and so, by following this scheme, we are able to reconstruct the approximate solution $\mathbf{x}_{k+1}$ by fixing those components of the solution relative to the largest eigenvalues, obtaining an implicit *filtering effect*, which amplifies the regularization effect held by the method itself. Note that a similar but veiled analysis could be carrie out for the SDA method.

The effects of this kind of reconstruction, added to the ones coming from a faster convergence speed, made the SDA and the SDC competitive with the CGLS, if, for example, applied to an image deblurring problem, as it will be shown in the next section. Similar and further considerations can be found in [25].

3.3 Numerical experiments

The following tests are aimed at showing the behaviour of the SDA and SDC methods as regularization methods in two test problems. In this context they can represent an alternative to the CGLS method because of their fast convergence, with respect to the slow SD method, and the absence of any preconditioning technique. In addition to this, they also have a more stable semiconvergence property that the one held by the CGLS method, and this can be an attracting feature when a good estimation of the stop condition is not available. The CGLS relative error curve usually decreases very rapidly, reaching its minimum point after few iterations, but soon after it increases again, making a very delicate question where to stop the method to get a good restored image. Conversely, the SDA and the SDC methods exhibit a better behaviour, in terms of semiconvergence trend, the relative error stands in fact in an almost *flat region* for a larger number of iterations.
Looking ahead, the presence of constraints, useful in some cases, may inhibit taking a full step size in the calculated direction, and methods such as CG or Newton-type can lose efficiency [3, 17], while in this case, gradient-based method, thus the SDA and the SDC methods, can be advised.

The first problem to be analysed is an image deblur problem, we choose the image *satellite*[5] as test image. This image has been wildly used to test image restoration algorithms (see, e.g, [52, 53, 46, 6]). We used a Gaussian blur with $\sigma = 2$, and a noise level ranging in $\{5 \cdot 10^{-3}, 10^{-2}, 5 \cdot 10^{-2}, 10^{-1}\}$. The true and blurred images have 256×256 pixels, and are shown in Figure 3.2 and 3.3 respectively.

We tested the SDA and SDC methods, comparing them with the CGLS method. In this case, we set to be 200 the maximum number of iterations for all the methods.

[5]This image is included in the image restoration package, *RestoreTools* which can be obtained from `http://www.mathcs.emory.edu/~nagy/RestoreTools`. A software documentation can be found in [54]

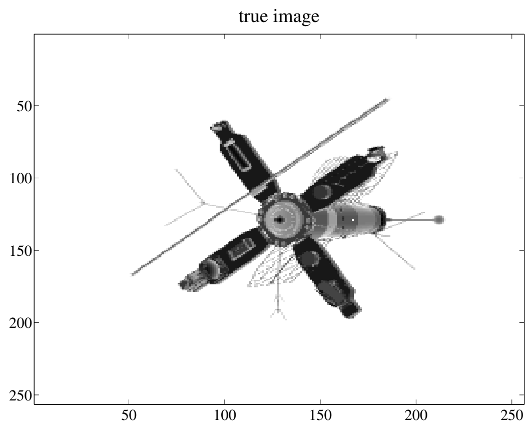


Figure 3.2. Satellite test problem, true image.

Table 3.1 shows the results of the comparison among the three methods. More precisely we show the value of the relative error e_k and the iteration index in two cases, when the discrepancy principle is satisfied (columns 3 and 5), and when it reaches its minimum point (columns 4 and 6). We can see that the SDA and SDC methods become more competitive with the CGLS as the noise level increases; the SDC method performs better than the SDA method as expected, reaching the iteration in which the discrepancy principle is satisfied earlier than the SDA method. About the values of the relative error we can see that there is not a great difference, but it is worth noting that in the case of the minimum relative error, the SDC method seems to give the best results.

To measure the number of iterations in which the relative error is relatively *flat*, we monitored the number of iterations in which

$$\frac{|e_k - e_k^*|}{e_k^*} < tol, \tag{3.22}$$

where e_k is the relative error at iteration k, and e_k^* is the minimum of the relative error; we fixed *tol* to be the noise level in each experiment. Table 3.2 shows the number of iterations in which (3.22) helds and the iterations in which (3.22) is satisfied for the first and the last time, respectively. In this way Table 3.2 can give a measure of the range length corresponding to the flat region of the relative error curve, while Figure 3.4 gives the trends of the relative error for the three methods using 10^{-2} as noise level to have a visual idea of what the table describes. The minimum point of each curve is in fact empathized by a square, while the bounds of the flat region are denoted by a dot. By this figure it is clearly visible the semi-convergence property of the methods, and the advantage concerning the choice of

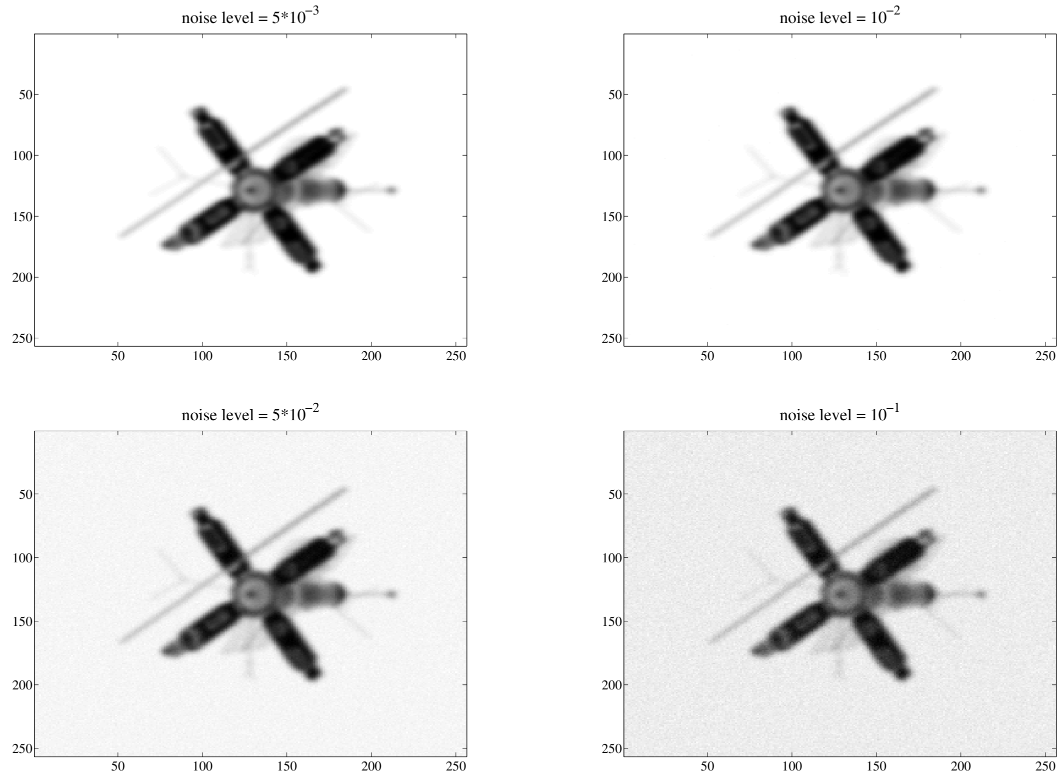


Figure 3.3. Satellite test problem, blurred and noisy images.

Noise Level	Method	e_k dp	e_k min	k dp	k min
0.005	SDA	1.886920e-01	1.798140e-01	58	145
	SDC	1.879430e-01	1.793376e-01	43	147
	CGLS	1.876028e-01	1.799671e-01	36	79
0.01	SDA	1.991304e-01	1.903892e-01	34	77
	SDC	1.991511e-01	1.888869e-01	27	69
	CGLS	1.983198e-01	1.897039e-01	22	45
0.05	SDA	2.236895e-01	2.185623e-01	10	17
	SDC	2.236079e-01	2.186493e-01	10	17
	CGLS	2.256961e-01	2.195784e-01	8	12
0.1	SDA	2.455619e-01	2.384617e-01	6	11
	SDC	2.437739e-01	2.362801e-01	6	11
	CGLS	2.407307e-01	2.372897e-01	6	8

Table 3.1. Satellite test problem, values of the relative error and iteration index.

the SDA and SDC method is understandable by looking at the flat region of the curves. This region starts almost at the same time for the three methods, but it

Noise Level	Method	#k	k first	k last
0.005	SDA	132	70	>200
	SDC	148	54	>200
	CGLS	78	43	120
0.01	SDA	129	33	161
	SDC	124	30	153
	CGLS	55	21	75
0.05	SDA	58	4	61
	SDC	34	4	37
	CGLS	24	4	27
0.1	SDA	48	2	49
	SDC	24	2	25
	CGLS	18	2	19

Table 3.2. Satellite test problem, relative error flat region features: number of iterations in the region (column 3), lower and upper bounds of the region (columns 4 - 5).

is much wider for the SDA and the SDC methods, as expected. From the results of Table 3.2 we can notice that, for all the values of the noise level, the SDA and SDC methods provide a clear improvement in the length of the flat region, even if a higher number of iterations is required by the two methods to reach the minimum point of the relative error. To better appreciate the effects of the methods on the reconstructions, Figure 3.5 reports the restored images by the different methods at the iteration in which the CGLS method reaches its minimum point using 10^{-2} as noise level, i.e. $k = 45$ (see the sixth row of Table 3.1). We are already in the flat region of the relative error curves, in fact this region started at iteration 33 for the SDA method, and at iteration 30 for the SDC method. The three images are comparable, but of course, being 45 closer to 69 than to 77, which are the iterations in which the SDC and the SDA methods reach the minimum relative error, the restored picture obtained using the SDC method is better than the one produced by the SDA method. This is confirmed by the analysis of the Improved Signal to Noise ratio (ISNR), which is 3.1674 for the SDA method, for 3.2533 the SDC method, and 3.3031 for the CGLS method. In Figure 3.6 we can see how at iteration $k = 153$, which is the upper bound of the flat region for the SDC method, the restored images of SDC and SDA are still comparable with the one obtained by stopping the methods in the minimum value of the CGLS relative error (Figure 3.5). Conversely this is not true for the CGLS method, giving back a worse restored image. Here again the values of the ISNR confirm what is appreciable to naked eye, being 2.8627 for the SDA method, 2.6279 for the SDC method, and -0.2875 for the CGLS method.

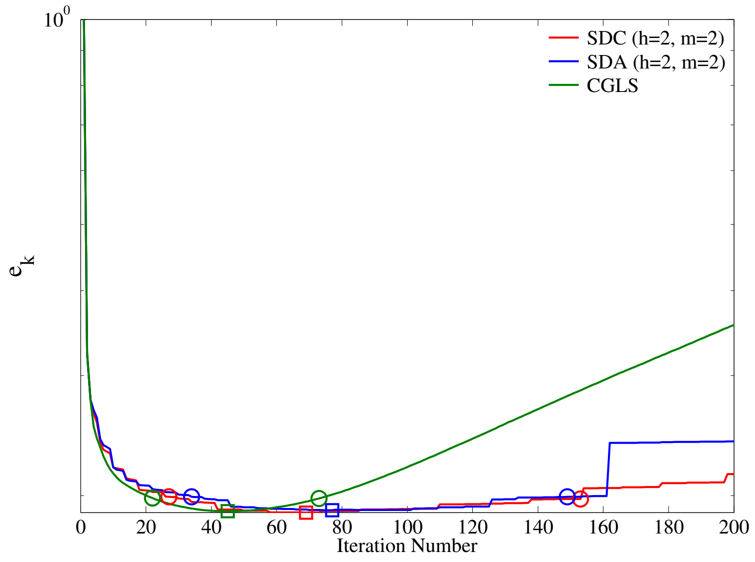


Figure 3.4. Satellite test problem, relative errors curves in the case of a noise level of 10^{-2}. The dots represent the first iteration (k first) and the last iteration (k last) in Table 3.2, while the squares represent the minimum point of the relative error (k min) in Table 3.2.

The second test problem is `heat`, the same problem we used in the filtering factor analysis in Section 3.2. Here again we set $n = 64$ and kappa $= 1$, but we used 1000 as maximum number of iterations. Table 3.3 shows the results of the comparison among the three methods, in terms of values of the relative error and the iteration index, when the discrepancy principle is satisfied (columns 3 and 5), and when it reaches its minimum point (columns 4 and 6), as already done for the satellite test problem. Again the SDC method behaves better than the SDA method, satisfying the discrepancy principle and reaching the minimum point of the relative error in fewer iterations, but in Table 3.4 there is a reverse trend, being the number of iterations included in the flat region of the SDA method bigger than the one associated to the SDC method.

Figure 3.7 shows the trend of the relative error curves for the SDA, SDC and CGLS methods using 10^{-3} as noise level. As we can see, the semiconvergence behaviour of the three methods clearly emerges. The curves of the SDA and SDC methods have a smoother behaviour, the region where they stand flat is emphasized by two dots, while the minimum point of each curve is again denoted by a square. Note that the same thing was also done for the CGLS method, but the flat region is inconsistent, being composed by just a few iterations, as it can be seen by the results in Table 3.4, where again (3.22) was used to estimate the range length.

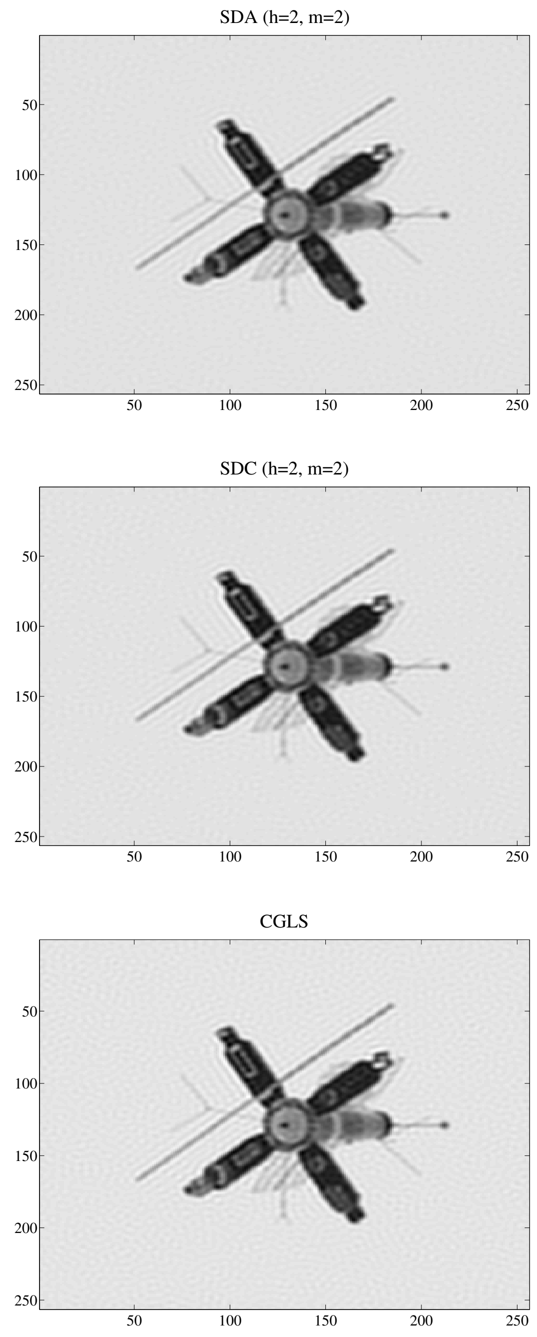


Figure 3.5. Satellite test problem, reconstructed images at the minimum point of the CGLS relative error - noise level 10^{-2}.

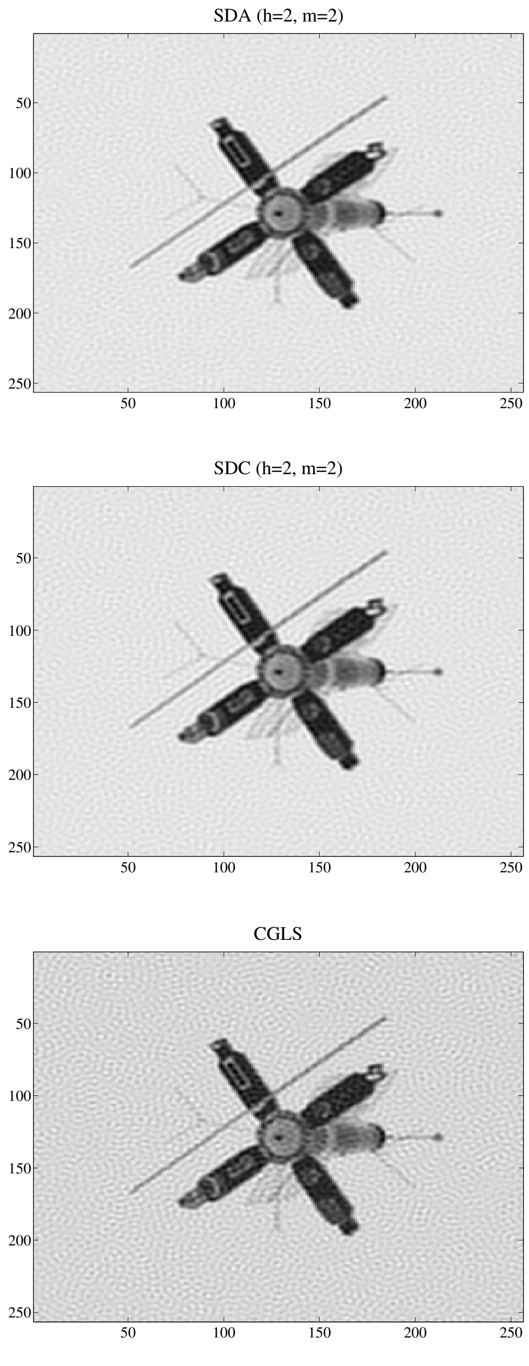


Figure 3.6. Satellite test problem, reconstructed images at the end of the flat region (k last in Table 3.2) of the SDC method (k last = 153) - noise level 10^{-2}.

Noise Level	Method	e_k dp	e_k min	k dp	k min
0.005	SDA	8.949351e-02	8.319592e-02	99	157
	SDC	9.436135e-02	8.347170e-02	78	102
	CGLS	8.914768e-02	8.683880e-02	15	16
0.01	SDA	1.206892e-01	1.081965e-01	70	96
	SDC	1.203297e-01	1.086671e-01	55	85
	CGLS	1.147467e-01	1.147185e-01	13	14
0.05	SDA	2.262340e-01	2.125727e-01	34	49
	SDC	2.357672e-01	2.178166e-01	26	45
	CGLS	2.240534e-01	2.109199e-01	8	9
0.1	SDA	3.460194e-01	2.862764e-01	26	34
	SDC	3.445608e-01	2.891912e-01	18	26
	CGLS	2.841307e-01	2.736020e-01	7	8

Table 3.3. heat test problem, values of the relative error and iteration index.

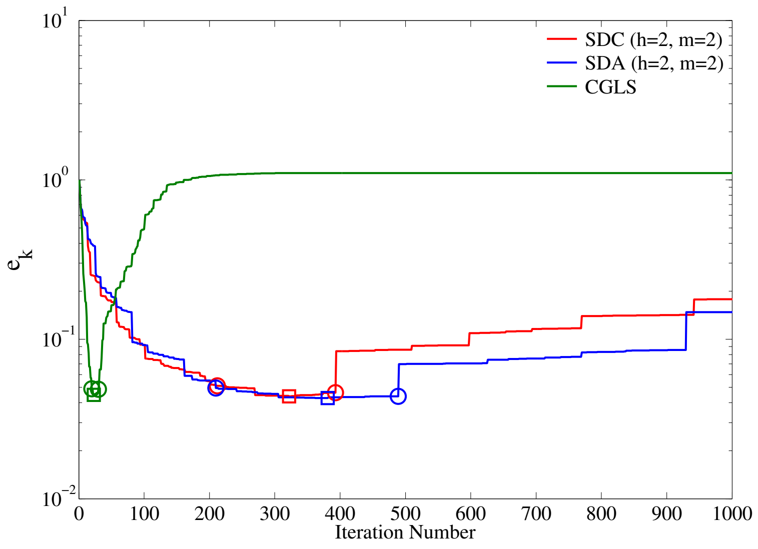


Figure 3.7. Heat test problem, relative errors curves. The dots represent the first iteration (k first) and the last iteration (k last) in Table 3.4, while the squares represent the minimum point of the relative error (k min) in Table 3.4.

Noise Level	Method	#k	k first	k last
0.005	SDA	52	106	157
	SDC	52	86	137
	CGLS	3	15	17
0.01	SDA	56	86	141
	SDC	55	59	113
	CGLS	3	13	15
0.05	SDA	68	30	97
	SDC	34	26	59
	CGLS	5	7	11
0.1	SDA	32	26	57
	SDC	24	18	41
	CGLS	3	7	9

Table 3.4. Heat test problem, relative error flat region features: number of iterations in the region (column 3), lower and upper bounds of the region (columns 4 - 5).

Chapter 4

Conclusions

There are many potential applications for fast gradient based methods, capable of good performances, with a limited cost per iteration, and a low memory requirement. Examples include image processing, denoising and deblurring problems, compressed sensing, machine learning, and data analysis.
This thesis focused on two new gradient methods for unconstrained nonlinear minimization problems, based on the use of some spectral properties associated with the Steepest Descent (SD) method. These properties can be suitably used to build some special steplengths to influence the performance of the SD method, dramatically improving its poor original behavior. To this end, we have considered different strategies, well known in literature, and we have developed a theoretical base above which we founded our two new methods, the Steepest Descent with Alignment (SDA) method, and the Steepest Descent with Constant steplength (SDC) method. We have tested our methods comparing them to some among the most competitive existing methods, and finally, we have exhibited a possible field of application in which our methods seem to have good and attractive features.

4.1 Summary

In Chapter 1 we formally introduced the unconstraint nonlinear quadratic problem, and we gave an overview of some methodologies to solve it. Starting from the spectral analysis made by Nocedal et al. [55], we moved from the classical Steepest Descent (SD) method, to the most recent Dai-Yuan (DY) method, through a long history of attempts to improve the performances of the too slow SD method, while keeping the antigradient as search direction.

In Chapter 2 we presented a theoretical analysis to support the development of two new gradient methods, the Steepest Descent with Alignment (SDA) method, and the Steepest Descent with Constant steplength (SDC) method. More in detail, we saw how a cyclical alternation of Cauchy steplengths and of constant steps computed through two “special” formula with favorable spectral properties, give birth to an algorithmic framework whose computational results are superior of those of the most competitive methods in literature. This alternation fosters a selective elimination of the components of the gradient along the eigenvectors of the Hessian matrix, thus pushing the search in subspaces of smaller dimensions, speeding up the convergence of the method. Furthermore we saw how monotony can be preserved without affecting the performances of the methods, but adding other spectral informations.

Finally, in Chapter 3 we considered the regularization properties of the SDA and SDC method. We first performed a filter factor analysis of the two methods, showing how they tend to approximate first the components of the solution corresponding to large singular values, producing an implicit filtering effect. We then presented the results of some numerical experiments highlighting the behaviour of the two methods.

4.2 Future Directions

In this thesis we looked at the following unconstrained quadratic minimization problem,

$$\min_{x \in R^n} \frac{1}{2} x^T A x - b^T x,$$

where $A \in R^{n \times n}$ is symmetric positive-definite and $b \in R^n$. Although its apparent simplicity, this setting allowed us to develop a theoretical analysis to study of the relevance of the eigenvalues of the Hessian of the objective function to the considered methods, highlighting the effects of the use of a particular steplength instead of another. Furthermore, we decided to consider this setting since it puts up the basis for the development of gradient methods for constrained problems and it can easily lead to the minimization of general non-quadratic functions. These two directions are briefly described here.

4.2.1 Quadratic constrained problems

We consider the box-constrained quadratic programming problem

$$\begin{aligned} \min f(x) = \frac{1}{2} x^T A x - b^T x \\ \text{s.t. } l \leq x \leq u, \end{aligned} \tag{4.1}$$

where $A \in R^{n\times n}$ is symmetric positive-definite, and b, l, u (with $l \leq u$) are vectors in R^n. Some interesting applications of this problem may be found in [49, 36, 37]. The peculiar characteristic of Problem 4.1 is that once the optimal face of the box is identified, it reduces to the unconstrained minimization of a quadratic function. Thus an algorithm for solving Problem 4.1 has to identify the optimal face and to minimize the quadratic (unconstrained) function on the face. Projected gradient (PG) methods can provide an efficient way of solving Problem 4.1 since they have the advantage that many constraints can be added or deleted from the working set on each iteration [17]. The need of faster projected gradient methods is due to the fact that even in this case the SD method proceeds very slow once the optimal face is identified. Early attempts in producing PG methods can be found in [38, 47, 9], while alternative developments fo PG methods improved the former by incorporating fast gradient based methods for unconstrained optimization. Among all is remarkable the use of the conjugate gradient (CG) method [57, 65, 50]. In particular, in [50] a new algorithm called the GPCG method is proposed. This method uses the PG method until either a suitable face is identified, or the PG method fails to make reasonable progress. If one of these two situations happened, the current face is explored by using the CG method, reserving the possibility to switch back to the PG method, if the CG method does not make significant progress. Other efficient PG methods are the Projected Barzilai-Borwein (PBB) and the Projected Alternate Barzilai-Borwein methods [17], and the Spectral Projected Gradient (SPG) method [11].
We started studying the behaviour of the SDA and SDC methods incorporated into a projected framework, therefore calling them the PSDA and PSDC methods.
If we define Ω to be the feasible set of Problem 4.1

$$\Omega = \{x \in R^n : l \leq x \leq u\},$$

and P to be the projection on to Ω,

$$P[x] = \text{mid}(l, u, x)$$

where $\text{mid}(l, u, x)$ is the vector whose i-th component is the median of the set $\{l_i, u_i, x_i\}$, assuming x_k to be a feasible point, the next feasible iterate x_{k+1} is then given by,

$$x_{k+1} = P[x_k - \alpha_k g_k],$$

where α_k is a positive steplength.
Preliminary results were obtained making one projection every time a cycle, formed by h iterates computed through the Cauchy step and m iterates computed through

Formulas (2.33)-(2.20), is completed. We followed a scheme similar to the one proposed in [61], incorporating this with our cyclical framework.
We defined the set of the ϵ-binding variables,

$$A = A(x) = \{i : l_i \le x_i \le l_i + \epsilon(x) \text{ and } g_i > 0, \text{ or } u_i - \epsilon(x) \le x_i \le u_i \text{ and } g_i < 0\},$$

where $w(x) = \|x - P[x-g]\|$, $\epsilon(x) = \min\{\epsilon, w(x)\}$, and we set I to be the complement of A. A projection onto the feasible set, involving the variables which are not ϵ-binding, is performed every time an inner cycle is completed. In this way the algorithm is divided in outer and inner iterations. Let x^{inn} be an iterate computed in an inner iteration and x_{out} be the one computed in the last outer iteration (thus x_{out} is feasible), the search direction $d = \|x_{inn} - x_{out}\|$ is chosen if it satisfies the following conditions:

$$\begin{aligned} \langle d, g_{out}\rangle_I &\le -\sigma_1 \|g_{out}\|_I^2 \\ \|d\|_I &\le \sigma_2 \|g_{out}\|_I. \end{aligned} \tag{4.2}$$

where g_{out} is the gradient computed in x_{out}, and σ_1 and σ_2 are two parameters. If d is not a descent direction, the last computed descent direction is taken and the inner cycle ends. Strict monotonicity is required at the outer iterations, and this is granted by an Armijo line search procedure [2].
We generated a test problem by using the Matlab function `sprandsym`, setting `density` $= 0.3$, `kind` $= 1$, and condition number $\kappa(A) = 10^5$. x^* was generated by `rand` with entries in $[-10, 10]$, and $b = Ax^*$ was used to build the linear term. The lower and upper bounds were set to be the constant vectors of -1 and 1, respectively, while the starting point was set to be a vector of zeros.
Table 4.1 shows the results of the application of this projected framework to the problem described above, where we multiplied both the lower and the upper bounds for a factor $\mu \in \{0.1, 0.2, \ldots, 1\}$. In the PSDA and PSDC methods we set $h = 4$ and $m = 6$[1]. We compared them with the PBB and the PABB methods, implemented using the same framework, making a projection every $t = 10$ inner BB or ABB iterations (these methods will be denoted by PBB$_t$ and PABB$_t$), and with the PBB and PABB implemented using the scheme and the nonmonotone line search reported in [17]. For the PSDA, PSDC, PBB$_t$ and PABB$_t$ methods we used $\|g_k\|_{I_k} \le 10^{-5}$, as in [61], as stop condition, while the PBB and PABB methods used,

$$\|\nabla f_\Omega(x_k)\| \le 10^{-5},$$

[1]We used $\sigma_1 = 0.1$, $\sigma_2 = 10$, and $\epsilon = 10^{-5}$.

where

$$[\nabla f_{\Omega}(x_k)]_i = \begin{cases} (g_k)_i & \text{if } (x_k)_i \in (l_i, u_i), \\ \min\{(g_k)_i, 0\} & \text{if } (x_k)_i = l_i, \\ \max\{(g_k)_i, 0\} & \text{if } (x_k)_i = u_i. \end{cases}$$

As we can see, the PSDC gives the best results, followed by the PSDA method.

μ	PSDA	PSDC	PBB$_t$	PABB$_t$	PBB	PABB
1	795	736	1357	680	1249	937
0.9	1100	926	1032	1050	1447	629
0.8	1069	1011	1456	1267	781	770
0.7	1107	951	1354	1145	1804	1184
0.6	1250	1008	1786	1068	1865	1321
0.5	1176	1002	1314	1976	1301	914
0.4	476	535	754	708	715	916
0.3	453	448	592	427	352	898
0.2	292	267	327	381	251	551
0.1	363	216	136	206	84	126
Total Iters	8081	7100	10108	8909	9849	8246

Table 4.1. Quadratic constrained problems, iteration number for the different projected methods.

Both the PABB$_t$ and the PABB methods are competitive with the PSDA and PSDC methods, and behave better than the correspondent PBB$_t$ and PBB methods, as expected.

This was a rough version of a possible strategy to extend the SDA and SDC methods in the contest of constrained quadratic problems, a future careful work will be devoted in designing a new projected framework similar to that of the GPCG method to better use the properties of the SDA and SDC methods.

4.2.2 Minimization of general non-quadratic functions

To generalize the SDA and SDC methods to nonlinear non-quadratic problems, we consider the following general unconstrained nonlinear minimization problem,

$$\min f(x)$$
$$\text{s.t. } x \in R^n,$$

where $f : R^n \to R$ is a continuously differentiable function.

In order to use our methods, we look for a good candidate to approximate the

Cauchy steplength. This can be given by,

$$arg \min_{\alpha} \phi(\alpha) = f(x_k - \alpha g_k), \tag{4.3}$$

where x_k is the iterate at iteration k and g_k is the gradient of f in the point x_k. Given an initial steplength α_0, we can perform a quadratic interpolation [56] of the function $\phi(\alpha)$ by interpolating the points $\phi(0)$, $\phi(\alpha_0)$ and $\phi'(0)$ obtaining the following expression,

$$\phi_2(\alpha) = \left(\frac{\phi(\alpha_0) - \phi(0) - \alpha_0 \phi'(0)}{\alpha_0^2} \right) \alpha^2 + \phi'(0)\alpha + \phi(0), \tag{4.4}$$

whose minimum is reached in:

$$\bar{\alpha}_0 = -\frac{\phi'(0)\alpha_0^2}{2\left[\phi(\alpha_0) - \phi(0) - \alpha_0 \phi'(0)\right]}. \tag{4.5}$$

We define the new candidate point to be, $\bar{x}_1 = x_0 - \bar{\alpha}_0 g_0$.
if the Armijo or the Wolfe conditions are satisfied in this point, then we set $x_1 = \bar{x}_1$, otherwise we divide $\bar{\alpha}_0$ by a factor of 2 and we check again the line search conditions until they are satisfied. Once a new point and a new gradient becomes available, the procedure is iterated computing a new quadratic interpolation. We iterate this procedure, summarized in Algorithm 4, until a suitable stop condition is satisfied.

Algorithm 4 General gradient method for non-quadratics (GSD method)

```
choose $\alpha_0 > 0$, $x_0 \in R^n$
compute $g_0$, $f_0$
set $k = 0$
while not stop_condition do
    compute the quadratic interpolation to select $\bar{\alpha}_k > 0$
    $x_{k+1} = x_k - \bar{\alpha}_k g_k$
    compute $f(x_{k+1})$, $g_{k+1}$
    while not line_search_condition do
        $\bar{\alpha}_k = \bar{\alpha}_k / 2$
        update $x_{k+1} = x_k - \bar{\alpha}_k g_k$
        compute $f(x_{k+1})$, $g_{k+1}$
    end while
    update $k = k + 1$
end while
```

Starting from Algorithm 4, we implemented the SDA and SDC variants.
When a new approximation of the Cauchy step, given by the quadratic interpolation, becomes available, Formulas (2.33)-(2.20) are computed and used as new

steplengths, respecting the scheme presented in Chapter 2. The SDA and SC methods for general non-quadratics will be denoted by GSDA and GSDC respectively.

To have a brief overview of the methods, we analyse the behaviour of the GSDA and GSDC on some problem contained in the CUTEr package [39]. In particular, we analysed the following problems:

1. AKIVA, CUTEr classification code OUR2-AN-2-0
2. BARD, CUTEr classification code SUR2-AN-3-0
3. DENSCHND, CUTEr classification code SUR2-AN-3-0
4. DENSCHNE, CUTEr classification code SUR2-AN-3-0

The results are given in Table 4.2. We can see that the GSDA and GSDC methods show a promising behaviour. For this reason future works will be devoted in the design of a better nonlinear framework, to compare the methods with the most efficient limited memory Conjugate Gradient and the CG_DESCENT methods [40, 41, 42].

Problem	GSD	GSDA	GSDC
AKIVA	6422	30	29
BARD	7127	258	235
DENSCHND	2207	266	249
DENSCHNE	1369	75	84

Table 4.2. CUTEr problems, number of iterations for the GSD, GSDA and GSDC methods.

Bibliography

[1] Akaike, H. On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method. *Annals of the Institute of Statistical Mathematics*, **11** (1959), 1.

[2] Armijo, L. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, **16** (1966), 1.

[3] Ascher, U., van den Doel, K., Huang, H., and Svaiter, B. Gradient descent and fast artificial time integration. *ESAIM: Mathematical Modelling and Numerical Analysis*, **43** (2009), 689.

[4] Barzilai, J. and Borwein, J. M. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, **8** (1988), 141.

[5] Beck, A. and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, **2** (2009), 183.

[6] Benvenuto, F., Zanella, R., Zanni, L., and Bertero, M. Nonnegative least-squares image deblurring: improved gradient projection approaches. *Inverse Problems*, **26** (2010), 025004.

[7] Berisha, S. and Nagy, J. G. Iterative methods for image restoration. In *Academic Press Library in Signal Processing: Volume 4. Image, Video Processing and Analysis, Hardware, Audio, Acoustic and Speech Processing* (edited by R. Chjellappa and S. Theodoridis), chap. 7, pp. 193–247. Academic Press, first edn. (2014).

[8] Bertero, M. and Boccacci, P. *Introduction to inverse problems in imaging.* CRC press (1998).

[9] Bertsekas, D. P. On the goldstein-levitin-polyak gradient projection method. *IEEE Transactions on Automatic Control*, **21** (1976), 174.

[10] Biemond, J., Lagendijk, R. L., and Mersereau, R. M. Iterative methods for image deblurring. *Proceedings of the IEEE*, **78** (1990), 856.

[11] Birgin, E. G., Martínez, J. M., and Raydan, M. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, **10** (2000), 1196.

[12] Bonettini, S., Zanella, R., and Zanni, L. A scaled gradient projection method for constrained image deblurring. *Inverse problems*, **25** (2009), 015002.

[13] Cauchy, A. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, **25** (1847), 536.

[14] Cornelio, A., Porta, F., Prato, M., and Zanni, L. On the filtering effect of iterative regularization algorithms for linear least-squares problems. *Inverse Problems*, **29** (2013), 125013.

[15] Dai, Y. H. Alternate step gradient method. *Optimization*, **52** (2003), 395.

[16] Dai, Y. H. and Fletcher, R. On the asymptotic behaviour of some new gradient methods. *Mathematical Programming*, **103** (2005), 541.

[17] Dai, Y. H. and Fletcher, R. Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming. *Numerische Mathematik*, **100** (2005), 21.

[18] Dai, Y. H., Hager, W. W., Schittkowski, K., and Zhang, H. The cyclic Barzilai-Borwein method for unconstrained optimization. *IMA Journal of Numerical Analysis*, **26** (2006), 604.

[19] Dai, Y. H. and Liao, L. Z. R-linear convergence of the Barzilai and Borwein gradient method. *IMA Journal of Numerical Analysis*, **22** (2002), 1.

[20] Dai, Y. H. and Yang, X. A new gradient method with an optimal step-size property. *Research report, Institute of Computational Mathematics and Scientific/Engineering Computing*, (2001).

[21] Dai, Y. H. and Yang, X. A new gradient method with an optimal stepsize property. *Computational optimization and applications*, **33** (2006), 73.

[22] Dai, Y. H. and Yuan, Y. X. Alternate minimization gradient method. *IMA Journal of Numerical Analysis*, **23** (2003), 377.

[23] Dai, Y. H. and Yuan, Y. X. Analysis of monotone gradient methods. *Journal of Industrial and Management Optimization*, **1** (2005), 181.

[24] De Asmundis, R., di Serafino, D., Hager, W. W., Toraldo, G., and Zhang, H. An efficient gradient method using the Yuan steplength. Tech. Rep. available from Optimization Online (http://www.optimization-online.org/DB_HTML/2013/10/4098.html) (2013).

[25] De Asmundis, R., di Serafino, D., and Landi, G. On the regularizing behaviour of recent gradient methods in the resolution of linear ill-posed problems. (In preparation).

[26] De Asmundis, R., di Serafino, D., Riccio, F., and Toraldo, G. On spectral properties of steepest descent methods. *IMA Journal of Numerical Analysis*, **33** (2013), 1416.

[27] Dennis, J. J. E. and Schnabel, R. B. *Numerical methods for unconstrained optimization and nonlinear equations*, vol. 16. Siam (1983).

[28] Elman, H. C. and Golub, G. H. Inexact and preconditioned Uzawa algorithms for saddle point problems. *SIAM Journal on Numerical Analysis*, **31** (1994), 1645.

[29] Engl, H., Hanke, M., and Neubauer, A. *Regularization of inverse problems*, vol. 375. Springer (1996).

[30] Figueiredo, M., Nowak, R., and Wright, S. Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems. *IEEE J. Sel. Top. Signal Process.*, **1** (2007), 586.

[31] Fletcher, R. Low storage methods for unconstrained optimization. *Lectures in Applied Mathematics (AMS)*, **26** (1990), 165.

[32] Fletcher, R. On the Barzilai-Borwein method. In *Optimization and control with applications*, pp. 235–256. Springer (2005).

[33] Fletcher, R. A limited memory steepest descent method. *Math. Program., Ser. A*, **135** (2012), 413.

[34] Frassoldati, G., Zanni, L., and Zanghirati, G. New adaptive stepsize selections in gradient methods. *J. Ind. Manag. Optim.*, **4** (2008), 299.

[35] Friedlander, A., Martínez, J., Molina, B., and Raydan, M. Gradient method with retards and generalizations. *SIAM Journal on Numerical Analysis*, **36** (1998), 275.

[36] Friedlander, A. and Martínez, J. M. On the maximization of a concave quadratic function with box constraints. *SIAM Journal on Optimization*, **4** (1994), 177.

[37] Galligani, E., Ruggiero, V., and Zanni, L. Variable projection methods for large-scale quadratic optimization in data analysis applications. In *Equilibrium Problems and Variational Models*, pp. 185–211. Springer (2003).

[38] Goldstein, A. A. Convex programming in hilbert space. *Bulletin of the American Mathematical Society*, **70** (1964), 709.

[39] Gould, N. I., Orban, D., and Toint, P. L. CUTEr and SifDec: A constrained and unconstrained testing environment, revisited. *ACM Transactions on Mathematical Software (TOMS)*, **29** (2003), 373.

[40] Hager, W. W. and Zhang, H. A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM Journal on Optimization*, **16** (2005), 170.

[41] Hager, W. W. and Zhang, H. Algorithm 851: Cg_descent, a conjugate gradient method with guaranteed descent. *ACM Transactions on Mathematical Software (TOMS)*, **32** (2006), 113.

[42] Hager, W. W. and Zhang, H. The limited memory conjugate gradient method. *SIAM Journal on Optimization*, **23** (2013), 2150.

[43] Hanke, M. *Conjugate Gradient Type Methods for Ill-Posed Problems*. Pitman Research Notes in Mathematics. Longman Scientific & Technical, Harlow, Essex (1995).

[44] Hansen, P. Regularization tools: A Matlab package for analysis and solution of discrete ill-posed problems. *Numer. Algorithms*, **6** (1994), 1.

[45] Hansen, P. *Rank-Deficient and Discrete Ill-Posed Problems*. SIAM, Philadelphia (1998).

[46] Huang, H. and Ascher, U. Faster gradient descent and the efficient recovery of images. *Vietnam Journal of Mathematics*, (2013), 1.

[47] Levitin, E. S. and Polyak, B. T. Constrained minimization methods. *USSR Computational mathematics and mathematical physics*, **6** (1966), 1.

[48] Luenberger, D. G. *Linear and nonlinear programming*. Springer (2003).

[49] Moré, J. J. and Toraldo, G. Algorithms for bound constrained quadratic programming problems. *Numerische Mathematik*, **55** (1989), 377.

[50] Moré, J. J. and Toraldo, G. On the solution of large quadratic programming problems with bound constraints. *SIAM Journal on Optimization*, **1** (1991), 93.

[51] Morozov, V. *Regularization Methods for Ill-Posed Problems*. CRC Press, Boca Raton, FL (1993).

[52] Nagy, J. and Palmer, K. Steepest descent, CG, and iterative regularization of ill-posed problems. *BIT*, **43** (2003), 1003.

[53] Nagy, J. G. and Palmer, K. Quasi-Newton methods for image restoration. In *Optical Science and Technology, the SPIE 49th Annual Meeting*, pp. 412–422. International Society for Optics and Photonics (2004).

[54] Nagy, J. G., Palmer, K., and Perrone, L. Iterative methods for image deblurring: a matlab object-oriented approach. *Numerical Algorithms*, **36** (2004), 73.

[55] Nocedal, J., Sartenaer, A., and Zhu, C. On the behavior of the gradient norm in the steepest descent method. *Computational Optimization and Applications*, **22** (2002), 5.

[56] Nocedal, J. and Wright, S. J. *Conjugate gradient methods*. Springer (2006).

[57] Polyak, B. T. The conjugate gradient method in extremal problems. *USSR Computational Mathematics and Mathematical Physics*, **9** (1969), 94.

[58] Porta, F., Cornelio, A., Zanni, L., and Prato, M. Filter factor analysis of scaled gradient methods for linear least squares. In *Journal of Physics: Conference Series*, vol. 464. IOP Publishing (2013).

[59] Raydan, M. On the Barzilai and Borwein choice of steplength for the gradient method. *IMA Journal of Numerical Analysis*, **13** (1993), 321.

[60] Raydan, M. and Svaiter, B. F. Relaxed steepest descent and Cauchy-Barzilai-Borwein method. *Computational Optimization and Applications*, **21** (2002), 155.

[61] Schwartz, A. and Polak, E. Family of projected descent methods for optimization problems with simple bounds. *Journal of Optimization Theory and Applications*, **92** (1997), 1.

[62] Serafini, T., Zanghirati, G., and Zanni, L. Gradient projection methods for quadratic programs and applications in training support vector machines. *Optimization Methods and Software*, **20** (2005), 353.

[63] Vogel, C. R. *Computational methods for inverse problems*, vol. 23. Siam (2002).

[64] Wolfe, P. Convergence conditions for ascent methods. *Siam Review*, **11** (1969), 226.

[65] Wright, S. J. Implementing proximal point methods for linear programming. *Journal of optimization Theory and Applications*, **65** (1990), 531.

[66] Yuan, Y. A new stepsize for the steepest descent method. *Journal of Computational Mathematics*, **24** (2006), 149.

[67] Yuan, Y. Step-sizes for the gradient method. *AMS/IP Studies in Advanced Mathematics*, **42** (2008), 785.