SAPIENZA
UNIVERSITÀ DI ROMA

# Advances in large scale unconstrained optimization: novel preconditioning strategies for Nonlinear Conjugate Gradient methods and new developments in Newton-Krylov methods

PhD in Automatica, Bioengineering and Operations Research

Curriculum in Operations Research – XXX Course


Candidate

Andrea Caliciotti

ID number 1310655


Thesis Advisors

Prof. Giovanni Fasano

Prof. Massimo Roma

A thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Automatica, Bioengineering and Operations Research

January 30, 2018

Thesis defended on 19 February 2018
in front of a Board of Examiners composed by:

Prof. Raffaele Pesenti (chairman)

Prof. Costanzo Manes

Prof. Mauro Ursino

---

**Advances in large scale unconstrained optimization: novel preconditioning strategies for Nonlinear Conjugate Gradient methods and new developments in Newton-Krylov methods**

Ph.D. thesis. Sapienza – University of Rome

ISBN: 000000000-0



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

Version: January 30, 2018

Author's email: andrea.caliciotti@uniroma1.it

# Acknowledgements

I owe my deepest gratitude to my Supervisors, Professor Giovanni Fasano and Professor Massimo Roma for the useful and interesting things they taught me during my PhD program, for their tireless guidance and patience, and for their meaningful assistance. Without these "ingredients", PhD program would have been hardly completed.
I am deeply grateful to Professor Mehiddin Al-Baali and to Professor Stephen G. Nash with whom we (my Supervisors and I) have cooperated.
I want to express my gratitude to the revisors of this PhD Thesis, Professor Gerardo Toraldo and Professor Luca Zanni.
I would like to thank my parents, Libera Di Bari and Piero Caliciotti, and my brother Valerio for their moral and material assistance. Special mention goes to my grandfather Giuseppe Di Bari, passed away recently, who would be proud of what I have done.
Finally, I would also like to thank myself for not giving up, keeping my way, in my PhD program.

# Common Notations

- $x \in \mathbb{R}^n$ is the column vector of the unknowns
- $x_k \in \mathbb{R}^n$ is $k$-th vector of the sequence $\{x_k\}$
- $I \in \mathbb{R}^{n \times n}$ is the identity matrix of size $n$
- $e_k \in \mathbb{R}^n$ is the $k$-th column of $I$
- $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is a real valued function over $\mathbb{R}^n$
- $f(x_k) = f_k$ is the value of $f$ at $x_k$
- $\nabla f \in \mathbb{R}^n$ is the gradient of $f$
- $\nabla f(x_k) = g(x_k) = g_k$ is the gradient of $f$ at $x_k$
- $\nabla^2 f \in \mathbb{R}^{n \times n}$ is the Hessian matrix of $f$
- $\nabla^2 f(x_k)$ is the Hessian matrix of $f$ at $x_k$
- $\|x\|$ is the Euclidean norm of $x$
- $\|x\|_\infty$ is the infinity norm of $x$
- $\|A\|_F$ is the Frobenius norm of $A$, with $A \in \mathbb{R}^{n \times n}$
- $A \succ 0$ indicates that matrix $A \in \mathbb{R}^{n \times n}$ is positive definite
- $A \succeq 0$ indicates that matrix $A \in \mathbb{R}^{n \times n}$ is positive semidefinite
- $\|x\|_A$, with $A \succ 0$, is the A-norm of $x$, i.e. $\|x\|_A = (x^T A x)^{\frac{1}{2}}$
- $\lambda_M(A)$ is the largest eigenvalue of the matrix $A \in \mathbb{R}^{n \times n}$
- $\lambda_m(A)$ is the smallest eigenvalue of the matrix $A \in \mathbb{R}^{n \times n}$
- $\kappa(A)$ is the condition number of the matrix $A \in \mathbb{R}^{n \times n}$
- $\bigoplus$ indicates the direct sum of matrices.

# List of Figures

# List of Tables

# Contents

# Abstract

In this thesis we propose new iteratively constructed preconditioners, to be paired with Conjugate Gradient-type algorithms, in order to efficiently solve large scale unconstrained optimization problems. On this guideline, the central thread of this thesis is the use of conjugate directions given by Conjugate Gradient or SYMMBK algorithms.

To this aim, in Chapter 1 we recall some results about iterative methods for solving linear systems. In particular, we introduce both the Conjugate Gradient method and the Lanczos process. Finally, the idea of preconditioning is given and the well known Preconditioned Conjugate Gradient method is provided. In Chapter 2 we deal with large scale unconstrained optimization problems, recalling the Nonlinear Conjugate Gradient (NCG) method and its preconditioned version, the quasi-Newton methods, including the damped techniques, and finally the linesearch-based Inexact Newton methods.

The main contribution in this thesis is given by Chapters 3-7. In particular, Chapter 3 provides new preconditioners to be used within the NCG method. This class of preconditioners draws inspiration from quasi-Newton updates, in order to obtain a good approximation of the inverse of the Hessian matrix. On detail, at the current iteration of the NCG we consider some preconditioners based on new low-rank quasi-Newton symmetric updating formulae, obtained as by-product of the NCG method at the previous steps. In particular, we propose five classes of preconditioners: in the first three classes the major drawback is that preconditioner might not be positive definite. In the fourth class, a preconditoner that is positive definite and satisfies the *secant equation* at the *current* iteration is given. Finally, in the last class, a preconditoner that is positive definite and satisfies the *modified secant equation* at *each* iteration is provided. However, in general, the search direction $p_k$ we compute at iteration $k$ of NCG could be not well scaled, which may introduce some ill-conditioning when applying Preconditioned Nonlinear Conjugate Gradient (PNCG). In order to reduce the scaling problem, in Chapter 4 we focus on the use of damped techniques within NCG methods. Damped techniques were introduced by Powell and recently reproposed by Al-Baali, in the framework of quasi-Newton methods. New damped techniques are proposed and are embedded within a class of preconditioners described in Chapter 3 (see Section 3.3.4). In Chapter 5, by referring to the Polak-Ribière (PR) method, we firstly give theoretical results on the global convergence for an effective PNCG algorithm. Secondly, we investigate the use of a damped vector in the definition of the scalar $\beta_k$, hence affecting the definition of the search direction and producing a modified NCG/PNCG method. On this purpose, we prove that some global convergence properties still hold for

the modified damped Polak-Ribière (D-PR-PNCG) method, while substantially preserving numerical performance. In Chapters 6-7 we focus on linesearch-based Newton-Krylov methods for solving large scale unconstrained optimization problems. In particular, in Chapter 6 we try to improve the efficiency of Newton-Krylov methods by proposing an *adaptive truncation rule*, for deciding the maximum number of inner iterations allowed at each outer iteration. The *adaptive truncation rule* is tested both within the unpreconditioned and the preconditioned framework proposed in [48]. Finally, Chapter 7 represents a work in progress: on detail, starting from [48] we deal with a class of preconditioners specifically suited for *large* indefinite linear systems, and may be obtained as *by-product* of Krylov subspace solvers, as well as by applying L-BFGS updates. In particular, in Chapter 7, only some preliminaries and the structure of our class of preconditioners, namely AINV$\mathcal{K}$, are provided. At the end, conclusions are given.

The material presented in this work has led to the following publications:

- A. Caliciotti, G. Fasano and M. Roma, *Preconditioning strategies for nonlinear conjugate gradient methods, based on quasi-Newton updates*, **The American Institute of Physics (AIP) Conference Proceedings**, vol. 1776, pp. 0900071-0900074, 2016.
- A. Caliciotti, G. Fasano and M. Roma, *Novel preconditioners based on quasi-Newton updates for nonlinear conjugate gradient methods*, **Optimization Letters**, vol. 11, pp. 835-853, 2017.
- M. Al-Baali, A. Caliciotti, G. Fasano and M. Roma, *Exploiting damped techniques for nonlinear conjugate gradient methods*, **Mathematical Methods of Operations Research**, vol. 86, pp. 501-522, 2017.
- A. Caliciotti, G. Fasano and M. Roma, *Preconditioned nonlinear conjugate gradient methods based on a modified secant equation*, **Applied Mathematics and Computation**, vol. 318, pp. 196-214, 2018.
- A. Caliciotti, G. Fasano, S. G. Nash and M. Roma, *An adaptive truncation criterion, for linesearch-based truncated Newton methods in large scale nonconvex optimization*, **Operations Research Letters**, vol. 46, pp. 7-12, 2018.
- A. Caliciotti, G. Fasano, S. G. Nash and M. Roma, *Data and performance profiles applying an adaptive truncation criterion, within linesearch-based truncated Newton methods, in large scale nonconvex optimization*, **Data in Brief**, vol. 17, pp. 246-255, 2018.
- M. Al-Baali, A. Caliciotti, G. Fasano and M. Roma, *Quasi-Newton based preconditioning and damped quasi-Newton schemes, for nonlinear conjugate gradient methods*, Accepted for Publication on **Springer Proceedings (PROMS)**, 2018.

# Motivations

A general symmetric linear system can be written in the form

$$Ax = b,$$

where

- $A \in \mathbb{R}^{n \times n}$ is a nonsingular symmetric (dense) matrix;
- $b \in \mathbb{R}^n$ is a vector of the constant terms.

In literature, the solution of linear systems is typically pursued by adopting one of the following two approaches:

- Direct methods;
- Iterative methods.

Direct methods are based on the idea to solve the problem by a finite sequence of operations. In the case of absence of rounding errors, direct methods provide an exact solution. Iterative methods generate a sequence of improving approximate solutions according to a suitable rule in which the $k$-th approximation is derived from the previous ones. When $n$ is large, due to the storage of full matrix and the use of factorization techniques, direct methods are very expensive. On this guideline iterative methods might be used rather than direct methods.

Given the linear system and an initial approximation $x_1 \in \mathbb{R}^n$, iterative methods try to produce a sequence $\{x_k\}$, $k = 1, 2, ...$, such that, under suitable hypotheses, $\{x_k\}$ converges in some sense to the solution of the linear system. Since $x_k$ is an approximation of the solution $x^* = A^{-1}b$, iterative methods may exhibit truncation errors (in addition to rounding errors). This issue highlights that a stopping criterion is needed.

However, in some applications, iterative methods often fail to converge "quickly" and preconditioning is necessary, though not always sufficient, to attain convergence in a reasonable amount of time (see e.g. [19]). It is widely recognized that preconditioning is among the critical ingredients in the development of efficient solvers for challenging problems in scientific computation involving symmetric linear systems, and that the importance of preconditioning is destined to increase even further. In particular, in [19], the author proposes a detailed overview of preconditioning strategies for large linear systems, mainly focusing on incomplete factorization techniques (ILU) and sparse approximate inverses (SPAI).

Briefly speaking, the term preconditioning refers to transforming the linear system $Ax = b$ into another system with "more favorable" properties for iterative solution. A preconditioner is a matrix that affects such a transformation. Other details about preconditioning will be described in Section 1.2.

In large number of real world applications, preconditioning techniques have been successfully applied. Some examples are:

- electromagnetism (see e.g. [1], [12]);
- geophysics (see e.g. [27], [28], [44]);
- hydrodynamics (see e.g. [47], [76], [80]);
- meteorology (see e.g. [57], [59], [64], [107], [110]);
- structural mechanics (see e.g. [21], [58], [70], [73], [81], [82]).

The use of preconditioning techniques can be also applied for parallel computation. For example, in [75] the authors adopt a parallel computation technique for rigid-viscoplastic approach, in order to increase the computational efficiency. On detail, the domain decomposition algorithm and Preconditioned Conjugate Gradient (described in Section 1.2.1) iterative solver is used for parallel computation, proposing a new block Jacobi preconditioner. Another example can be seen in [73]: in this paper the authors present a parallel preconditioned iterative solver for large sparse symmetric positive definite linear systems. The preconditioner is constructed as a proper combination of advanced preconditioning strategies. In particular, it can be formally seen as being of domain decomposition type with algebraically constructed overlap. Similar to the classical domain decomposition technique, inexact subdomain solvers are used, based on incomplete Cholesky factorization.

Very interesting is the use of preconditioners for the numerical solution of the Helmholtz equation for two-dimensional (see [44]) and three-dimensional ([27], [28]) applications in geophysics. The solution of heterogeneous Helmholtz problems is recognized as of high interest in many application fields.

Solution of linear systems of equations arising in the discretization of the incompressible Navier-Stokes equations remains an active area of research. On this guidelines, good preconditioning techniques for flow computations attracted much attention (see [47], [76], [80]).

The literature on preconditioning is extremely rich: in particular, many suggestions have been made concerning either purely algebraic or application-based preconditioners able to exploit the structure of the matrix. A novel approach, known as limited memory preconditioner, markedly differs from purely preconditioning, in the sense that a strategy to update a given preconditioner exploiting existing information is proposed. This existing information can be for instance the knowledge of matrix-vector products or an approximate invariant subspace. Borrowing idea from numerical optimization literature, update strategies have been provided in the context of the solution of nonlinear equations with quasi-Newton based methods (see Section 2.1.2). When the coefficient matrices are symmetric positive definite, in [83] the authors have proposed a preconditioner to be used in combination with the conjugate gradient method (see Section 1.1.1), which has the form of a limited

memory quasi-Newton matrix (see, e.g., [86] and [96] for earlier attempts). In [60] the authors Gratton, Sartenaer and Tshimanga have similarly defined a class of limited memory preconditioners based on limited memory quasi-Newton formulas, that ensures good spectral properties of the preconditioned matrix. These preconditioners require a small number $k$ of linearly independent vectors. This family can be seen as a block variant of the BFGS updating formula (see Section 2.1.2) for quadratic problems. An extension of this class of limited memory preconditioners for the solution of sequences of linear systems with symmetric indefinite matrices has been provided in [58].

In this thesis, we propose novel general-purpose preconditioners iteratively constructed to be used within Conjugate Gradient-type algorithms, in order to efficiently solve large scale unconstrained optimization problems: on this guideline our preconditioners are matrix-free.

Future work will be to apply our preconditioners to solve real problems, in order to further confirm the effectiveness of our proposals.

# Chapter 1

# Iterative Methods for solving Symmetric Linear Systems

In this chapter we recall some relevant results about iterative methods for solving linear systems. In Section 1.1 we recall the most commonly used class of iterative methods for solving large linear systems, the *Krylov subspace methods*. Finally, in Section 1.2 we introduce preconditioning and one of its applications. Most of the material of this chapter is taken from [23], [36], [54], [63], [95], [108].

## 1.1 Krylov subspace methods

The basic idea of iterative methods is replacing original system

$$Ax = b \tag{1.1}$$

by one that can be easily solved. Starting from (1.1) we try to solve an easier system

$$Kx_1 = b, \tag{1.2}$$

where $K \in \mathbb{R}^{n\times n}$ and $x_1 \in \mathbb{R}^n$ are an approximation of $A$ and $x^*$ (the solution of (1.1)), respectively. At any step, considering the correction $z \in \mathbb{R}^n$, we want that

$$A(x_1 + z) = b. \tag{1.3}$$

This leads to a new system

$$Az = b - Ax_1. \tag{1.4}$$

Solving (1.4) for $z$ through the modified system, we obtain the vector $z_1$ such that

$$Kz_1 = b - Ax_1. \tag{1.5}$$

The solution $x_2 = x_1 + z_1$ is considered to some extent a better approximation of $x^*$. Moreover, at the general $i$-th step, from (1.1)-(1.5) it follows that:

$$x_{i+1} = x_i + z_i = x_i + K^{-1}(b - Ax_i) = x_i + \bar{b} - \bar{A}x_i, \tag{1.6}$$

where $\bar{b} = K^{-1}b$ and $\bar{A} = K^{-1}A$. Obviously, we will never compute $K^{-1}$, and writing $\bar{b} = K^{-1}b$ we intend the vector $\bar{b}$ such that $K\bar{b} = b$ (similarly for $\bar{A} = K^{-1}A$).

In order to simplify (1.6), we consider $K = I$ which yields for (1.6) the well known Richardson iteration:

$$x_{i+1} = x_i + z_i = x_i + b - Ax_i = b + (I - A)x_i = x_i + r_i, \tag{1.7}$$

where $r_i = b - Ax_i$ is the residual.

From now on, in order to reduce the complexity of the formalae, we can assume without loss of generality $x_1 = 0$.

On this guideline, our aim is to identify a particular subspace on which, at each iteration, the approximate solutions of (1.1) are sought. Iterating the basic Richardson iteration (1.7) we note that ($x_1 = 0$)

$$x_{i+1} = x_1 + r_1 + r_2 + \ldots + r_i = r_1 + r_2 + \ldots + r_i = \sum_{j=1}^{i} (I - A)^j r_1 \tag{1.8}$$

that is

$$x_{i+1} \in span\left\{r_1, Ar_1, A^2 r_1, \ldots, A^i r_1\right\}. \tag{1.9}$$

**Definition 1.1.1.** *Given a nonzero vector $v \in \mathbb{R}^n$ and a symmetric matrix $A \in \mathbb{R}^{n\times n}$,*

$$K_m(A, v) = span\left\{v, Av, A^2 v, \ldots, A^{m-1} v\right\} \tag{1.10}$$

*is called the m-dimensional Krylov subspace.*

Krylov subspace is a particular subspace of $\mathbb{R}^n$ given by all linear combinations of vectors $v, Av, A^2 v, \ldots, A^{m-1} v$.

In this regard, setting $m = i + 1$ and $v = r_1$ we can state that

$$x_{i+1} \in K_{i+1}(A, r_1). \tag{1.11}$$

Since optimality usually refers to some sort of projection, Krylov methods are also called Krylov projection methods: in fact, they seek an approximate solution $x_{i+1}$ of the linear system in (1.1) that belongs to $K_{i+1}(A, r_1)$. Krylov subspace methods can be divided into four different classes: each class is characterized by an appropriate criterion to determine $x_{i+1}$. On this guideline, we report the four Krylov methods approaches (see [108]):

- the Ritz-Galerkin approach: its aim is to find $x_{i+1}$ such that the residual $r_{i+1}$ is orthogonal to the current Krylov subspace, that is $r_{i+1} = b - Ax_{i+1} \perp K_{i+1}(A, r_1)$;
- the Minimum Norm Residual approach: its aim is to find $x_{i+1}$ such that $\|b - Ax_{i+1}\|$ is minimized over $K_{i+1}(A, r_1)$;
- the Petrov-Galerkin approach: its aim is to find $x_{i+1}$ such that the residual $r_{i+1} = b - Ax_{i+1}$ is orthogonal to some other suitable $(i + 1)$-dimensional subspace;
- the Minimum Norm Error approach: its aim is to find $x_{i+1} \in A^T K_{i+1}(A^T, r_1)$ such that the error Euclidean norm $\|x_{i+1} - x\|_2$ is minimized, where $x \in K_{i+1}(A, r_1)$.

In this work we focus on Ritz-Galerkin approach, recalling the two well known and most common methods:

- Conjugate Gradient (CG) method;
- Lanczos process.

### 1.1.1 The Conjugate Gradient (CG) method

In this section we are going to recall the CG method (see [69]) along with some theoretical properties. We will carry on our analysis with reference to the problem of minimizing a convex quadratic function. Consider the unconstrained optimization problem

$$\min_{x\in\mathbb{R}^n} \varrho(x) \tag{1.12}$$

where:

- $\varrho : \mathbb{R}^n \longrightarrow \mathbb{R}$ is the convex quadratic function, that is $\varrho(x) = \frac{1}{2}x^T Ax - b^T x$;
- $A \in \mathbb{R}^{n\times n}$ is a nonsingular symmetric matrix;
- $b \in \mathbb{R}^n$ is the column vector of constant values;

As well known, $\varrho(x)$ has a minimum point if and only if both $A \succeq 0$ and there exists at least a point $x^*$ such that:

$$\nabla\varrho(x^*) = Ax^* - b = 0. \tag{1.13}$$

We note that if $A \succ 0$, $\varrho(x)$ is strictly convex and has a unique global minimum point $x^* = A^{-1}b$, that is the solution of linear system (1.1). CG method has been introduced as iterative method to solve linear system in (1.1): obviously, solving (1.1) with $A \succ 0$ is equivalent to the problem of minimizing a strictly convex quadratic function.

The main feature of CG method is to easily generate a set of vectors with a property known as *conjugacy*.

**Definition 1.1.2.** *Given a symmetric matrix $A \in \mathbb{R}^{n\times n}$, two nonzero vectors $p_i, p_j \in \mathbb{R}^n$ are said to be conjugate with respect to $A$ if*

$$p_i^T A p_j = 0. \tag{1.14}$$

Since we study the problem of minimizing quadratic functions, hereafter we consider matrix $A$ symmetric and at least positive semidefinite. Furthermore, if $A = I$, Definition 1.1.2 yields the definition of the orthogonal vectors $p_i, p_j$. It is immediate to prove that any set of nonzero vectors *conjugate* with respect to matrix $A$ is also linearly independent (see [63]):

**Proposition 1.1.3.** *Let us consider a symmetric and positive definite matrix $A \in \mathbb{R}^{n\times n}$. Let $\{p_1, p_2, \ldots, p_k\}$ be a set of nonzero conjugate vectors with respect to $A$. Then, $\{p_1, p_2, \ldots, p_k\}$ are linearly independent.*

We can now introduce the CG scheme. In particular CG method, starting from an arbitrary initial point $x_1 \in \mathbb{R}^n$, generates the sequence of iterates $x_{k+1} = x_k + \alpha_k p_k$, where $p_k \in \mathbb{R}^n$ is the search direction and $\alpha_k \in \mathbb{R}$ is a positive steplength obtained by an exact linesearch procedure.

**CG algorithm**

**Data**: $x_1 \in \mathbb{R}^n$;

Set $k = 1$, $g_1 = Ax_1 - b$ and $p_1 = -g_1$;

**While** $g_k \neq 0$

$$\alpha_k = \frac{\|g_k\|^2}{p_k^T A p_k} \tag{1.15}$$

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

$$g_{k+1} = g_k + \alpha_k A p_k$$

$$\beta_{k+1} = \frac{\|g_{k+1}\|^2}{\|g_k\|^2} \tag{1.16}$$

$$p_{k+1} = -g_{k+1} + \beta_{k+1} p_k$$

$$k = k + 1$$

**End While**

More interesting is that CG method can be viewed as an iterative method in which in a finite number of iterations the residual $r_k = b - Ax_k = -g_k$ annihilates.

Now we recall an important result for the CG (see [95]).

**Proposition 1.1.4.** *Suppose that iterate $x_k$ generated by CG scheme does not coincide with the solution point $x^*$ of* (1.1)*. Then, the following properties hold:*

$$r_k^T r_i = 0, \qquad i = 1, 2, \ldots, k-1, \tag{1.17}$$

$$g_k^T g_i = 0, \qquad i = 1, 2, \ldots, k-1, \tag{1.18}$$

$$span\{r_1, r_2, \ldots, r_k\} = span\left\{r_1, Ar_1, \ldots, A^{k-1} r_1\right\}, \tag{1.19}$$

$$span\{p_1, p_2, \ldots, p_k\} = span\left\{r_1, Ar_1, \ldots, A^{k-1} r_1\right\}, \tag{1.20}$$

$$p_k^T A p_i = 0, \qquad i = 1, 2, \ldots, k-1. \tag{1.21}$$

*The sequence $\{x_k\}$ converges to $x^*$ in at most $n$ steps and there exists $m \leq n-1$ such that $g_{m+1} = 0$.*

Proposition 1.1.4 ensures that the CG method:

- generates a set of linearly independent directions $\{p_1, p_2, \ldots, p_k\}$ and residuals $\{r_1, r_2, \ldots, r_k\}$ that form an orthogonal basis for a Krylov subspace;
- determines in a finite number of steps the minimum point of strictly convex quadratic functions or equivalently solves linear system (1.1), with $A \succ 0$, in at most $n$ steps.

Furthermore, a particular result is reported (see [63]).

**Proposition 1.1.5.** *Suppose that matrix $A$ has (n-k) eigenvalues in the range $[a, c]$ and $k$ eigenvalues greater than $c$. Then iterate $x_{k+1}$ generated by CG method is such that*

$$\|x_{k+1} - x^\star\|_A^2 \leq \left(\frac{c-a}{c+a}\right)^2 \|x_1 - x^\star\|_A^2. \tag{1.22}$$

This result shows that the effectiveness of CG methods depends directly on the eigenvalue distribution of $A$. In fact, we note that the smaller the range $[a, c]$ the better the approximation of optimal solution is obtained in few iterations.

Another convergence scheme for CG is based on Euclidean condition number of $A$, that is

$$\kappa(A) = \|A\|_2 \|A^{-1}\|_2 = \frac{\lambda_n}{\lambda_1}, \tag{1.23}$$

where $\lambda_1$ and $\lambda_n$ are respectively the smallest and the largest eigenvalue of $A$.

On this guideline, a convergence result provided by Luenberger (see [79]) is given.

**Proposition 1.1.6.** *If $\kappa(A)$ is the condition number of matrix $A$, then at the $k$-th iteration of CG algorithm we have*

$$\|x_{k+1} - x^\star\|_A \leq 2\left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}\right)^k \|x_1 - x^\star\|_A. \tag{1.24}$$

This result gives a large overestimate of the error $x_{k+1} - x^\star$, but it can be useful in those cases where the only information we have about $A$ is the estimation of the extreme eigenvalues $\lambda_1$ and $\lambda_n$. As immediate consequence of Proposition 1.1.5 and Proposition 1.1.6, we report the following result (see [63]).

**Corollary 1.1.7.** *Suppose that matrix $A$ has $k$ distinct eigenvalues. Then CG methods will terminate in at most $k$ iterations.*

As we have seen above, CG method can be used for solving linear system (1.1) with matrix $A$ positive definite. If matrix $A$ is indefinite, the algorithm could break down because $p_k^T A p_k$ (see (1.15)) might be nearly zero. A first attempt to try to use CG method also in the indefinite nonsingular case consists to solve the equivalent system $A^2 x = Ab$ of (1.1): in fact, we note that $A^2$ is positive definite. Nevertheless, this approach presents two disadvantages:

- the new system will be strongly ill-conditioned, being $\kappa(A^2) = [\kappa(A)]^2$;
- at each iteration two matrix-vector products are needed.

On this guideline, in order to cope also with indefinite linear systems, we introduce the well known Lanczos process: first when $A$ is positive definite, then in the indefinite case.

### 1.1.2 The Lanczos process

Lanczos process is a particular technique to estimate a good approximation of extreme eigenvalues of a matrix. This method is very useful in case of large scale problems. Given a nonsingular symmetric matrix $A \in \mathbb{R}^{n\times n}$, Lanczos process generates a sequence of symmetric tridiagonal matrices $T_k \in \mathbb{R}^{k\times k}$ (with $k \leq n$) such that their extreme eigenvalues converge monotonically to the extreme eigenvalues of $A$. From a theoretical point of view, Lanczos process is conceived as a method for tridiagonalizing the matrix $A$.

Hence, the Lanczos process is also an efficient tool to determine a good approximation of the solution of the large scale linear system (see (1.1)).

Given a symmetric matrix $A$ (not necessarily definite positive), we report the well know Lanczos scheme.

**Lanczos algorithm**

**Data**: $x_1 \in \mathbb{R}^n$;

Set $k = 1$, $u_1 = r_1 = b - Ax_1 = -g_1$, $q_1 = \frac{u_1}{\|u_1\|}$, $\delta_1 = q_1^T A q_1$ and $u_2 = Aq_1 - \delta_1 q_1$;

**For** $j = 2, \ldots, k-1$

$$\gamma_j = \|u_j\|$$
$$q_j = \frac{u_j}{\gamma_j}$$
$$\delta_j = q_j^T A q_j$$
$$u_{j+1} = Aq_j - \delta_j q_j - \gamma_j q_{j-1}$$

**End For**

$$\gamma_k = \|u_k\|$$
$$q_k = \frac{u_k}{\gamma_k}$$
$$\delta_k = q_k^T A q_k$$

After $k$ iterations, Lanczos process generates:

- $k$ vectors $q_1, \ldots, q_k$, named *Lanczos vectors*;
- $k$ scalars $\delta_1, \ldots, \delta_k$;
- $k-1$ scalars $\gamma_2, \ldots, \gamma_k$.

Defining $Q \in \mathbb{R}^{n\times k}$ a matrix whose columns are the Lanczos vectors, that is

$$Q_k = \begin{bmatrix} q_1 & \vdots & \vdots & q_k \end{bmatrix} \tag{1.25}$$

and the symmetric tridiagonal matrix $T_k \in \mathbb{R}^{k\times k}$

$$T_k = \begin{pmatrix} \delta_1 & \gamma_2 & & & \\ \gamma_2 & \delta_2 & \cdot & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \delta_{k-1} & \gamma_k \\ & & & \gamma_k & \delta_k \end{pmatrix}, \tag{1.26}$$

we can report the following result (see [36]).

**Proposition 1.1.8.** *Let be given a nonsingular symmetric matrix $A \in \mathbb{R}^{n\times n}$ and let $k \leq n$. According to the Lanczos algorithm, we obtain:*

$$AQ_k = Q_k T_k + \gamma_{k+1} q_{k+1} e_k^T, \tag{1.27}$$

$$Q_k^T A Q_k = T_k, \tag{1.28}$$

$$Q_k^T Q_k = I, \tag{1.29}$$

$$Q_k^T q_{k+1} = 0, \tag{1.30}$$

$$span\{q_1, q_2, \ldots, q_k\} = span\left\{r_1, Ar_1, \ldots, A^k r_1\right\}, \tag{1.31}$$

*Hence, the Lanczos process converges in at most n steps.*

#### 1.1.2.1 Lanczos process for positive definite systems

In this section our aim is to show how the Lanczos process can be used to solve linear systems in (1.1) with $A$ positive definite. As already mentioned, Lanczos process generates a sequence of orthonormal vectors $\{q_i\}$(the Lanczos vectors), $i = 1, \ldots, k$, which form a basis for the Krylov subspace $K_k(A, r_1)$.

If $A$ is positive definite, solving $Ax = b$ is equivalent to find a minimum point of the quadratic function $\varrho(x) = \frac{1}{2}x^T Ax - b^T x$ over $\mathbb{R}^n$, being $x^\star = A^{-1}b$ the unique minimum point of $\varrho$. On this guideline, an approximate minimum point of $\varrho$ can be viewed as an approximate solution of linear system $Ax = b$. A particular way to produce a sequence $\{x_k\}$ that converges to $x^\star$ consists to generate a sequence of orthonormal vectors. To do this, we can use the Lanczos vectors and define a minimum point $x_k$ of $\varrho$ over the subspace $span\{q_1, q_2, \ldots, q_k\}$. Roughly speaking, by the Lanczos process at step $k$ we try to seek a minimum point in a particular subspace (instead of $\mathbb{R}^n$). Since $x_k \in x_1 + span\{q_1, q_2, \ldots, q_k\}$, it results ($x_1 = 0$)

$$x_k = x_1 + Q_k y_k = Q_k y_k, \tag{1.32}$$

where $Q_k$ is the matrix in (1.25) and $y_k \in \mathbb{R}^k$ is a suitable vector. Considering (1.32), the quadratic function in (1.12) can be rewritten as

$$\varrho(y_k) = \frac{1}{2} y_k^T (Q_k^T A Q_k) y_k - b^T (Q_k y_k). \tag{1.33}$$

The problem of finding minimum points of $\varrho$ over the subspace $span\{q_1, \ldots, q_k\}$ consists to determine $y_k^\star$ such that

$$\nabla\varrho(y_k) = Q_k^T A Q_k y_k^\star - Q_k^T b = 0. \tag{1.34}$$

The espression in (1.34) is equivalent to solve the system

$$Q_k^T A Q_k y_k = Q_k^T b. \tag{1.35}$$

Considering that $Q_k^T A Q_k = T_k$ (see (1.28)), the linear system $Ax = b$ in (1.1) can be transformed in

$$\begin{aligned} T_k y_k &= Q_k^T b \\ x_k &= Q_k y_k. \end{aligned} \tag{1.36}$$

Firstly, we calculate $y_k^\star$ from $T_k y_k = Q_k^T b$, then we obtain $x_k^\star = Q_k y_k^\star$ that is an approximate solution of the original linear system in (1.1). Obviously we note that if $\gamma_k = 0$ for a certain $k$, the Lanczos process is arrested and the corresponding $x_k$ is the solution of the linear system $Ax_k = b$.

#### 1.1.2.2 Lanczos process for indefinite systems

After studying Lanczos process applied to solve linear systems in (1.1) with $A$ positive definite, we consider the indefinite case. As already stated, one of the advantages of using Lanczos process in place of CG algorithm is that the first one does not break down in the indefinite case. An efficient strategy to use Lanczos process for solving linear systems where $A$ is indefinite (similarly when $A$ is positive definite) is the following. Starting from vector $q_1$, without loss of generality, we define $q_1 = \frac{b}{\|b\|}$. Therefore, considering (1.30), the expression $T_k y_k = Q_k^T b$ in (1.36) becomes $T_k y_k = \|b\| e_1$. Thus, we can reformulate the system in (1.30) as follows:

$$\begin{aligned} T_k y_k &= \|b\| e_1 \\ x_k &= Q_k y_k. \end{aligned} \tag{1.37}$$

From a theoretical point of view, using Lanczos procedure to solve linear systems in (1.1), both in case of $A$ positive definite and in case of $A$ indefinite, is relatively easy.

### 1.1.3 Relationship between Lanczos process and CG method

In this section we recall the relationship between Lanczos process and CG method in the positive definite case. As we have seen, CG method computes an orthogonal basis $\{r_1, r_2, \ldots, r_k\}$ for the Krylov subspace $K_k(A, r_1)$. The Lanczos process generates an orthonormal basis $\{q_1, q_2, \ldots, q_k\}$ for the same Krylov subspace $K_k(A, r_1)$.

As long as the CG method does not break down (see [36]),

$$T_k = \begin{pmatrix} \frac{1}{\alpha_1} & \frac{\sqrt{\beta_1}}{|\alpha_1|} & & & & \\ \frac{\sqrt{\beta_1}}{|\alpha_1|} & \frac{1}{\alpha_2} + \frac{\beta_1}{\alpha_1} & \cdot & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \frac{1}{\alpha_{k-1}} + \frac{\beta_{k-2}}{\alpha_{k-2}} & \frac{\sqrt{\beta_{k-1}}}{|\alpha_{k-1}|} & \\ & & & \frac{\sqrt{\beta_{k-1}}}{|\alpha_{k-1}|} & \frac{1}{\alpha_k} + \frac{\beta_{k-1}}{\alpha_{k-1}} \end{pmatrix}, \tag{1.38}$$

and thus the Lanczos tridiagonal matrix $T_k$ is obtained as a trivial by-product of the CG parameters $\alpha_k$ in (1.15) and $\beta_k$ in (1.16).

Collating the Lanczos tridiagonal matrix $T_k$ in (1.26) with the tridiagonal matrix in (1.38) we get, for $j = 1, \ldots, k-1$, the following relations:

$$\delta_1 = \frac{1}{\alpha_1} \tag{1.39}$$

$$\delta_{j+1} = \frac{1}{\alpha_{j+1}} + \frac{\beta_j}{\alpha_j} \tag{1.40}$$

$$\gamma_{j+1} = \frac{\sqrt{\beta_j}}{|\alpha_j|}. \tag{1.41}$$

In order to complete the relationship between CG and Lanczos process, in the positive definite case, we report the following result in (see [36]).

**Proposition 1.1.9.** *Let be given a nonsingular positive definite symmetric matrix $A \in \mathbb{R}^{n\times n}$. Suppose that both CG and Lanczos algorithms adopt the same starting vector. Then the solution $x_k$ obtain by Lanczos procedure coincides with the solution $x_k$ obtained by CG algorithm.*

The latter result is very important because we can ensure that, from a theoretical point of view, in the positive definite case, Lanczos process and CG method are equivalent. Finally, an application of this relationship can be viewed in [55].

### 1.1.4 Decomposition of tridiagonal matrix $T_k$

Lanczos process is a powerful method for solving linear systems in (1.1). Unlike CG method, Lanczos algorithm does not break down if the matrix $A$ is indefinite. On the other hand, Lanczos scheme needs to decompose the tridiagonal matrix $T_k$.

#### 1.1.4.1 Decomposition for positive definite systems

In this paragraph, we study how to factorize efficiently the tridiagonal matrix $T_k$ in case of matrix $A$ positive definite. It is well known that if $A \succ 0$, using (1.28), matrix $T_k$ is positive definite (see [54]). To decompose the matrix $T_k$ we can use the banded version of Cholesky factorization (see [54])

$$T_k = L_k D_k L_k^T, \tag{1.42}$$

where $L_k \in \mathbb{R}^{k\times k}$ is a unit lower bidiagonal matrix and $D_k \in \mathbb{R}^{k\times k}$ is a diagonal matrix. Denoting

$$L_k = \begin{pmatrix} 1 & & & & \\ \mu_2 & 1 & & & \\ & \cdot & \cdot & & \\ & & \mu_{k-1} & 1 & \\ & & & \mu_k & 1 \end{pmatrix} \tag{1.43}$$

and

$$D_k = \begin{pmatrix} d_1 & & & & \\ & d_2 & & & \\ & & \cdot & & \\ & & & d_{k-1} & \\ & & & & d_k \end{pmatrix} \tag{1.44}$$

we get, for $j = 1, \ldots, k-1$, the following relations with the matrix $T_k$ (see (1.26) and (1.38)):

$$\begin{aligned} d_1 &= \frac{1}{\alpha_1} = \delta_1, \\ d_{j+1} &= \frac{1}{\alpha_{j+1}} = \delta_{j+1}, \\ \mu_{j+1} &= -\alpha_{j+1}\frac{\sqrt{\beta_{j+1}}}{|\alpha_{j+1}|} = \gamma_{j+1}. \end{aligned} \tag{1.45}$$

In order to compute $x_k$ in (1.37), we introduce the $n \times k$ matrix $C_k = \begin{bmatrix} c_1 & \vdots & \vdots & c_k \end{bmatrix}$ and the vector $\rho^{(k)} = (\rho_1, \ldots, \rho_k)^T$ such that

$$C_k L_k^T = Q_k \tag{1.46}$$

$$\rho^{(k)} = L_k^T y_k. \tag{1.47}$$

Hence, (1.37) can be rewritten as follows:

$$\begin{aligned} L_k D_k \rho^{(k)} &= \|b\| e_1 \\ C_k L_k^T &= Q_k \\ x_k &= C_k \rho^{(k)}. \end{aligned} \tag{1.48}$$

After some computations, from the equations (1.48), we obtain the following recursive formulae to calculate $x_k$ within the Lanczos algorithm:

$$c_k = q_k - c_{k-1}\mu_{k-1} \tag{1.49}$$

$$\rho_k = -\frac{\mu_{k-1} d_{k-1} \rho_{k-1}}{d_k} \tag{1.50}$$

$$x_k = x_{k-1} + \rho_k c_k. \tag{1.51}$$

#### 1.1.4.2 Decomposition for indefinite systems

When the matrix $A$ is indefinite, the Cholesky factorization in (1.42) may not exist. In this section we present the SYMMBK algorithm (see [34]) for the indefinite systems, based on the Bunch and Kaufman decomposition of $T_k$.

First of all, to describe the SYMMBK scheme, we must recall the Bunch and Kaufman decomposition. Suppose that for any $k > 0$, $T_k$ and $T_{k+1}$ in (1.37) are both singular. Consider the generic tridiagonal matrix $U_k$, that is

$$U_k = \begin{pmatrix} a_1 & b_1 & & & \\ c_1 & a_2 & b_2 & & \\ & \cdot & \cdot & \cdot & \\ & & c_{k-2} & a_{k-1} & b_{k-1} \\ & & & c_{k-1} & a_k \end{pmatrix}, \tag{1.52}$$

it is well known that the determinant of matrix $U_k$ can be computed using the following iterative formula:

$$\det(U_k) = a_k \det(U_{k-1}) - c_{k-1}b_{k-1}\det(U_{k-2}). \tag{1.53}$$

Applying this fact to $T_k$, we get that $T_{k+j}$, for all $j > 1$, will be singular. Obviously, this is not possible since $T_n$ is similar to $A$ which is nonsingular. Hence, if $T_{k+1}$ is singular, then both $T_k$ and $T_{k+2}$ are nonsingular. On the basis of these considerations, suppose that we are computing the Cholesky decomposition for the tridiagonal matrices $T_j$, for $j \leq k$. Consider the following matrix $T_{k+1}$, that is

$$T_{k+1} = \begin{pmatrix} \delta_1 & \gamma_2 & & & & \\ \gamma_2 & \delta_2 & \cdot & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \delta_{k-1} & \gamma_k & \\ & & & \gamma_k & \delta_k & \\ & & & & & 0 \end{pmatrix}, \tag{1.54}$$

where the $(k+1)$-th pivot element is zero. In this case, $T_{k+1}$ is singular and Cholesky decomposition breaks down. However, if we consider the tridiagonal nonsingular matrix $T_{k+2}$

$$T_{k+2} = \begin{pmatrix} \delta_1 & \gamma_2 & & & & & \\ \gamma_2 & \delta_2 & \cdot & & & & \\ & \cdot & \cdot & \cdot & & & \\ & & \cdot & \delta_{k-1} & \gamma_k & & \\ & & & \gamma_k & \delta_k & & \\ & & & & & 0 & \gamma_{k+2} \\ & & & & & \gamma_{k+2} & \delta_{k+2} \end{pmatrix} \tag{1.55}$$

and we take a $2 \times 2$ pivot instead of a single element, we can consider the $2 \times 2$ diagonal block

$$\begin{pmatrix} 0 & \gamma_{k+2} \\ \gamma_{k+2} & \delta_{k+2} \end{pmatrix}. \tag{1.56}$$

Since the determinant of (1.56) is $-\gamma_{k+2}^2 \neq 0$, the decomposition of $T_{k+2}$ in (1.55) depends directly on that of $T_k$ in (1.26), in order to skip the singularity of $T_{k+1}$ in (1.54). On this guideline we can introduce the Bunch and Kaufman Decomposition

$$T_k = S_k B_k S_k^T, \tag{1.57}$$

where $B_k$ is a block diagonal matrix with $1 \times 1$ or $2 \times 2$ diagonal blocks, and $S_k$ is a unit lower triangular matrix such that its non-zeroes are restricted to the three main diagonals with $S_{j+1,j} = 0$ if and only if $B_{j+1,j} \neq 0$. The Bunch and Kaufman scheme in (1.57) decomposes the matrix $T_k$ in (1.26) by generating a sequence of tridiagonal matrices $T_j$, with $j \leq k$. Given a scalar $\eta \in (0, 1)$, the choice of $1 \times 1$ pivot or $2 \times 2$ pivot for $B_{j,j}$ is based on the following rule:

- if $|\delta_1| \geq \eta|\gamma_2|^2$, then $\delta_1$ is used as a $1 \times 1$ pivot to generate $T_{k-1}$ and $T_k = S_k B_k S_k^T$ (see (1.57)) is equal to

$$T_k = \left(\begin{array}{c|c} 1 & 0\dots 0 \\ \hline S_{2,1} & \\ 0 & I \\ \vdots & \\ 0 & \end{array}\right) \left(\begin{array}{c|c} \delta_1 & 0\dots 0 \\ \hline 0 & \\ 0 & T_{k-1} \\ \vdots & \\ 0 & \end{array}\right) \left(\begin{array}{c|cc} 1 & S_{2,1} & 0\dots 0 \\ \hline 0 & & \\ 0 & I & \\ \vdots & & \\ 0 & & \end{array}\right) \tag{1.58}$$

- if $|\delta_1| < \eta|\gamma_2|^2$, then

$$\left(\begin{array}{cc} \delta_1 & \gamma_2 \\ \gamma_2 & \delta_2 \end{array}\right) \tag{1.59}$$

is used as a $2 \times 2$ pivot to generate $T_{k-2}$ and $T_k = S_k B_k S_k^T$ (see (1.57)) is equal to

$$T_k = \left(\begin{array}{cc|c} 1 & 0 & 0\dots 0 \\ 0 & 1 & 0\dots 0 \\ \hline S_{3,1} & S_{3,2} & \\ 0 & 0 & I \\ \vdots & \vdots & \\ 0 & 0 & \end{array}\right) \left(\begin{array}{cc|c} \delta_1 & \gamma_2 & 0\dots 0 \\ \gamma_2 & \delta_2 & 0\dots 0 \\ \hline 0 & 0 & \\ 0 & 0 & T_{k-2} \\ \vdots & \vdots & \\ 0 & 0 & \end{array}\right) \left(\begin{array}{cc|cc} 1 & 0 & S_{3,1} & 0\dots 0 \\ 0 & 1 & S_{3,2} & 0\dots 0 \\ \hline 0 & 0 & & \\ 0 & 0 & & \\ \vdots & \vdots & & I \\ 0 & 0 & & \end{array}\right) \tag{1.60}$$

In order to determinate the elements of the matrix $S_k$ in (1.57),

- if we compute $T_k$ in (1.58) and we compare it with $T_k$ in (1.26), we get

$$\delta_1 S_{2,1} = \gamma_2, \tag{1.61}$$

that is

$$S_{2,1} = \frac{\gamma_2}{\delta_1}; \tag{1.62}$$

- if we compute $T_k$ in (1.60) and we compare it with $T_k$ in (1.26), we get

$$\begin{cases} S_{3,1}\delta_1 + S_{3,2}\gamma_2 = 0 \\ \\ S_{3,1}\gamma_2 + S_{3,2}\delta_2 = \gamma_3, \end{cases} \tag{1.63}$$

that is

$$\begin{cases} S_{3,1} = -\dfrac{\gamma_3\gamma_2}{\delta_1\delta_2 - \gamma_2^2} \\ \\ S_{3,2} = \dfrac{\gamma_3\delta_1}{\delta_1\delta_2 - \gamma_2^2}. \end{cases} \tag{1.64}$$

From now on, we describe a particular scheme, called SYMMBK, that uses the Bunch and Kaufman factorization in (1.57) to compute $x_k$ from (1.37). We introduce the $n \times k$ matrix $W_k = \begin{bmatrix} w_1 & \vdots & \vdots & w_k \end{bmatrix}$ and the vector $\zeta^{(k)} = (\zeta_1, \ldots, \zeta_k)^T$ such that

$$W_k S_k^T = Q_k \tag{1.65}$$

$$\zeta^{(k)} = S_k^T y_k. \tag{1.66}$$

By (1.57), (1.65) and (1.66), the equations in (1.37) can be rewritten as follows:

$$\begin{aligned} S_k B_k \zeta^{(k)} &= \|b\| e_1 \\ W_k S_k^T &= Q_k \\ x_k &= W_k \zeta^{(k)}. \end{aligned} \tag{1.67}$$

On this guideline, to compute $x_k$, we must analyze both $1 \times 1$ and $2 \times 2$ pivoting steps. If the pivot chosen is $1 \times 1$, $S_k B_k \zeta^{(k)} = \|b\| e_1$ in (1.67) is equivalent to

$$\left(\begin{array}{ccc|c} & S_{k-1} & & \begin{matrix}0\\0\\\vdots\\0\end{matrix} \\ \hline 0\ldots0 & s_1 & s_2 & 1 \end{array}\right) \left(\begin{array}{ccc|c} B_{k-3} & & & \begin{matrix}0\\0\\\vdots\end{matrix} \\ & b_1 & b_3 & 0 \\ & b_2 & b_4 & \\ \hline 0\ldots0 & 0 & 0 & \delta_k \end{array}\right) \left(\begin{array}{c} \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_{k-1} \\ \zeta_k \end{array}\right) = \left(\begin{array}{c} \|b\| \\ 0 \\ \vdots \\ 0 \\ 0 \end{array}\right), \tag{1.68}$$

where $s_1$, $s_2$, $b_1$, $b_2$, $b_3$, $b_4$ and $\delta_k$ are computed from the last iteration. In particular, if the last pivot was $1 \times 1$, then $s_1 = b_2 = b_3 = 0$. From (1.68) we have

$$\left(\begin{array}{ccc|c} & S_{k-1}B_{k-1} & & \begin{matrix}0\\0\\\vdots\\0\end{matrix} \\ \hline 0\ldots0 & (s_1b_1 + s_2b_2) & (s_1b_3 + s_2b_4) & \delta_k \end{array}\right) \left(\begin{array}{c} \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_{k-1} \\ \zeta_k \end{array}\right) = \left(\begin{array}{c} \|b\| \\ 0 \\ \vdots \\ 0 \\ 0 \end{array}\right) \tag{1.69}$$

where

$$B_{k-1} = \begin{pmatrix} B_{k-3} & & \\ & b_1 & b_3 \\ & b_2 & b_4 \end{pmatrix}. \tag{1.70}$$

After some computations, from equations in (1.67), we obtain the following recursive formulae to calculate $x_k$ within the Lanczos algorithm:

$$w_k = q_k - s_1 w_{k-2} - s_2 w_{k-1} \tag{1.71}$$

$$\zeta_k = -\frac{(s_1 b_1 + s_2 b_2)\zeta_{k-2} + (s_1 b_3 + s_2 b_4)\zeta_{k-1}}{\delta_k} \tag{1.72}$$

$$x_k = x_{k-1} + \zeta_k w_k. \tag{1.73}$$

Similarly to the first case, if the pivot chosen is now $2 \times 2$, $S_k B_k \zeta^{(k)} = \|b\| e_1$ in (1.67) is equivalent to

$$\left(\begin{array}{ccc|cc} & & & 0 & 0 \\ & S_{k-2} & & 0 & 0 \\ & & & \vdots & \vdots \\ & & & 0 & 0 \\ \hline 0\dots0 & s_1 & s_2 & 1 & 0 \\ 0\dots0 & 0 & 0 & 0 & 1 \end{array}\right) \left(\begin{array}{ccc|cc} & & & 0 & 0 \\ & B_{k-4} & & 0 & 0 \\ & & & \vdots & \vdots \\ & & b_1 \; b_3 & 0 & 0 \\ & & b_2 \; b_4 & & \\ \hline 0\dots0 & 0 & 0 & \delta_{k-1} & \gamma_k \\ 0\dots0 & 0 & 0 & \gamma_k & \delta_k \end{array}\right) \left(\begin{array}{c} \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_{k-2} \\ \zeta_{k-1} \\ \zeta_k \end{array}\right) = \left(\begin{array}{c} \|b\| \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{array}\right) \quad (1.74)$$

where $s_1$, $s_2$, $b_1$, $b_2$, $b_3$, $b_4$, $\delta_{k-1}$, $\gamma_k$ and $\delta_k$ are computed from the last two iterations. In particular, if the last pivot was $1 \times 1$, then $s_1 = b_2 = b_3 = 0$. By multiplying in (1.74) we have

$$\left(\begin{array}{ccc|cc} & & & 0 & 0 \\ & S_{k-2}B_{k-2} & & 0 & 0 \\ & & & \vdots & \vdots \\ & & & 0 & 0 \\ \hline 0\dots0 & (s_1b_1+s_2b_2) & (s_1b_3+s_2b_4) & \delta_{k-1} & \gamma_k \\ 0\dots0 & 0 & 0 & \gamma_k & \delta_k \end{array}\right) \left(\begin{array}{c} \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_{k-2} \\ \zeta_{k-1} \\ \zeta_k \end{array}\right) = \left(\begin{array}{c} \|b\| \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{array}\right), \quad (1.75)$$

where now

$$B_{k-2} = \left(\begin{array}{ccc} & & \\ & B_{k-4} & \\ & & \end{array}\right. \left.\begin{array}{cc} & \\ b_1 & b_3 \\ b_2 & b_4 \end{array}\right). \quad (1.76)$$

After some computations, from equations in (1.67), we obtain the following recursive formulae to calculate $x_k$ within the Lanczos algorithm:

$$\begin{aligned} w_{k-1} &= q_{k-1} - s_1 w_{k-3} - s_2 w_{k-2} \\ w_k &= q_k \end{aligned} \quad (1.77)$$

$$\begin{aligned} \zeta_{k-1} &= -\delta_k \frac{(s_1b_1+s_2b_2)\zeta_{k-3} + (s_1b_3+s_2b_4)\zeta_{k-2}}{\delta_{k-1}\delta_k - \gamma_k^2} \\ \zeta_k &= \gamma_k \frac{(s_1b_1+s_2b_2)\zeta_{k-3} + (s_1b_3+s_2b_4)\zeta_{k-2}}{\delta_{k-1}\delta_k - \gamma_k^2} \end{aligned} \quad (1.78)$$

$$x_k = x_{k-2} + \zeta_{k-1} w_{k-1} + \zeta_k w_k. \quad (1.79)$$

## 1.2 Preconditioning

Up to now we have seen Krylov methods to solve symmetric linear systems: in particular we focused on CG and Lanczos process. Since CG algorithm may break down in the indefinite case, we investigated the relationship between CG and Lanczos scheme only in the positive definite systems.

In this section we recall a particular technique, known as *Preconditioning* (see e.g. [19], [61], [63], [95], [106]), to improve the efficiency when solving linear system in (1.1) by iterative methods. As already mentioned, the convergence rate of CG method (see Proposition 1.1.5, Proposition 1.1.6 and Corollary 1.1.7) depends on the eigenvalues of the matrix $A$ in (1.1): on this guideline, linear system in (1.1) can be transformed so that the properties of the matrix $A$ may be controlled.

Considering that $A$ in (1.1) is symmetric, if we introduce a nonsingular symmetric matrix $M \in \mathbb{R}^{n\times n}$, the following preconditioned system

$$MAx = Mb, \tag{1.80}$$

gets the same solutions of (1.1). Observe that if we solve (1.80) iteratively, the convergence will not depend on the structural properties of $A$, but on those of $MA$. A good choice of preconditioner enables to solve the preconditioned system in (1.80) much more rapidly than the unpreconditioned system in (1.1). In particular, when dealing with preconditioners, we have two extreme cases:

- if $M = A^{-1}$, the use of preconditioner is as hard as solving the unpreconditioned problem (1.1), and there is no gain to improve efficiency;
- if $M = I$, the use of preconditioner is trivially irrelevant when solving (1.80).

Preconditioners in-between these two extremes can be considered and, an iterative method for (1.80) might converge more quickly than an iterative method for (1.1). The main drawback is how to define that preconditioner $M$. For example, we might claim the eigenvalues of $MA$ to be close to 1 and $\|MA - I\|$ small. A general rule to build good preconditioners is the following given by [106]: "*A preconditioner $M$ is good if $MA$ is not too far from normal and its eigenvalues are clustered*".

### 1.2.1 Preconditioned Conjugate Gradient method

In this section we recall preconditioning the CG method. Starting from CG scheme, we define its improved version known as Preconditioned Conjugate Gradient method (PCG).

In order to apply preconditioning techniques to CG method, hereafter we recall the mathematical ideas which lead to PCG scheme. Let us consider the following preconditioned system in (1.80), that is

$$MAx = Mb.$$

A possible choice of preconditioner might correspond to set $M$ close to the inverse of matrix $A$, such that the eigenvalues of $MA$ could be clustered about the number 1. However, we cannot use this strategy on CG method because we are not guaranteed about the symmetry and positive definiteness of matrix $MA$, even if $M \succ 0$. To overcame this drawback, we consider a symmetric and positive definite matrix $Z \in \mathbb{R}^{n\times n}$ such that $A = Z^2$ (if and only if $A \succ 0$). On this guideline, (1.1) can be rewritten the system as follows:

$$ZZx = b. \tag{1.81}$$

Setting $x = Py$, where $P \in \mathbb{R}^{n\times n}$ is a symmetric and nonsingular matrix and $y \in \mathbb{R}^n$, we get

$$PZZPy = Pb, \tag{1.82}$$

that is

$$PAPy = Pb. \tag{1.83}$$

Note that $PAP$ is a symmetric and positive definite matrix and, in case $P \approx Z^{-1}$, then $PAP$ is close to the identity matrix. Now we apply CG method to the system in (1.83) to calculate the solution $y^*$ and therefore, to obtain the solution given by the original system $x^* = Py^*$. On this guideline we can define a PCG method considering linear system in (1.83), without explicity evaluating $PAP$ matrix and updating iteratively the current iterate $x_k$ in (1.1). Applying CG method to (1.83), we get at each iteration $k \geq 1$:

$$y_{k+1} = y_k + \alpha_k \tilde{p}_k, \tag{1.84}$$

where

$$\alpha_k = \frac{\tilde{g}_k^T \tilde{g}_k}{\tilde{p}_k^T PAP\tilde{p}_k} \tag{1.85}$$

and

$$\tilde{g}_k = PAPy_k - Pb = P(APy_k - b). \tag{1.86}$$

Consider $x_k = Py_k$, $p_k = P\tilde{p}_k$ and setting $M = P^2$ we obtain:

$$x_{k+1} = x_k + \alpha_k p_k \tag{1.87}$$

$$\tilde{g}_k = P(Ax_k - b) = Pg_k \tag{1.88}$$

$$\alpha_k = \frac{g_k^T PPg_k}{p_k^T Ap_k} = \frac{g_k^T Mg_k}{p_k^T Ap_k}. \tag{1.89}$$

Furthermore,

$$\tilde{p}_1 = -\tilde{g}_1, \tag{1.90}$$

$$\tilde{p}_k = -\tilde{g}_k + \beta_k \tilde{p}_{k-1}, \qquad k \geq 2, \tag{1.91}$$

where

$$\beta_k = \frac{\|\tilde{g}_k\|^2}{\|\tilde{g}_{k-1}\|^2} = \frac{g_k^T PPg_k}{g_{k-1}^T PPg_{k-1}} = \frac{g_k^T Mg_k}{g_{k-1}^T Mg_{k-1}}. \tag{1.92}$$

Using (1.90) and (1.91) we get:

$$p_0 = -P\tilde{g}_1 = -PPg_1 = -Mg_1, \tag{1.93}$$

$$p_k = -g_k + \beta_k p_{k-1}, \qquad k \geq 2, \tag{1.94}$$

On the basis of these considerations, we can describe the PCG algorithm for $Ax = b$ in (1.1).

**PCG algorithm**

**Data**: $x_1 \in \mathbb{R}^n$, $M \in \mathbb{R}^{n\times n}$ is a symmetric and positive definite matrix;

Set $k = 1$, $g_1 = Ax_1 - b$ and $p_1 = -Mg_1$;

**While** $g_k \neq 0$

$$\alpha_k = \frac{g_k^T M g_k}{p_k^T A p_k}$$
$$x_{k+1} = x_k + \alpha_k p_k$$
$$g_{k+1} = g_k + \alpha_k A p_k$$
$$\beta_{k+1} = \frac{g_{k+1}^T M g_{k+1}}{g_k^T M g_k}$$
$$p_{k+1} = -Mg_{k+1} + \beta_{k+1} p_k$$
$$k = k + 1$$

**End While**

We note that if $M = I$, then PCG coincides with CG method. Some theoretical properties of CG method can be generalized in case of PCG method. In particular, the orthogonality property (1.17) in Proposition 1.1.4 becomes

$$r_k^T M r_i = 0, \qquad for \qquad i = 1, 2, \ldots, k-1. \tag{1.95}$$

As follows by the PCG scheme, preconditioner $M$ compares only as a product with the gradient: on this guideline it is not necessary to store the preconditioner $M$ but only the matrix-vector product. This fact is very important in large scale setting (that is when $n$ is large). From a computational point of view, the main difference between the PCG and CG methods is that the first one needs an additional matrix-vector product ($p_k = -Mg_k$).

In its favour, we have two improved convergence results (see [36]) .

**Corollary 1.2.1.** *If $\kappa(MA)$ is the condition number of matrix $MA$, then at the $k$-th iteration of PCG algorithm we have*

$$\|x_{k+1} - x^\star\|_A \leq 2\left(\frac{\sqrt{\kappa(MA)} - 1}{\sqrt{\kappa(MA)} + 1}\right)^k \|x_1 - x^\star\|_A. \tag{1.96}$$

**Corollary 1.2.2.** *Suppose that matrix $MA$ has $k$ distinct eigenvalues. Then PCG methods will terminate in at most $k$ iterations.*

Thus, it is now the matrix $MA$ that determines the convergence behaviour of the method. In particular, Corollary 1.2.1 shows that the worst-case convergence

rate depends upon the condition number of $MA$ (instead of the condition number of $A$ as in Proposition 1.1.6), and thus that we should aim to choose $M$ to make this number as small as possible.

In summary, an iteration of the PCG method is more expensive than CG method, but a suited choice of preconditioner may reduce the number of iterations required to achieve a given level of accuracy in the solution. Thus, the overall cost of the preconditioned method may actually be comparably smaller. The difficulty is, of course, in finding a preconditioner with the required spectral properties that is also inexpensive to use. On this guideline, especially when solving difficult large scale and ill-conditioned problems, building good preconditioners is currently still considered a challenging research topic.

## 1.3 Conclusions

In this chapter we focused on the solution of linear systems in large scale setting. To this aim, we introduced Krylov methods, focusing on CG and Lanczos process. Finally, preconditioning idea was briefly reported and an accelerate version of CG method (namely PCG) was presented.

# Chapter 2

# Methods for Large Scale Unconstrained Optimization

In this chapter we deal with methods for large scale unconstrained optimization. In particular, we both detail the meaning of large scale optimization problems, and we introduce some methods commonly used to solve them. Most of the material of this chapter is taken from [23], [63], [95].

## 2.1 Introduction to Large Scale Unconstrained Optimization

To introduce a large scale unconstrained optimization problem, we consider the problem of finding a local minimizer of a real valued objective function $f$ over the space $\mathbb{R}^n$, i.e. to solve the problem

$$\min_{x \in \mathbb{R}^n} f(x), \tag{2.1}$$

where the dimension $n$ is large. The definition of "large scale" is obviously machine dependent. Nowadays, an unconstrained optimization problem with more than $10^3 - 10^4$ variables can be considered a large scale problem. A considerably large number of real world applications can be modeled (or reformulated as) an optimization problem of the form (2.1), strongly motivating the interest for the solution of such problems in several contexts. Large scale problems can be solved efficiently only if the storage and computational costs of the optimization algorithm can be kept at a tolerable level. In the next sections we recall well known methods for large scale unconstrained optimization, namely:

- Nonlinear Conjugate Gradient (NCG) method;
- Quasi-Newton methods;
- Inexact Newton methods.

### 2.1.1 Nonlinear Conjugate Gradient (NCG) method

In Section 1.1.1 we have seen that the CG method can be viewed as a minimization algorithm for the convex quadratic function $f$ defined by (1.12). Anyhow, it is natural to ask how we can extend the CG method to minimize general convex functions, or even a general nonlinear function $f$. To this aim, recalling the CG method for quadratic problems, it is necessary:

- defining a search direction $p_{k+1}$ such that in the coefficient $\beta_{k+1}$ the matrix $A$ does not appear;
- substituting the optimal steplength $\alpha_k$ computed by exact linesearch procedures.

As well known the NCG method is a natural extension of the CG method to general nonconvex functions. In this regard, we remark a noticeable difference between CG and NCG methods. Whenever the CG method is applied, the Hessian matrix does not change during the iterations of the algorithm. On the contrary, when NCG algorithm is applied to a general nonlinear function, the Hessian matrix (possibly indefinite) changes with the iterations. The latter fact implies that the mutual conjugacy of the search directions, generated by the NCG scheme, may be hardly fulfilled.

The NCG methods have been widely studied and are often very efficient when solving large scale problems. The positive steplength $\alpha_k$ is obtained by an appropriate linesearch. Different values of $\beta_k$ give rise to different algorithms (see [67] for a survey), endowed with different convergence properties. Among them, the most common and historically settled schemes are

- $\beta_{k+1} = \dfrac{\|g_{k+1}\|^2}{\|g_k\|^2}$ (Fletcher and Reeves (FR) [50]),
- $\beta_{k+1} = \dfrac{y_k^T g_{k+1}}{\|g_k\|^2}$ (Polak and Ribière (PR) [97]),
- $\beta_{k+1} = \dfrac{y_k^T g_{k+1}}{y_k^T p_k}$ (Hestenes and Stiefel (HS) [69]).

where $y_k = g_{k+1} - g_k$.

However, more recently several other efficient proposals have been introduced in the literature, among them we can report, for instance,

- $\beta_{k+1} = \left( y_k - 2p_k \dfrac{\|y_k\|^2}{y_k^T p_k} \right)^T \dfrac{g_{k+1}}{y_k^T p_k}$ (Hager and Zhang (HZ) [65]),
- $\beta_{k+1} = \dfrac{\|g_{k+1}\|^2}{y_k^T p_k}$ (Dai and Yuan (DY) [37]).

We recall that, if $f$ is a quadratic function and the linesearch procedure is exact,

$$y_k^T g_{k+1} = \|g_{k+1}\|^2 - g_k^T g_{k+1} = \|g_{k+1}\|^2 \text{ (by (1.18) in Proposition 1.1.4),}$$

$$y_k^T p_k = g_{k+1}^T p_k - g_k^T p_k = \|g_k\|^2,$$

being

$$g_{k+1}^T p_k = g_k^T p_k + \alpha_k p_k^T A p_k = g_k^T p_k - \left(\frac{g_k^T p_k}{p_k^T A p_k}\right) p_k^T A p_k = 0,$$
$$g_k^T p_k = -g_k^T g_k + g_k^T(\beta_k p_{k-1}) = -\|g_k\|^2 + \beta_k g_k^T p_{k-1} = -\|g_k\|^2.$$

Thus we can observe that in case $f$ is quadratic the different values of $\beta_k$ are equivalent.

On this guideline, starting from CG scheme in Section 1.1.1, a general NCG method can be reported as follows.

**NCG algorithm**

**Data**: $x_1 \in \mathbb{R}^n$;

Set $k = 1$, compute $g_1$ and $p_1 = -g_1$;

**While** $g_k \neq 0$

compute the steplength $\alpha_k$ by using a linesearch procedure along $p_k$;

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

compute $\beta_{k+1}$;

$$p_{k+1} = -g_{k+1} + \beta_{k+1} p_k \tag{2.2}$$

$$k = k + 1$$

**End While**

To make the NCG scheme complete, in order to compute $\alpha_k$, we need to perform a linesearch procedure that identifies an approximate local minimum of the nonlinear function $f$ along the search direction $p_k$. If $\alpha_k$ does not satisfy certain conditions then the direction $p_{k+1}$ in (2.2) may be ineffective. Consider the inner product at both the sides of (2.2) at the $k$-th iteration, that is $p_k = -g_k + \beta_k p_{k-1}$, we get

$$g_k^T p_k = -\|g_k\|^2 + \beta_k g_k^T p_{k-1}. \tag{2.3}$$

In case of exact linesearch, since $\alpha_{k-1}$ corresponds to compute a stationary point of the function $f$ along the search direction $p_{k-1}$, we have $g_k^T p_{k-1} = 0$. On this guideline, from (2.3), $g_k^T p_k < 0$; this implies that $p_k$ is a descent direction. In case of inexact linesearch, since the second term in the right hand side of (2.3) could

dominate the first term, we may obtain $g_k^T p_k \geq 0$; this implies that $p_k$ is not a descend direction. To overcame this problem, starting from FR algorithm (the first scheme proposed to calculate the parameter $\beta_k$) we require the steplength $\alpha_k$ to satisfy the following conditions (namely the strong Wolfe conditions):

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k g_k^T p_k, \tag{2.4}$$

$$|g(x_k + \alpha_k p_k)^T p_k| \leq c_2 |g_k^T p_k|, \tag{2.5}$$

where $0 < c_1 < c_2 < \frac{1}{2}$.

In order to give descent conditions on search directions, we report the following Proposition (see [95]).

**Proposition 2.1.1.** *Consider the NCG algorithm and assume that $\alpha_k$ satisfies the strong Wolfe conditions* (2.4)-(2.5) *with $0 < c_2 < \frac{1}{2}$. Then, NCG method generates descent directions $p_k$ that satisfy the following inequalities:*

$$-\frac{1}{1-c_2} \leq \frac{g_k^T p_k}{\|g_k\|^2} \leq \frac{2c_2 - 1}{1 - c_2}, \qquad for\ all\ k = 1, 2, \ldots \tag{2.6}$$

By applying Proposition 2.1.1, we can conclude that any linesearch procedure based on the strong Wolfe conditions (2.4)-(2.5) will ensure that all directions $\{p_k\}$ are descent directions for the function $f$.

An important variant of the parameter $\beta_k$ is proposed by Polak and Ribière (PR algorithm). As already said, in case $f$ is strongly convex quadratic function and the linesearch is exact, FR and PR formulae for $\beta_k$ coincide. Instead, when applying PR method to a general nonlinear function with inexact linesearch, the strong Wolfe conditions do not guarantee that the direction $p_k$ is a descent direction. To overcome this drawback, if we define the parameter $\beta$ (see [95]) such that

$$\beta_{k+1}^+ = \max\{\beta_{k+1}, 0\}, \tag{2.7}$$

where $\beta_{k+1}$ is given by PR scheme and we obtain a new algorithm to evaluate $\beta$ called PR+. On this guideline, if the steplength $\alpha_k$ satisfies the strong Wolfe conditions and $\beta_k^+$ is adopted, then the descent property holds. From a numerical point of view, PR method tends to be much more robust and efficient than FR method. An algorithm similar to PR scheme, which is competitive both in terms of theoretical convergence properties and practical performance, is given by Hestenes and Stiefel (HS). Other variants of the parameter $\beta_k$ have been proposed in the literature. In particular, both Hager and Zhang (HZ) (see [65]) and Dai and Yuan (DY) (see [37]) suggested new choices for $\beta_{k+1}$ with appealing theoretical properties and a good computational behaviour.

Some techniques are investigated in order to improve the efficiency of the NCG methods. Usually, a quadratic or cubic interpolation along $p_k$ is implemented into the linesearch procedure. Another modification used in NCG scheme is to restart the iteration every $n$ steps by setting $\beta_k = 0$, that is $p_{k+n} = -g_{k+n}$. This technique is useful to periodically refresh the algorithm, crossing out old information that may not be beneficial. On this guideline, formula (2.7) can be seen as a restarting strategy because $p_{k+1}$ will revert to the steepest descent direction whenever $\beta_k$ (according to PR scheme) is negative.

Unlike CG algorithm, "NCG methods possess surprising, sometimes bizarre, convergence properties" (J. Nocedal in [95]). Here below we recall some results for the FR and PR methods. In order to prove a global convergence for the FR method, we have to introduce the Zoutendijk's conditions (see [63]), reported in the following Proposition.

**Proposition 2.1.2.** *Let $f$ be a continuously differentiable function on $\mathbb{R}^k$ and assume it is bounded below. Let $\{x_k\}$ be an infinite sequence with $g_k \neq 0$. Suppose there exists $\mu > 0$ such that*

$$f(x_k) - f(x_{k+1}) \geq \mu \|g_k\|^2 \cos^2 \theta_k, \tag{2.8}$$

*where* $\cos \theta_k = \dfrac{g_k^T p_k}{\|g_k\| \|p_k\|}$. *Then, if*

$$\sum_{k=1}^{\infty} \cos^2 \theta_k = \infty \tag{2.9}$$

*it results*

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

Considering Proposition 2.1.1 and Proposition 2.1.2, global convergence results for FR method can be proved if an exact linesearch is used. Al-Baali in [2] proved the global convergence of FR method with inexact linesearch. However, as already said, from a numerical point of view, for general nonconvex functions, the FR algorithm performs worse than the PR algorithm. Note that if $x_{k+1} \approx x_k$, then we have $g_{k+1} \approx g_k$, so that using PR method we obtain $\beta_{k+1} \approx 0$ and $p_{k+1} = -g_{k+1}$ (conversely, with FR method it is $\beta_{k+1} \approx 1$). Thus, PR algorithm has a kind of "automatic restart" along the steepest descent direction $-g_{k+1}$. Restart technique is the simplest criterion to guarantee global convergence because we use the steepest descent direction: nevertheless, numerical experiences show that globalization techniques based on the modification of $\beta_{k+1}$ or the use of appropriate linesearch procedures are preferable with respect to the restart technique. The global convergence of the PR method with exact linesearch has been proved in [97] under strong convexity assumption on $f$. Grippo & Lucidi in [62] proved the global convergence of PR method with particular inexact linesearch conditions. Further references on NCG convergence can be found in [14], [15], [43], [71], [113].

### 2.1.1.1 Preconditioned Nonlinear Conjugate Gradient method

"A major drawback of NCG methods is that the search directions tend to be poorly scaled, and the linesearch typically requires several function evaluations to obtain an acceptable steplength $\alpha_k$" (J. Nocedal in [94]).

On this guideline, a keynote issue for enhancing NCG efficiency is to include a preconditioning strategy, especially when solving difficult ill-conditioned problems. In this section first we report the scheme of a general Preconditioned Nonlinear Conjugate Gradient (PNCG) algorithm (see e.g. [101]), where $M_k \succ 0$ denotes the preconditioner at the $k$-th iteration.

**PNCG algorithm**

**Data**: $x_1 \in \mathbb{R}^n$, $M_1 \in \mathbb{R}^{n\times n}$ is a symmetric and positive definite matrix;

Set $k = 1$, compute $g_1$ and $p_1 = -M_1 g_1$;

**While** $g_k \neq 0$

compute the steplength $\alpha_k$ by using a linesearch procedure along $p_k$;

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

compute $\beta_{k+1}$;

$$p_{k+1} = -M_{k+1} g_{k+1} + \beta_{k+1} p_k \tag{2.10}$$

$$k = k + 1$$

**End While**

By setting $M_k = I$ for any $k$, the popular (unpreconditioned) NCG method is trivially obtained. The parameter $\beta_{k+1}$ depends on the preconditioner, too, and can be chosen in a variety of ways. For PNCG algorithm, among the most recurrent choices of $\beta_{k+1}$ from the literature there are the following ones (they all require $M_k \succ 0$ for any $k$):

$$\beta_{k+1}^{\mathrm{FR}} = \frac{g_{k+1}^T M_{k+1} g_{k+1}}{g_k^T M_k g_k}, \tag{2.11}$$

$$\beta_{k+1}^{\mathrm{PR}} = \frac{y_k^T M_{k+1} g_{k+1}}{g_k^T M_k g_k}, \tag{2.12}$$

$$\beta_{k+1}^{\mathrm{HS}} = \frac{y_k^T M_{k+1} g_{k+1}}{y_k^T p_k}, \tag{2.13}$$

$$\beta_{k+1}^{\mathrm{HZ}} = \frac{y_k^T M_{k+1} g_{k+1}}{p_k^T y_k} - \Theta_k \frac{y_k^T M_{k+1} y_k}{p_k^T y_k} \frac{p_k^T g_{k+1}}{p_k^T y_k}, \tag{2.14}$$

where $\Theta_k$ is a suitable parameter.

We recall that to guarantee global convergence, an accurate linesearch technique is required to determine the steplength $\alpha_k$ in a PNCG algorithm. The latter fact justifies the use of a linesearch procedure, ensuring the strong Wolfe conditions (2.4)-(2.5). This also guarantees that the condition

$$s_k^T y_k > 0, \qquad \text{for any } k \tag{2.15}$$

holds, where $s_k = \alpha_k p_k = x_{k+1} - x_k$. As we will see, (2.15) is a fundamental relation to our future purposes.

As already said, preconditioning is applied for increasing the efficiency of the NCG method.

### 2.1.2 Quasi-Newton methods

Quasi-Newton methods represent an efficient tool for large scale optimization. Without using the second-order derivatives and maintaining superlinear convergence (under reasonable assumptions), they provide a good approximation of the Newton method.

Consider the following Taylor series of the convex objective function $f(x)$ at the current iterate $x_k$:

$$f(x_k + p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T \nabla^2 f(x_k) p + \gamma(x_k, p) \tag{2.16}$$

where $\lim_{p \to 0} \frac{\gamma(x_k, p)}{\|p\|^2} = 0$. If $\|p\|$ is small, $f(x_k + p)$ in (2.16) can be approximate with the quadratic function

$$q_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T \nabla^2 f(x_k) p. \tag{2.17}$$

Since $\nabla q_k(p) = \nabla f(x_k) + \nabla^2 f(x_k) p$ and if matrix $\nabla^2 f(x_k)$ is positive definite, a minimum point of $q_k(p)$ will given by

$$p_k = -[\nabla^2 f(x_k)]^{-1} \nabla f(x_k). \tag{2.18}$$

On this guideline, quasi-Newton methods try to generate the sequence $\{x_k\}$ by iteratively solving the "modified Newton's equation"

$$p_k = -[B_k]^{-1} \nabla f(x_k) \tag{2.19}$$

and computing

$$x_{k+1} = x_k + s_k = x_k - \alpha_k B_k^{-1} \nabla f(x_k), \tag{2.20}$$

where $B_k \in \mathbb{R}^{n \times n}$ is positive definite and in some sense approximates the Hessian matrix $\nabla^2 f(x_k)$ of the quadratic function $q_k(p)$.

Now, the question is this: is it possible to update $B_{k+1}$ using the knowledge gained during the latest step?

Suppose that we have generated a new iterate $x_{k+1}$ and we want to construct a new quadratic function

$$q_{k+1}(p) = f(x_{k+1}) + \nabla f(x_{k+1})^T p + \frac{1}{2} p^T B_{k+1} p. \tag{2.21}$$

An important idea underlying many quasi-Newton methods is that the gradient of $q_{k+1}$ should match the gradient of the objective function $f$ at the latest two iterates $x_k$ and $x_{k+1}$. On this guideline,

$$\nabla q_{k+1}(-\alpha_k p_k) = \nabla f(x_{k+1}) - \alpha_k B_{k+1} p_k = \nabla f(x_k), \tag{2.22}$$

that is

$$B_{k+1} \alpha_k p_k = \nabla f(x_{k+1}) - \nabla f(x_k). \tag{2.23}$$

Recalling that

$$s_k = x_{k+1} - x_k = \alpha_k p_k \tag{2.24}$$

and

$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k), \tag{2.25}$$

by (2.23) we obtain the "secant equation"

$$B_{k+1}s_k = y_k. \tag{2.26}$$

Now we recall an important property to ensure that $B_{k+1}$ is positive definite.

**Proposition 2.1.3.** *Let $B_k \succ 0$. Then $B_{k+1} \succ 0$ if and only if $s_k^T y_k > 0$.*

On this guideline, if we consider the iterate $x_{k+1}$ in (2.20), it can be possible to give some conditions on $\alpha_k$ such that the "curvature condition" in (2.15) is satisfied. In fact, (2.15) can be written as follows:

$$p_k^T \nabla f(x_{k+1}) > p_k^T \nabla f(x_k). \tag{2.27}$$

We note that condition in (2.27) can be satisfied using any linesearch procedure based on the (strong) Wolfe conditions ((2.4)-(2.5)). When the curvature condition in (2.27) is satisfied, the secant equation in (2.26) always has a solution $B_{k+1}$. To determine $B_{k+1}$ uniquely, we solve the following problem

$$\begin{array}{ll} \min\limits_{B} & \|B - B_k\|_F \\ s.t. & B = B^T \\ & Bs_k = y_k. \end{array}$$

Up to now, in quasi-Newton methods, an approximation of $\nabla^2 f(x_k)$ has been considered. However, if we focus on the inverse of $B_k$, denoted by

$$H_k = B_k^{-1}, \tag{2.28}$$

we easily get an analogous result obtained by using $B_k$. On this guideline, a sequence $H_k$ of approximations of $[\nabla^2 f(x_k)]^{-1}$ is generated, with $H_k \succ 0$. Starting from the search direction

$$p_k = -H_k \nabla f(x_k) \tag{2.29}$$

we compute

$$x_{k+1} = x_k + s_k = x_k - \alpha_k H_k \nabla f(x_k) \tag{2.30}$$

and the secant equation can be rewritten in this way:

$$H_{k+1}y_k = s_k. \tag{2.31}$$

According to Proposition 2.1.3, similar conditions on $H_{k+1}$ are achieved. Finally, the condition of closeness to $H_k$ is specified as follows:

$$\begin{array}{ll} \min\limits_{H} & \|H - H_k\|_F \\ s.t. & H = H^T \\ & Hy_k = s_k. \end{array}$$

In the sequel we recall several updating formulae:

- Symmetric Rank-1 (SR1);
- Symmetric Rank-2 (SR2).

The SR1 updating formulae are chiefly used to solve nonlinear systems and the SR2 updating formulae are used for unconstrained optimization problems.

The SR1 update has the general form

$$B_{k+1} = B_k + \rho_k u_k v_k^T, \tag{2.32}$$

where $\rho_k \in \mathbb{R}$ and $u_k, v_k \in \mathbb{R}^n$. By substituting (2.32) in (2.26), we obtain

$$B_k s_k + \rho_k u_k v_k^T s_k = y_k. \tag{2.33}$$

Assuming

$$u_k = y_k - B_k s_k, \tag{2.34}$$

$$\rho_k = \frac{1}{v_k^T s_k}, \tag{2.35}$$

and considering $v_k$ an arbitrary vector such that $v_k^T s_k \neq 0$, equation (2.33) is satisfied. Hence, we get

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k) v_k^T}{v_k^T s_k}. \tag{2.36}$$

In particular, setting $v_k = s_k$, we obtain the well known "Broyden updating formula", that is

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k) s_k^T}{s_k^T s_k}. \tag{2.37}$$

The corresponding update formula for the inverse Hessian approximation $H_k$ is easily obtained by applying the well known "Sherman-Morrison-Woodbury formula" (see [63]) into (2.37), that is

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k) s_k^T H_k}{s_k^T H_k s_k}, \tag{2.38}$$

provided that $s_k^T H_k s_k \neq 0$.

As already said SR1 updating formulae are not suitable for unconstrained optimization problem: in fact, (2.37) neither ensures that $B_{k+1}$ is a symmetric matrix, nor guarantees that $-B_k^{-1}\nabla f(x_k)$ is a descent direction.

However, setting $v_k = y_k - B_k s_k$ and recalling that $v_k^T s_k \neq 0$, we obtain

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}. \tag{2.39}$$

The corresponding update formula for the inverse Hessian approximation $H_k$ is

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T s_k}. \tag{2.40}$$

It can be proved that, in quadratic case, if $(s_k - H_k y_k)^T y_k \neq 0$ holds and if $s_0, s_1, \ldots$ are linearly indipendent, SR1 method defined in (2.30) determines the minimum point

of the quadratic function $f$ in at most $n$ iterations and the matrix $H_n$ coincides with the inverse of the Hessian matrix of $f$. However, in non-quadratic case, (2.40) does not ensure that the search direction is a descent direction and if $(s_k - H_k y_k)^T s_k$ is too close to zero there may be instability issues. On this guideline, for unconstrained optimization problem, it is preferable to use SR2 updating formulae.

The SR2 update has the general form

$$H_{k+1} = H_k + a_k u_k u_k^T + b_k v_k v_k^T, \tag{2.41}$$

where $u_k, v_k \in \mathbb{R}^n$ and $a_k, b_k \in \mathbb{R}$. Consider the secant equation (2.31) we obtain

$$H_k y_k + a_k u_k u_k^T y_k + b_k v_k v_k^T y_k = s_k. \tag{2.42}$$

Assuming

$$a_k = \frac{1}{s_k^T y_k}, \tag{2.43}$$

$$u_k = s_k, \tag{2.44}$$

$$b_k = -\frac{1}{y_k^T H_k y_k}, \tag{2.45}$$

$$v_k = H_k y_k, \tag{2.46}$$

equation (2.42) is satisfied. Hence, we get the well known "Davidon-Fletcher-Powell (DFP) updating formula", that is

$$H_{k+1} = H_k + \frac{s_k s_k^T}{s_k^T y_k} - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k}. \tag{2.47}$$

The corresponding updating formula for the Hessian approximation $B_k$ is

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k) y_k^T + y_k (y_k - B_k s_k)^T}{s_k^T y_k} - \frac{s_k^T (y_k - B_k s_k) y_k y_k^T}{(s_k^T y_k)^2}. \tag{2.48}$$

A particular class of updating formulae, which includes (2.47)-(2.48), is called "Broyden family" that is

$$H_{k+1} = H_k + \frac{s_k s_k^T}{s_k^T y_k} - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \phi v_k v_k^T, \tag{2.49}$$

where $\phi \geq 0$ and

$$v_k = (y_k^T H_k y_k)^{\frac{1}{2}} \left( \frac{s_k}{s_k^T y_k} - \frac{H_k y_k}{y_k^T H_k y_k} \right). \tag{2.50}$$

We note that setting $\phi = 0$ into (2.49), we get DFP formula in (2.47). Fixing $\phi = 1$ we obtain the well known "Broyden-Fletcher-Goldfarb-Shanno (BFGS) updating formula", that is

$$H_{k+1} = H_k + \left( 1 + \frac{y_k^T H_k y_k}{s_k^T y_k} \right) \frac{s_k s_k^T}{s_k^T y_k} - \frac{s_k y_k^T H_k + H_k y_k s_k^T}{s_k^T y_k}. \tag{2.51}$$

The corresponding updating formula for the Hessian approximation $B_k$ is

$$B_{k+1} = B_k + \frac{y_k y_k^T}{s_k^T y_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}, \tag{2.52}$$

moreover, we get

$$H_{k+1}^{(BFGS)} = H_{k+1}^{(DFP)} + v_k v_k^T, \tag{2.53}$$

where $v_k$ is given by (2.50), and the Broyden family's matrices can be defined as follows:

$$H_{k+1}^{(Broyden)} = (1-\phi)H_{k+1}^{(DFP)} + \phi H_{k+1}^{(BFGS)}. \tag{2.54}$$

On the basis of Proposition 2.1.3, all Broyden family's updating formulae ensure that $H_{k+1}$ is positive definite as long as $s_k^T y_k > 0$ and $\phi \geq 0$.

Numerical experience shows that BFGS updating formula seems to be preferable than other ones. The BFGS scheme is now reported.

**BFGS algorithm**

**Data**: $x_1 \in \mathbb{R}^n$, $H_1 \succ 0$;

Set $k = 1$ and compute $\nabla f(x_1)$;

**While** $\nabla f(x_k) \neq 0$

$$p_k = -H_k \nabla f(x_k);$$

compute the steplength $\alpha_k$ by using a linesearch procedure, which ensures the *strong Wolfe conditions* (2.4)-(2.5)

$$x_{k+1} = x_k + \alpha_k p_k; \tag{2.55}$$

$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k); \tag{2.56}$$

$$s_k = x_{k+1} - x_k;$$

compute $H_{k+1}$ by means of (2.51);

$$k = k + 1$$

**End While**

Although BFGS method is considered very robust in practice, global convergence results for general nonlinear objective function do not exist. However, under the following assumptions, global convergence result can be established.

**Assumption 2.1.4.**

*i) Let be given an arbitrary starting point $x_1 \in \mathbb{R}^n$ and an initial Hessian approximation $B_1$ that is symmetric and positive definite.*

*ii) Let $f$ be a twice continuously differentiable objective function.*

*iii) The level set $\mathcal{L} = \{x \in \mathbb{R}^n : f(x) \leq f(x_1)\}$ be convex and assume there exist positive constants $\gamma$ and $\Gamma$ such that*

$$\gamma\|z\|^2 \leq z^T \nabla^2 f(x) z \leq \Gamma\|z\|^2 \tag{2.57}$$

*for all $z \in \mathbb{R}^n$ and $x \in \mathcal{L}$.*

On detail, the result is reported.

**Proposition 2.1.5.** *Let Assumption 2.1.4 hold. Then the sequence $x_k$ generated by BFGS scheme converges to the minimizer $x^*$ of $f$.*

The main drawback of this result is that it can be applied to the entire Broyden class, except for the DFP method.

Finally we can establish that, under some hypotheses, the rate of convergence of the iterates generated by BFGS method is superlinear. Consider the following assumption.

**Assumption 2.1.6.** *The Hessian matrix $\nabla^2 f(x)$ is Lipschitz continuous at $x^*$. In particular, there exists $L > 0$ such that*

$$\|\nabla^2 f(x) - \nabla^2 f(x^*)\| \leq L\|x - x^*\| \qquad \text{for all } x \text{ near } x^*. \tag{2.58}$$

The result reported shows the superlinear convergence of the BFGS method.

**Proposition 2.1.7.** *Let $f$ be a twice continuously differentiable objective function. Suppose that the sequence $x_k$ generated by BFGS scheme converges to the minimizer $x^*$ of $f$. Let Assumption 2.1.6 hold. If*

$$\sum_{k=1}^{\infty} \|x_k - x^*\| < \infty \tag{2.59}$$

*is satisfied, then $x_k$ converges to $x^*$ at a superlinear rate, that is*

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x_k\|}{\|x_k - x^*\|} = 1. \tag{2.60}$$

#### 2.1.2.1 Damped Quasi-Newton methods

On nonlinear "difficult" problems where the Hessian matrix is possibly highly ill-conditioned, also quasi-Newton methods may be inefficient. This was already known since 1986 when Powell in [100] analyzed the performance of the BFGS and DFP algorithms in the case of a quadratic function of two variables (see also [10]).

To overcome the latter drawback, a technique originated by Powell in [99], and was reproposed by Al-Baali in [3] for quasi-Newton methods: the so called "*damped technique*". In that paper, Powell deals with SQP Lagrangian BFGS methods for nonlinearly constrained problems. He proposes a modification of BFGS that, to

some extent, offsets the lack of positive definiteness in the Hessian of the Lagrangian at the solution. Indeed, due to the presence of negative curvature directions of the Lagrangian function, using BFGS for approximating the Hessian matrix with a positive definite matrix, may be seriously inappropriate (see also [95] Section 18.3). Damped techniques have been recently extended by Al-Baali also to the restricted Broyden class of quasi-Newton methods for unconstrained optimization problems in [3]. The author extends the global and superlinear convergence properties that the Broyden family of methods fulfills for convex functions, to a novel class of methods, namely the D-Broyden class (see also [7] [8] [9] [10] [11]). The rationale behind damped techniques is the following.

As already mentioned, in dealing with the BFGS update, a crucial issue in order to guarantee positive definiteness of the updated Hessian approximation is the curvature condition in (2.15), that is $s_k^T y_k > 0$. If $f$ is strongly convex on an open set containing the level set $\mathcal{L}$, then (2.15) holds for any two points $x_k$ and $x_{k+1}$ belonging to $\mathcal{L}$ (see, e.g. [25]). In case of nonconvex functions, as already mentioned, the satisfaction of condition (2.15) must be ensured by means of the linesearch procedure used for determining the stepsize $\alpha_k$. Indeed, the satisfaction of (2.15) can be always obtained by a linesearch procedure if the objective function is bounded below on $\mathcal{L}$. To this aim the Wolfe conditions (in practice, strong Wolfe conditions in (2.4)-(2.5)) are usually adopted, which ensure condition (2.15). However, if the linesearch is not fairly accurate, the value of $s_k^T y_k$ may not be sufficiently positive. In addition, if only the backtracking linesearch framework is employed, the curvature condition (2.15) may not hold.

A first possible strategy to cope with this issue is to reinitialize the model Hessian to the identity matrix or skip the update whenever $s_k^T y_k \leq 0$ (see e.g. Section 4.2.2 of [74]). However, this strategy is usually not recommended, due to the loss of information on the curvature of the function. A more successful strategy is the *damped technique* proposed by Powell in [99], in the context of SQP Lagrangian BFGS method for constrained optimization, for which (2.15) may not hold even when the Wolfe conditions are employed. To overcome this difficulty, the author proposes to modify the difference of the gradients vector $y_k$ in (2.56) before performing the update. Let $B_k$ be the available positive definite Hessian approximation at $k$-th iteration of the method, the following modified (damped) vector is used:

$$\widehat{y}_k = \varphi_k y_k + (1 - \varphi_k) B_k s_k, \tag{2.61}$$

where $\varphi_k$ is chosen in $(0, 1]$ such that $s_k^T \widehat{y}_k$ is "sufficiently positive". Namely, given $\sigma \in (0, 1]$, the value of $\varphi_k$ is set as follows:

$$\varphi_k = \begin{cases} \dfrac{\sigma s_k^T B_k s_k}{s_k^T B_k s_k - s_k^T y_k}, & \text{if} \quad s_k^T y_k < (1 - \sigma) s_k^T B_k s_k, \\ \\ 1, & \text{otherwise}, \end{cases} \tag{2.62}$$

(see also Section 18.3 in [95]). This choice ensures that $s_k^T \widehat{y}_k = (1 - \sigma) s_k^T B_k s_k$ is sufficiently positive, since $B_k$ is imposed to be positive definite at each iteration. In [99] the value of $\sigma = 0.8$ is suggested as a "suitable size" to be used in (2.62) (see also [95]); the value of $\sigma = 0.9$ is sometimes used, too (see e.g. [3] [10]).

To the best of our knowledge, Powell's damped technique was never applied to unconstrained optimization problems until Al-Baali used it for improving the performance of the standard BFGS and DFP methods (see [3] [7] [8] [9] [10] [11]). In particular, the author extends the damped technique to the Broyden family of quasi-Newton methods for unconstrained optimization. The choice given in (2.62) is modified so that the damped vector $\widehat{y}_k$ replaces $y_k$ in the quasi-Newton updating formulae whenever the ratio $\frac{s_k^T y_k}{s_k^T B_k s_k}$ is sufficiently close to zero or negative (like in the Powell's strategy). This choice enforces both global and superlinear convergence properties of the novel class of methods proposed in [3], namely the *D-Broyden* class. We note that (2.62) does not modify $y_k$ when $\frac{s_k^T y_k}{s_k^T B_k s_k}$ is larger than 1. Therefore, Al-Baali also suggests using the modified damped vector (2.61) when the ratio $\frac{s_k^T y_k}{s_k^T B_k s_k}$ is large enough by extending the above choice as follows:

$$\varphi_k = \begin{cases} \dfrac{\sigma s_k^T B_k s_k}{s_k^T B_k s_k - s_k^T y_k}, & \text{if} \quad s_k^T y_k < (1-\sigma) s_k^T B_k s_k, \\ \dfrac{\hat{\sigma} s_k^T B_k s_k}{s_k^T B_k s_k - s_k^T y_k}, & \text{if } s_k^T y_k > (1+\hat{\sigma}) s_k^T B_k s_k, \\ 1, & \text{otherwise}, \end{cases} \tag{2.63}$$

where $\hat{\sigma} \geq 2$. In this work, we consider the value of $\hat{\sigma} = \infty$ which reduces the above choice to (2.62).

#### 2.1.2.2 Quasi-Newton methods for Large Scale Optimization

As we have seen, BFGS method can be described by $x_{k+1} = x_k - \alpha_k H_k \nabla f(x_k)$ in (2.30) where the steplength $\alpha_k$ is provided by a linesearch procedure that satisfies the strong Wolfe conditions ((2.4)-(2.5)) and matrix $H_k$ fulfills the secant equation $H_{k+1} y_k = s_k$ in (2.31).

The unique solution $H_{k+1}$ for (2.31) is given by

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T, \tag{2.64}$$

where

$$\rho_k = \frac{1}{y_k^T s_k}, \tag{2.65}$$

$$V_k = I - \rho_k y_k s_k^T. \tag{2.66}$$

Since matrix $H_k$ in general is dense, storing fully matrix can be considered prohibitive. To overcame this drawback,

- Memoryless BFGS;
- Limited Memory BFGS (L-BFGS);

have been proposed.

#### 2.1.2.2.1 Memoryless Quasi-Newton methods

The basic idea of these methods is to restart the matrix $H_k$ at each iteration $k$ by setting $H_k = I$ in (2.64), that is

$$H_{k+1} = V_k^T V_k + \rho_k s_k s_k^T. \tag{2.67}$$

At each iteration $k$, Memoryless BFGS algorithms only need to store the pair of vectors $(s_k, y_k)$. In order to investigate Memoryless BFGS methods, we remind that there exists a close connection between Memoryless BFGS methods and NCG methods. In particular we can see that any quasi-Newton method based on (2.67) is equivalent to Hestenes and Stiefel (HS) NCG method. On this guideline, using (2.65)-(2.66), (2.67) can be rewritten as follows:

$$H_{k+1} = \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right)^T \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k}. \tag{2.68}$$

If the linesearch procedure is exact, we get (as already seen above) $g_{k+1}^T p_k = 0$: the search direction for quasi-Newton method at the iteration $k+1$ is given by

$$p_{k+1} = -H_{k+1} \nabla f(x_{k+1}). \tag{2.69}$$

Using (2.68) in (2.69) we obtain

$$p_{k+1} = -\left(I - \frac{y_k s_k^T}{y_k^T s_k}\right)^T \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) \nabla f(x_{k+1}) - \frac{s_k s_k^T}{y_k^T s_k} \nabla f(x_{k+1}). \tag{2.70}$$

Recalling that $g_{k+1}^T p_k = 0$, we have

$$p_{k+1} = -\nabla f(x_{k+1}) + \frac{s_k y_k^T}{y_k^T s_k} \nabla f(x_{k+1}). \tag{2.71}$$

From (2.24)-(2.25), (2.71) can be transformed as follows:

$$p_{k+1} = -\nabla f(x_{k+1}) + \frac{\nabla f(x_{k+1})^T (\nabla f(x_{k+1}) - \nabla f(x_k))}{p_k^T (\nabla f(x_{k+1}) - \nabla f(x_k))} p_k = -\nabla f(x_{k+1}) + \beta_{k+1} p_k \tag{2.72}$$

where $\beta_{k+1}$ is given by Hestenes and Stiefel (HS) algorithm. We note that (2.72) is the search direction computed by HS method and is equivalent to the search direction (2.69) computed by Memoryless BFGS method.

#### 2.1.2.2.2 Limited Memory BFGS (L-BFGS)

Limited Memory quasi-Newton methods are useful for solving large problems whose Hessian matrices cannot be computed at a reasonable cost or are not sparse. As already mentioned (see (2.29)), when using quasi-Newton methods we generate a search direction of the form $p_k = -H_k \nabla f(x_k)$, where $H_k$ is an approximation of the inverse of the Hessian matrix $\nabla^2 f(x_k)$. Instead of storing full dense $n \times n$

approximations, Limited Memory quasi-Newton methods only save a few vectors of length $n$, which allow to represent the approximations $\{H_k\}$ implicitly.

Among the quasi-Newton schemes, the L-BFGS method is usually considered one of the most efficient [78, 93]. It is well suited for large scale problems because the amount of storage it requires is limited and controlled by the user. This method is based on the construction of the approximation of the inverse of the Hessian matrix, by exploiting curvature information gained only from the most recent iterations. Specifically, consider (2.64), (2.65) and (2.66), the inverse of the Hessian matrix is updated by L-BFGS at the $k$-th iteration as

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T. \tag{2.73}$$

Observe that rearranging the expression of $H_k$ we can also iteratively obtain relation

$$\begin{aligned} H_k &= (V_{k-1}^T \cdots V_{k-m}^T) H_k^0 (V_{k-m} \cdots V_{k-1}) \\ &\quad + \rho_{k-m}(V_{k-1}^T \cdots V_{k-m+1}^T) s_{k-m} s_{k-m}^T (V_{k-m+1} \cdots V_{k-1}) \\ &\quad + \rho_{k-m+1}(V_{k-1}^T \cdots V_{k-m+2}^T) s_{k-m+1} s_{k-m+1}^T (V_{k-m+2} \cdots V_{k-1}) \\ &\quad + \cdots \\ &\quad + \rho_{k-1} s_{k-1} s_{k-1}^T, \end{aligned}$$

where $m$ is the *memory* of the method and $H_k^0$ is an initial approximation of the inverse of the Hessian matrix (see [78], [93], [95]).

The well known reasons for the success of the L-BFGS method can be summarized in the following two points: firstly, even when $m$ is small, $H_{k+1}$ proves to be an effective approximation of the inverse of the Hessian matrix. Secondly $H_{k+1}$ is the unique (positive definite) matrix which solves the subproblem

$$\begin{aligned} \min_H \ & \|H - H_k\|_F \\ s.t. \ & H = H^T \\ & H y_k = s_k. \end{aligned}$$

in Section 2.1.2.

The strategy of keeping the $m$ most recent pairs $\{(s_i, y_i)\}$ works well in practice. Moreover, during its first $m-1$ iterations, L-BFGS algorithm is equivalent to the BFGS algorithm in Section 2.1.2 if L-BFGS chooses $H_k^0 = H_1$ at each iteration. However, as well known L-BFGS method presents some drawbacks, including the slow convergence on ill-conditioned problems, namely when the eigenvalues of the Hessian matrix are very spread. On certain applications, the NCG methods are competitive with L-BFGS method.

### 2.1.3 Inexact Newton methods

Another effective approach to large scale unconstrained optimization is represented by inexact Newton methods. As well known, Newton's method requires to compute $p_k$ in (2.18), which is equivalent to solving Newton's equation

$$\nabla^2 f(x_k) p = -\nabla f(x_k). \tag{2.74}$$

The basic idea of inexact Newton method is the following: as long as the iterate $x_k$ is still far from the stationary point $x^*$, an accurate solution of (2.74) may be useless. In addition, in order to preserve the global convergence of Newton's method, a suitable choice of $\alpha_k$ must be computed. On this guideline inexact Newton method tries to obtain approximation of $p_k$ that is inexpensive to calculate and is a good search direction. In the following section we show that inexact Newton strategies can be incorporated in practical linesearch implementation of Newton's method (also in trust region implementation, too), maintaining good local and global convergence properties.

#### 2.1.3.1 Newton-Krylov method

Newton-Krylov methods are widely used for solving large scale problems. They are so called because a Krylov subspace method is usually employed, for approximately solving the Newton equation at each iteration. A general description of the Newton-Krylov methods can be found in the survey paper [88]. It is well known that, given an initial guess $x_1$ of a local minimizer of problem (2.1), a Newton-Krylov method is based on two nested loops: the *outer iterations* which represent the actual steps of the method, where the current estimate of the solution is updated; the *inner iterations* which carry out an iterative algorithm for computing, at each outer iteration $k$, a search direction $p_k$ by approximately solving the Newton equation (2.74). The iterative algorithm used for solving (2.74) is actually "truncated", i.e. terminated before the exact solution is obtained. This strategy is based on the fact that, since the benefits of using a Newton direction are local, i.e. in the neighborhood of a stationary point, an accurate solution of (2.74) may be unjustified when $x_k$ is far from a local optimizer. As matter of fact, in this case, a much simpler search direction can often perform comparably well. Instead, more accuracy is required when the iterates approach a local minimizer. A good trade-off between the accuracy in solving the Newton equation (2.74) and the computational effort employed per outer iteration is a key point, for the overall efficiency of a Newton-Krylov method. Indeed, there might be significant advantages to terminating the inner iterations early, when we are still far from a solution and the problem has significant nonlinearities. On the contrary, when close to a solution, there may be disadvantages to early terminating, inasmuch as the corresponding search direction $p_k$ might be poor.

Since the early papers [39], [40] where Newton-Krylov methods were introduced, the importance of an efficient truncation criterion for the inner iterations was pointed out. The stopping rule proposed therein is based on controlling the magnitude of the residual. Under suitable assumptions, this allows to guarantee local convergence and a good convergence rate. Another truncation rule has been proposed in [90]. It is based on a comparison between the reduction of the quadratic model of the objective function, at the current iteration, and the average reduction per iteration of the model.

In this section we recall only a linesearch approach: observe that the trust region approach will not be investigated in this work. In large scale context, iterative methods like the CG algorithm can be used to compute the inner iterations: in case of the Hessian matrix in (2.74) is indefinite, Lanczos algorithm may successfully be used as alternative to the CG method.

#### 2.1.3.1.1 Common truncation criteria

In order to briefly recall the truncation criteria commonly used in the literature, we denote by $p_k$ an approximate solution of (2.74), and by $r_k = \nabla^2 f(x_k)p_k - \nabla f(x_k)$ the corresponding residual.

A natural stopping criterion for the inner iterations is the *residual-based criterion*, proposed in the seminal papers [39] and [40]. Indeed, the authors propose to terminate the inner iterations whenever the residual $r_k$ is sufficiently small, namely

$$\frac{\|r_k\|}{\|\nabla f(x_k)\|} \leq \eta_k, \tag{2.75}$$

for a specified value of $\eta_k$. It is well known that criterion (2.75) is scale invariant and that the choice of the *forcing sequence* $\{\eta_k\}$ is crucial for controlling the convergence rate of the algorithm. A widely used choice proposed in [40] is $\eta_k = \min\{1/k, \|\nabla f(x_k)\|^r\}$, with $0 < r \leq 1$. Other forcing sequences have been proposed later in [45] and [46]. The possibility to easily control the rate of convergence of the algorithm, by means of suitable choices of the sequence $\{\eta_k\}$, is a key point for this criterion. On the other hand, some drawbacks deriving from the adoption of this rule are well known. Indeed, at the $j$-th inner iteration of the Krylov subspace method adopted for solving the Newton equation (2.74), a stationary point of the quadratic model

$$q_k(p_k) = \frac{1}{2}p_k^T \nabla^2 f(x_k)p_k + \nabla f(x_k)^T p_k \tag{2.76}$$

over the Krylov subspace

$$K_j(\nabla^2 f(x_k), \nabla f(x_k))$$

is sought. In the case of positive definite Hessian $\nabla^2 f(x_k)$, the quadratic model (2.76) has a global minimizer which exactly solves the Newton equation (2.74). Of course, this case corresponds to a null residual. Conversely, whenever an approximate solution is sought, monitoring the magnitude of the residual might be, as discussed in [90], misleading. Indeed, the actual decrease of the objective function values can be alternatively predicted by means of the quadratic model decrease; however, the magnitude of the residual $r_k$ and the quadratic model $q_k(p_k)$ could be significantly different. Moreover, the rounding error in computing $\|r_k\|$ could be relevant since $r_k$ is usually computed by recurrence. In addition, whenever a CG method is used in the inner iterations and $H_k$ is positive definite, the quadratic model monotonically decreases as the inner iterations progress, while the sequence $\{\|r_k\|\}$ is not monotone (unlike, for instance, using MINRES (see [54])).

These remarks induced the authors to propose in [90] also a truncation rule based on *the decrease of the quadratic model*, rather than considering only the residual. Namely, the truncation criterion proposed is the following: the inner iterations are terminated if, for a specified value of $\eta_k \in (0, 1)$,

$$\frac{\dfrac{q_k(p_j) - q_k(p_{j-1})}{j}}{\dfrac{q_k(p_j)}{}} \leq \eta_k, \tag{2.77}$$

where $p_j$ denotes the approximate solution of (2.74) at the $j$-th inner iteration. This criterion is then based on the comparison between the reduction of the quadratic model $q_k(p_j) - q_k(p_{j-1})$, and the average reduction per iteration $q_k(p_j)/j$. The criterion (2.77) is often considered preferable to (2.75), since it gains information directly from the values of the quadratic model. Moreover, in [48] it was extended to possibly consider also an indefinite Hessian matrix $\nabla^2 f(x_k)$, providing some theoretical results, too.

However, in the framework of Newton-Krylov methods, in unconstrained as well as in constrained optimization, some codes currently available on the web and commonly used by the optimizers community still adopt the residual-based truncation criterion (2.75), both within linesearch-based and trust region-based codes (see e.g. [35], [77], [85] - page 9, [103], [104], [72], [112] and URL https://neos-guide.org/content/truncated-newton-methods).This might be due also to the fact that, as well known, the adoption of (2.75) ensures theoretical superlinear convergence. Conversely, the criterion based on the quadratic model reduction (2.77), with the suggested value of $\eta_k = 0.5$ (constant), guarantees only the theoretical linear rate of convergence [90], even if it works very efficiently in practice.

In the following, the standard linesearch-based Newton-Krylov method is reported.

**Linesearch-based Newton-Krylov algorithm**

**Data**: $x_1 \in \mathbb{R}^n$;

Set $k = 1$ and compute $\nabla f(x_1)$;

Set $\eta_k \in [0, 1)$ for any $k$, with $\{\eta_k\} \to 0$;

**OUTER ITERATIONS**

**For** $k = 1, \ldots$

if $\|\nabla f(x_k)\|$ is small then STOP;

**INNER ITERATIONS**

compute $p_k$ which approximately solves $\nabla^2 f(x_k)p = -\nabla f(x_k)$ and satisfies the truncation rule (see (2.75) and (2.77));

compute the steplength $\alpha_k$ by using a linesearch procedure;

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

**End For**

## 2.2 Conclusions

In this chapter we have focused on methods for solving large scale unconstrained optimization problems. First of all we started with NCG methods and their preconditioned versions: however, the search direction $p_k$ could be not well scaled. Later on we introduced quasi-Newton methods and their main schemes, like L-BFGS: these methods are fairly robust, inexpensive, and easy to implement, but they do not converge rapidly. Finally, inexact Newton methods have been described. They have attractive global convergence properties and may be superlinearly convergent for suitable choices of parameters.

# Chapter 3

# Preconditioners based on quasi-Newton updates for NCG methods

In order to solve large scale unconstrained problems, in this chapter we propose new preconditioners to be used within the NCG method. The rationale behind this proposal draws inspiration from quasi-Newton updates, and its aim is to possibly approximate in some sense the inverse of the Hessian matrix. In particular, at the current iteration of the NCG we consider some preconditioners based on new low-rank quasi-Newton symmetric updating formulae, obtained as by-product of the NCG method at the previous steps. Observe that the main focus of this study is not to define a challenging algorithm for large scale unconstrained optimization, but it aims at introducing a preconditioning strategy and showing its effectiveness. Most of the material of this chapter is contained in [5], [31], [32], [33].

## 3.1 Introduction

In this chapter we consider the large scale unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x), \tag{3.1}$$

where $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is a real valued function and the dimension $n$ is large. It is assumed that $f$ is a twice continuously differentiable function and that for a given $x_1 \in \mathbb{R}^n$ the level set $\mathcal{L}_1 = \{x \in \mathbb{R}^n \ : \ f(x) \leq f(x_1)\}$ is compact.

In the case we consider the minimization of a convex quadratic function, then the NCG and the BFGS quasi-Newton method show a well studied correspondence of the search directions they respectively generate [92]. This spots some light on the relation between the latter two classes of methods, and we want to partially exploit benefits from possibly coupling their respective underlying ideas.

On this purpose let us first consider the PNCG method. We recall (see Section 2.1.1.1) essentially three choices at current step $k$ strongly determine both the effectiveness and the efficiency of the overall method. In particular, the first choice refers to the *linesearch procedure*, which selects the steplength $\alpha_k > 0$ used to

compute the next iterate $x_{k+1}$, being

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

Then, the second choice refers to the selection of the parameter $\beta_{k+1}$, which determines the next search direction as

$$p_{k+1} = -g_{k+1} + \beta_{k+1} p_k.$$

Finally, a proper choice $M_{k+1} \in \mathbb{R}^{n \times n}$ for a preconditioner may also be part of the computation of $p_{k+1}$, as in

$$p_{k+1} = -M_{k+1} g_{k+1} + \beta_{k+1} p_k.$$

The latter three choices are not independent, inasmuch as for instance an improper preconditioner risks to possibly destroy both convergence properties and numerical performance of the PNCG. This observation imposes some care before adopting a preconditioner, in order to first verify that it complies with the requirements claimed by the convergence analysis.

Observe that addressing good preconditioners for NCG methods still remains an intriguing research issue, in particular when the solution of large scale problems is sought and no structure of the problem is known in advance [20, 38, 67, 101, 111]. Similarly, matrix-free preconditioning for linear systems or sequences of linear systems is currently an appealing research topic, too (see e.g. [17, 18, 48, 49]). On this guideline, this study is devoted to investigate ideas from quasi-Newton updates, in order to build possible preconditioners for NCG. In particular, we are interested both to obtain a numerically efficient preconditioner, and to analyze its theoretical properties. The preconditioners we propose are iteratively constructed and partially recover the structure of quasi-Newton updates. To sum up, we definitely remark that our preconditioners are designed for the PNCG and

- do not rely on the structure of the minimization problem in hand;
- are matrix-free, hence they are naturally conceived for large scale problems;
- are built drawing inspiration from quasi-Newton schemes;
- convey information from previous iterations of the PNCG.

Finally, for the sake of completeness we urge to recall that the idea of using a quasi-Newton update as a possible preconditioner, within NCG algorithms, is not new; examples of such an approach can be found for instance in [13, 26, 83].

## 3.2 Preliminaries

Consider the general scheme of a PNCG algorithm in Section 2.1.1.1. As already observed the steplength $\alpha_{k+1}$ and the parameter $\beta_{k+1}$ can be chosen in a variety of ways, in order to ensure convergence properties or to improve the overall efficiency (see e.g. [65, 67]). Here we neither intend to propose a novel choice of $\beta_{k+1}$, nor we want to consider any specific linesearch procedure to compute $\alpha_{k+1}$ for

PNCG algorithm. To this regards, Wolfe conditions are well-suited for our purposes, inasmuch as they easily guarantee that the curvature condition (2.15)

$$s_k^T y_k > 0$$

is fulfilled for any $k$.

As already said, preconditioning is usually applied for increasing the efficiency of the NCG method. In this regard, in this study we exploit some matrix updates and their capability to possibly mimic quasi-Newton schemes, in order to generate a 'reasonable' approximation of the inverse Hessian matrix and use it as a preconditioner within a PNCG framework. Among well known reasons for the success of the L-BFGS method we find that $H_{k+1}$ is the positive definite matrix 'as close as possible' (in the sense of Frobenius norm) to the matrix $H_k$, and satisfying the *secant equation*

$$H_{k+1} y_k = s_k.$$

Now, we explicitly focus on the case where $f(x)$ is quadratic, i.e. $f(x) = 1/2 x^T A x - b^T x$, with $A \in \mathbb{R}^{n \times n}$ positive definite and $b \in \mathbb{R}^n$. The latter case is particularly appealing to our purposes, since it allows to exploit the strong relation between BFGS and the conjugacy of search directions with respect to matrix $A$ [92]. Indeed, the BFGS update (2.67) is explicitly given by

$$H_k = \left( I - \frac{y_{k-1} s_{k-1}^T}{y_{k-1}^T s_{k-1}} \right)^T H_{k-1} \left( I - \frac{y_{k-1} s_{k-1}^T}{y_{k-1}^T s_{k-1}} \right) + \frac{s_{k-1} s_{k-1}^T}{y_{k-1}^T s_{k-1}}, \tag{3.2}$$

and recalling the expression of $f(x)$, along with relation $y_k = A s_k$, (2.66) can be reformulated as

$$V_k = I - \frac{A s_k s_k^T}{s_k^T A s_k}, \tag{3.3}$$

where the vectors $\{p_1, \ldots, p_k\}$ are mutually conjugate with respect to $A$. Then, using BFGS recursion (3.2), we can write

$$\begin{aligned} H_k &= V_{k-1}^T H_{k-1} V_{k-1} + \frac{s_{k-1} s_{k-1}^T}{y_{k-1}^T s_{k-1}} \\ &= V_{k-1}^T (V_{k-2}^T H_{k-2} V_{k-2}) V_{k-1} + V_{k-1}^T \frac{s_{k-2} s_{k-2}^T}{y_{k-2}^T s_{k-2}} V_{k-1} + \frac{s_{k-1} s_{k-1}^T}{y_{k-1}^T s_{k-1}}. \end{aligned} \tag{3.4}$$

Finally, the conjugacy among vectors $\{p_1, \ldots, p_k\}$ with respect to $A$ implies that for any $k$

$$V_k^T s_{k-1} = \left( I - \frac{A s_k s_k^T}{s_k^T A s_k} \right)^T s_{k-1} = s_{k-1},$$

and again recalling that $y_k = A s_k$, relation (3.4) yields

$$H_k = V_{k-1}^T V_{k-2}^T \cdots V_1^T H_k^0 V_1 \cdots V_{k-2} V_{k-1} + \sum_{i=1}^{k-1} \frac{s_i s_i^T}{s_i^T A s_i}. \tag{3.5}$$

Formula (3.5) can be used to potentially generate preconditioners for the PNCG, by looking at the rightmost contribution

$$\sum_{i=1}^{k-1} \frac{s_i s_i^T}{s_i^T A s_i}, \tag{3.6}$$

whose range is exactly *span* $\{s_1, \ldots, s_{k-1}\}$.Indeed, we can draw our inspiration from (3.5) and [48], where a new preconditioner for Newton-Krylov methods is described. In particular, in [48] the set of directions generated by a Krylov subspace method is used to provide an approximate inverse preconditioner, for the solution of Newton's systems. On this guideline, observe that when $f(x)$ is quadratic, with $A$ positive definite, the CG method may generate $n$ conjugate directions $\{p_j\}$ (see e.g. [54]) such that

$$A^{-1} = \sum_{j=1}^{n} \frac{p_j p_j^T}{p_j^T A p_j}. \tag{3.7}$$

This implies that the rightmost contribution in (3.5) might be viewed and used as an approximate inverse of the Hessian matrix $A$. In the following we aim at extending the latter idea, to the case where $f(x)$ is nonlinear, following similar guidelines.

## 3.3 Guidelines for new Symmetric Rank-2 updates

In this section we consider new quasi-Newton updating formulae, by considering the properties of a parameter dependent Symmetric Rank-2 (SR2) update of the inverse of the Hessian matrix. Suppose that after $k$ iterations of NCG the sequence of iterates $\{x_1, \ldots, x_{k+1}\}$ is generated. Then, the idea is that of building a structured preconditoner $M_{k+1}$ which satisfies the secant equation at least at the current iteration, i.e.

$$M_{k+1} y_k = s_k. \tag{3.8}$$

Observe that the latter appealing property of the matrix $M_{k+1}$ is satisfied by all the updates of the Broyden class, provided that the linesearch adopted is exact (see e.g. [95]). We would like to recover the motivation underlying the latter class of updates, and by using a novel rank-2 updates we would like to define preconditioners for PNCG.

In a more general framework, similarly to the Broyden class, we address a family of preconditioners of the form

$$M_{k+1}(\gamma_{k+1}) = M_k(\gamma_k) + \Delta_k, \qquad \Delta_k \in \mathbb{R}^{n \times n}, \text{ symmetric}, \tag{3.9}$$

where the sequence $\{M_k(\gamma_k)\}$ depends on the parameters $\gamma_k \in \mathbb{R}^p$, $p \geq 1$, and provides an approximation of $[\nabla^2 f(x)]^{-1}$ according with (3.7). The new update $M_{k+1}(\gamma_{k+1})$ is claimed to satisfy the following appealing conditions:

(1) $M_{k+1}(\gamma_{k+1})$ is well-defined and nonsingular;

(2) $M_{k+1}(\gamma_{k+1})$ can be iteratively updated;

(3) $M_{k+1}(\gamma_{k+1})$ collects information from the iterations $k-m, k-m+1, \ldots, k$, of an NCG method, being $m \geq 0$ integer;

(4) $M_{k+1}(\gamma_{k+1})$ satisfies the secant equation at a subset $K$ of iterations, with $K \subseteq \{1, 2, \ldots, k\}$;

(5) $M_{k+1}(\gamma_{k+1})$ "tends to preserve" in some sense the inertia of the inverse Hessian $\left[\nabla^2 f(x_{k+1})\right]^{-1}$, in case $f(x)$ is a quadratic function.

Observe that the Symmetric Rank-1 (SR1) quasi-Newton update (see Section 6.2 in [95]) satisfies properties *(2)-(5)* but not the property (1), i.e. it might be possibly not well-defined for a general nonlinear function. The latter result follows from the fact that SR1 update provides only a rank-1 quasi-Newton update, unlike BFGS and DFP. On the other hand, while BFGS and DFP quasi-Newton formulae provide only positive definite updates, the SR1 formula is able to recover the inertia of the Hessian matrix, by generating possibly indefinite updates. Thus, now we want to study SR2 quasi-Newton updates, such that at iteration $k$

- they satisfy *(1)-(5)*;
- at least one of the newest dyads used for the update is provided using information from iterations $k - m, \ldots, k$ of the NCG method.

In the following sections we introduce some new quasi-Newton updating formulae in order to build preconditioners for PNCG.

### 3.3.1 A new Symmetric Rank-2 update

In this section we want to study an SR2 quasi-Newton update, which satisfies *(1)-(5)* in Section 3.3 and where one of the two newest dyads of the update is provided by information from the NCG method. To this aim, assuming that $M_k = M(\gamma_k)$ is given, we consider the relation (3.9) where we set

$$\Delta_k = \gamma_k^{(1)} v_k v_k^T + \gamma_k^{(2)} \frac{p_k p_k^T}{y_k^T p_k}, \qquad \gamma_k^{(1)}, \gamma_k^{(2)} \in \mathbb{R} \setminus \{0\}, \qquad v_k \in \mathbb{R}^n,$$

and $p_k$ is generated at the $k-$th iteration of the (unpreconditioned) NCG method. Thus, we will have the new update

$$M_{k+1} = M_k + \gamma_k^{(1)} v_k v_k^T + \gamma_k^{(2)} \frac{p_k p_k^T}{y_k^T p_k}, \qquad \gamma_k^{(1)}, \gamma_k^{(2)} \in \mathbb{R} \setminus \{0\}, \qquad v_k \in \mathbb{R}^n, \quad (3.10)$$

and in order to satisfy the secant equation $M_{k+1} y_k = s_k$ the following equality must hold

$$M_k y_k + \gamma_k^{(1)} (v_k^T y_k) v_k + \gamma_k^{(2)} \frac{p_k p_k^T}{y_k^T p_k} y_k = s_k,$$

that is

$$\gamma_k^{(1)} (v_k^T y_k) v_k = s_k - M_k y_k - \gamma_k^{(2)} p_k. \quad (3.11)$$

Therefore it results

$$v_k = \sigma_k \left( s_k - M_k y_k - \gamma_k^{(2)} p_k \right), \quad (3.12)$$

for some scalar $\sigma_k \in \mathbb{R}$. By substituting the expression (3.12) of $v_k$ in (3.11) we have

$$\gamma_k^{(1)} \sigma_k^2 \left[ y_k^T (s_k - M_k y_k - \gamma_k^{(2)} p_k) \right] (s_k - M_k y_k - \gamma_k^{(2)} p_k) = s_k - M_k y_k - \gamma_k^{(2)} p_k.$$

Thus, the following relation among the parameters $\gamma_k^{(1)}$, $\sigma_k$ and $\gamma_k^{(2)}$ must hold

$$\gamma_k^{(1)}\sigma_k^2 = \frac{1}{s_k^T y_k - y_k^T M_k y_k - \gamma_k^{(2)} p_k^T y_k}. \tag{3.13}$$

Note that from the arbitrariness of $\gamma_k^{(1)}$, without loss of generality, we can set $\sigma_k \in \{-1, 1\}$.

Now, in the next proposition we first consider the case of quadratic functions, and prove that the update (3.10) satisfies the secant equation, along all previous directions.

**Proposition 3.3.1.** *Assume that $f$ is the quadratic function $f(x) = \frac{1}{2}x^T Ax + b^T x$, where $A \in \mathbb{R}^{n\times n}$ is symmetric and $b \in \mathbb{R}^n$. Suppose that $k$ steps of the (unpreconditioned) CG are performed, in order to detect the stationary point (if any) of the function $f$, and that the vectors $p_1, \ldots, p_k$ are generated. Then, the matrix $M_{k+1}$ in* (3.10) *satisfies the secant equations*

$$M_{k+1} y_j = s_j, \qquad j = 1, \ldots, k, \tag{3.14}$$

*provided that the nonzero coefficients $\gamma_j^{(1)}$, $\gamma_j^{(2)}$, $j = 1, \ldots, k$ are computed such that*

$$\begin{aligned} \gamma_j^{(1)} &= \frac{1}{s_j^T y_j - y_j^T M_j y_j - \gamma_j^{(2)} p_j^T y_j}, && j = 1, \ldots, k, \\ \gamma_j^{(2)} &\neq \frac{s_j^T y_j - y_j^T M_j y_j}{p_j^T y_j}, && j = 1, \ldots, k. \end{aligned} \tag{3.15}$$

*Proof.* The proof proceeds by induction. Equations (3.14) hold for $k = 1$, that is $M_2 y_1 = s_1$, as long as

$$s_1 = \left[ M_1 + \gamma_1^{(1)}\sigma_1^2 (s_1 - M_1 y_1 - \gamma_1^{(2)} p_1)(s_1 - M_1 y_1 - \gamma_1^{(2)} p_1)^T + \gamma_1^{(2)} \frac{p_1 p_1^T}{y_1^T p_1} \right] y_1,$$

or equivalently

$$s_1 - M_1 y_1 - \gamma_1^{(2)} p_1 = \gamma_1^{(1)} (s_1^T y_1 - y_1^T M_1 y_1 - \gamma_1^{(2)} p_1^T y_1) \left[ s_1 - M_1 y_1 - \gamma_1^{(2)} p_1 \right],$$

which is satisfied selecting $\gamma_1^{(1)}$ and $\gamma_1^{(2)}$ according with (3.15).
Now, suppose that the relations (3.14) hold for the index $k-1$. To complete the induction we need to prove that the relations (3.14) hold for the index $k$. Firstly, note that $M_{k+1} y_k = s_k$ holds. In fact

$$s_k = \left[ M_k + \gamma_k^{(1)}\sigma_k^2 (s_k - M_k y_k - \gamma_k^{(2)} p_k)(s_k - M_k y_k - \gamma_k^{(2)} p_k)^T + \gamma_k^{(2)} \frac{p_k p_k^T}{y_k^T p_k} \right] y_k$$

holds if and only if

$$s_k - M_k y_k - \gamma_k^{(2)} p_k = \gamma_k^{(1)} (s_k^T y_k - y_k^T M_k y_k - \gamma_k^{(2)} p_k^T y_k)(s_k - M_k y_k - \gamma_k^{(2)} p_k),$$

and the latter holds from (3.15) with $j = k$. Now, we have to prove that (3.14) hold for any $j < k$. For $j < k$ we have

$$M_{k+1}y_j = M_k y_j + \gamma_k^{(1)}\sigma_k^2(s_k - M_k y_k - \gamma_k^{(2)}p_k)(s_k - M_k y_k - \gamma_k^{(2)}p_k)^T y_j + \gamma_k^{(2)}\frac{p_k^T y_j}{y_k^T p_k}p_k,$$

where $M_k y_j = s_j$ by the inductive hypothesis. Moreover,

$$(s_k - M_k y_k)^T y_j = s_k^T y_j - y_k^T M_k y_j = s_k^T y_j - y_k^T s_j = s_k^T A s_j - (As_k)^T s_j = 0,$$

where the third equality holds since $y_j = As_j$, for any $j$, for the quadratic function $f$. Finally,

$$\gamma_k^{(2)}p_k^T y_j = \gamma_k^{(2)}p_k^T A s_j = \gamma_k^{(2)}\alpha_j p_k^T A p_j = 0,$$

which follows from the conjugacy of the directions $\{p_1, \ldots, p_k\}$ generated by the CG. Thus, (3.14) hold for any $j \leq k$ and the induction is complete. □

As an immediate consequence of the previous proposition, we prove now the finite termination property for a quadratic function, i.e. after at most $n$ steps, $M_{n+1}$ is the inverse of the Hessian of the quadratic function.

**Corollary 3.3.2.** *Assume that $f$ is the quadratic function $f(x) = \frac{1}{2}x^T Ax + b^T x$, where $A \in \mathbb{R}^{n\times n}$ is symmetric and $b \in \mathbb{R}^n$. Suppose that $n$ steps of the (unpreconditioned) CG are performed, in order to detect the stationary point of the function $f$, and that the vectors $p_1, \ldots, p_n$ are generated. If* (3.15) *holds, we have $M_{n+1} = A^{-1}$.*

*Proof.* By applying Proposition 3.3.1, we have that (3.14) hold for $k = n$, i.e.

$$M_{n+1}y_j = s_j, \qquad j = 1, \ldots, n.$$

Since $f$ is quadratic then $y_j = As_j$, for any $j$, i.e.

$$M_{n+1}As_j = s_j, \qquad j = 1, \ldots, n.$$

Now, since $s_j = \alpha_j p_j$, $j = 1, \ldots, n$, the conjugacy of the vectors $\{p_1, \ldots, p_n\}$ implies that $M_{n+1} = A^{-1}$. □

We highlight that, whenever $k = n$, Corollary 3.3.2 justifies the first part of the statement *(5)* in Section 3.3. Moreover, later on we show that for $k < n$, the update matrix in (3.10) can be suitably modified to provide a preconditioner.

After analyzing the case of $f(x)$ quadratic, we turn now to the general case of a nonlinear twice continuously differentiable function. In particular, since we are interested in using the matrix $M_{k+1}$ in (3.10) as a preconditioner, we need to investigate if there exists a suitable setting of the parameters such that $M_{k+1}$ is positive definite, provided that (3.15) are satisfied. In the next proposition we prove that if the parameter $\gamma_k^{(2)}$ is below a threshold value, then the matrix $M_{k+1}$ is *almost always* positive definite.

**Proposition 3.3.3.** *Let $f$ be a nonlinear twice continuously differentiable function. Suppose that the (unpreconditioned) NCG method is used to minimize the function $f$. Suppose that* (3.15) *is satisfied and*

$$0 \leq \gamma_k^{(2)} < \frac{s_k^T y_k - y_k^T M_k y_k}{p_k^T y_k}, \tag{3.16}$$

*with*

$$y_k^T s_k + y_k^T M_k y_k \leq 0 \qquad or \qquad y_k^T s_k - y_k^T M_k y_k \geq 0, \tag{3.17}$$

*where $s_j = \alpha_j p_j$. Then the matrix $M_{k+1}$ in* (3.10) *is positive definite.*

*Proof.* By substituting (3.12) in (3.10), recalling that $\sigma_k^2 = 1$ we obtain

$$\begin{aligned} M_{k+1} = \quad & M_k + \gamma_k^{(1)} \left[ \left(\alpha_k - \gamma_k^{(2)}\right)^2 p_k p_k^T - \left(\alpha_k - \gamma_k^{(2)}\right) \left( (M_k y_k)\, p_k^T + p_k\, (M_k y_k)^T \right) \right. \\ & \left. + (M_k y_k)\, (M_k y_k)^T \right] + \gamma_k^{(2)} \frac{p_k p_k^T}{y_k^T p_k}. \end{aligned}$$

Hence $M_{k+1}$ can be rewritten in the form

$$\begin{pmatrix} p_k & \vdots & M_k y_k \end{pmatrix} \begin{pmatrix} \gamma_k^{(1)}(\alpha_k - \gamma_k^{(2)})^2 + \dfrac{\gamma_k^{(2)}}{y_k^T p_k} & & -\gamma_k^{(1)}(\alpha_k - \gamma_k^{(2)}) \\ & & \\ -\gamma_k^{(1)}(\alpha_k - \gamma_k^{(2)}) & & \gamma_k^{(1)} \end{pmatrix} \begin{pmatrix} p_k^T \\ \dots \\ (M_k y_k)^T \end{pmatrix}.$$

Therefore $M_{k+1}$ is positive definite if and only if the following inequalities hold:

$$\begin{aligned} \gamma_k^{(1)}(\alpha_k - \gamma_k^{(2)})^2 + \frac{\gamma_k^{(2)}}{y_k^T p_k} &> 0 \\ \gamma_k^{(1)} \left( \gamma_k^{(1)}(\alpha_k - \gamma_k^{(2)})^2 + \frac{\gamma_k^{(2)}}{y_k^T p_k} \right) - (\gamma_k^{(1)})^2 (\alpha_k - \gamma_k^{(2)})^2 &> 0. \end{aligned} \tag{3.18}$$

Using the expression of $\gamma_k^{(1)}$ in (3.13) and recalling that $y_k^T s_k > 0$ (as a consequence of the Wolfe conditions), (3.18) are equivalent to

$$\begin{aligned} \frac{(\alpha_k - \gamma_k^{(2)})^2 y_k^T p_k}{(\alpha_k - \gamma_k^{(2)}) p_k^T y_k - y_k^T M_k y_k} + \gamma_k^{(2)} &> 0 \\ \frac{\gamma_k^{(2)}}{(\alpha_k - \gamma_k^{(2)}) p_k^T y_k - y_k^T M_k y_k} &> 0. \end{aligned}$$

After some computation we obtain that there exist values of the parameter $\gamma_k^{(2)}$ for which the latter inequalities admit solutions, with only one exception. In fact, they are satisfied for any value of $\gamma_k^{(2)}$ such that

$$0 \leq \gamma_k^{(2)} < \frac{\alpha_k p_k^T y_k - y_k^T M_k y_k}{p_k^T y_k}$$

but they do not admit solution in case

$$\alpha_k y_k^T p_k + y_k^T M_k y_k > 0 \qquad \text{and} \qquad \alpha_k y_k^T p_k - y_k^T M_k y_k < 0,$$

i.e. when (3.17) does not hold. □

From Proposition 3.3.1 and Corollary 3.3.2, we could use the matrix $M_{k+1}$ as an approximate inverse of $\nabla^2 f(x)$. However, Proposition 3.3.3 evidences that conditions (3.15) and (3.16) do not suffice to ensure $M_{k+1}$ positive definite. In fact, whenever (3.17) occurs, additional safeguard is needed since $M_{k+1}$ is possibly indefinite. Thus, the definition of $M_{k+1}$ should be possibly modified in order to obtain positive definite updates. On the other hand, the scheme PNCG can be also modified to cope with the case of $M_{k+1}$ indefinite in (2.10). In particular, Step While may be replaced by the following instructions:

**While** $g_k \neq 0$

compute the steplength $\alpha_k$ by using a linesearch procedure along $p_k$;

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

if $M_{k+1} g_{k+1} = 0$, then set $M_{k+1} \leftarrow I$

compute $\beta_{k+1}$;

$$p_{k+1} = -M_{k+1} g_{k+1} + \beta_{k+1} p_k;$$

$$k = k + 1$$

**End While**

### 3.3.2 A Generalized Symmetric Rank-2 update

In the previous section we have introduced a SR2 update as a good approximation of the inverse of the Hessian matrix to be used as preconditioner; however, the main drawback is that this preconditioner could be not positive definite. In this section, similarly to the previous one, we again study a new quasi-Newton updating formula, by considering the properties of a parameter dependent SR2 update of the inverse of the Hessian matrix, which is alternative with respect to the expression (3.10). To this aim, assuming that $M_k = M(\gamma_k)$ is given, we consider the relation (3.9) where now we set

$$\Delta_k = \gamma_k^{(1)} v_k v_k^T + \gamma_k^{(2)} \frac{p_k p_k^T}{y_k^T p_k} + \gamma_k^{(3)} (p_k v_k^T + v_k p_k^T),$$

where $\gamma_k^{(1)}, \gamma_k^{(2)}, \gamma_k^{(3)} \in \mathbb{R} \setminus \{0\}$, $v_k \in \mathbb{R}^n$, and $p_k$ is generated at the $k-$th iteration of the (unpreconditioned) NCG method. Thus we will have the new update

$$M_{k+1} = M_k + \gamma_k^{(1)} v_k v_k^T + \gamma_k^{(2)} \frac{p_k p_k^T}{y_k^T p_k} + \gamma_k^{(3)} (p_k v_k^T + v_k p_k^T), \tag{3.19}$$

for any $\gamma_k^{(1)}, \gamma_k^{(2)}, \gamma_k^{(3)} \in \mathbb{R} \setminus \{0\}, v_k \in \mathbb{R}^n$, where we set

$$v_k^T y_k = 0, \tag{3.20}$$

for any $k$. Thus, in order to satisfy the secant equation $M_{k+1} y_k = s_k$ the following equality must hold

$$\left( M_k + \gamma_k^{(1)} v_k v_k^T + \gamma_k^{(2)} \frac{p_k p_k^T}{y_k^T p_k} + \gamma_k^{(3)} (p_k v_k^T + v_k p_k^T) \right) y_k = s_k,$$

that is by (3.20)

$$M_k + \gamma_k^{(2)} p_k + \gamma_k^{(3)} v_k p_k^T y_k = s_k. \tag{3.21}$$

Therefore, (3.21) is satisfied provided that

$$v_k = \frac{s_k - M_k y_k - \gamma_k^{(2)} p_k}{\gamma_k^{(3)} p_k^T y_k}, \tag{3.22}$$

for some scalar $\gamma_k^{(3)} \in \mathbb{R} \setminus \{0\}$. Note that to satisfy also (3.20) it suffices to set

$$0 = v_k^T y_k \qquad \Longrightarrow \qquad \gamma_k^{(2)} = \frac{s_k^T y_k - y_k^T M_k y_k}{p_k^T y_k}. \tag{3.23}$$

Now, in the next proposition we first consider the case of quadratic functions, and prove that the update (3.19) satisfies the secant equation, along all previous directions.

**Proposition 3.3.4.** *Assume that $f$ is the quadratic function $f(x) = \frac{1}{2} x^T A x + b^T x$, where $A \in \mathbb{R}^{n \times n}$ is symmetric and $b \in \mathbb{R}^n$. Suppose that $k$ steps of the (unpreconditioned) CG are performed, in order to detect the stationary point (if any) of the function $f$, and that the vectors $p_1, \ldots, p_k$ are generated. Then, the matrix $M_{k+1}$ in* (3.19) *satisfies the secant equations*

$$M_{k+1} y_j = s_j, \qquad j = 1, \ldots, k, \tag{3.24}$$

*provided that* (3.22) *holds with*

$$\gamma_j^{(2)} = \frac{s_j^T y_j - y_j^T M_j y_j}{p_j^T y_j} \neq 0. \tag{3.25}$$

*Proof.* The proof proceeds by induction. Equations (3.24) hold for $k = 1$, that is $M_2 y_1 = s_1$, as long as

$$s_1 = \left[ M_1 + \gamma_1^{(1)} v_1 v_1^T + \gamma_1^{(2)} \frac{p_1 p_1^T}{y_1^T p_1} + \gamma_1^{(3)} (p_1 v_1^T + v_1 p_1^T) \right] y_1,$$

or equivalently

$$s_1 - M_1 y_1 - \gamma_1^{(2)} p_1 = \gamma_1^{(1)} (v_1^T y_1) v_1 + \gamma_1^{(3)} ((v_1^T y_1) p_1 + (p_1^T y_1) v_1),$$

which is satisfied with (3.25).

Now, suppose that the relations (3.24) hold for the index $k-1$. To complete the induction we need to prove that the relations (3.24) hold for the index $k$. Firstly, note that $M_{k+1}y_k = s_k$ holds. In fact

$$s_k = \left[ M_k + \gamma_k^{(1)} v_k v_k^T + \gamma_k^{(2)} \frac{p_k p_k^T}{y_k^T p_k} + \gamma_k^{(3)} (p_k v_k^T + v_k p_k^T) \right] y_k$$

holds if and only if

$$s_k - M_k y_k - \gamma_k^{(2)} p_k = \gamma_k^{(1)} (v_k^T y_k) v_k + \gamma_k^{(3)} ((v_k^T y_k) p_k + (p_k^T y_k) v_k),$$

and the latter holds from (3.25) with $j = k$. Now, we have to prove that (3.24) hold for any $j < k$. For $j < k$ we have

$$M_{k+1} y_j = M_k y_j + \gamma_k^{(1)} (v_k v_k^T) y_j + \gamma_k^{(2)} \left( \frac{p_k p_k^T}{y_k^T p_k} \right) y_j + \gamma_k^{(3)} (p_k v_k^T + v_k p_k^T) y_j,$$

where $M_k y_j = s_j$ by the inductive hypothesis. Moreover,

$$p_k^T y_j = p_k^T A s_j = \alpha_j p_k^T A p_j = 0,$$

which follows from the conjugacy of the directions $\{p_1, \ldots, p_k\}$ generated by the CG. Finally, since $y_j = A s_j$, for any $j$, then

$$s_k^T y_j = \alpha_k p_k^T (A s_j) = \alpha_k \alpha_j p_k^T A p_j = 0,$$

$$y_k^T M_k y_j = y_k^T s_j = s_k^T A s_j = 0.$$

Thus, thanks to the previous relations, we have by (3.22)

$$v_k^T y_j = \frac{s_k^T y_j - y_k^T M_k y_j - \gamma_k^{(2)} p_k^T y_j}{\gamma_k^{(3)} p_k^T y_k} = 0.$$

Thus, (3.24) hold for any $j \leq k$ and the induction is complete. □

After analyzing the case of $f(x)$ quadratic, we turn now to the general case of a nonlinear twice continuously differentiable function. In particular, since we are interested in using the matrix $M_{k+1}$ in (3.19) as a preconditioner, we need to investigate if there exists a suitable setting of the parameters such that $M_{k+1}$ is positive definite, provided that (3.25) are satisfied. In the next proposition we investigate values of the parameters $\gamma_k^{(1)}$, $\gamma_k^{(2)}$ and $\gamma_k^{(3)}$ such that the matrix $M_{k+1}$ in (3.19) is positive definite.

**Proposition 3.3.5.** *Let $f$ be a nonlinear twice continuously differentiable function. Suppose that the (unpreconditioned) NCG method is used to minimize the function $f$. Suppose that* (3.25) *is satisfied and*

$$\begin{aligned} \gamma_j^{(2)} &> 0, \qquad j = 1, \ldots, k, \\ \gamma_j^{(1)} \gamma_j^{(2)} &> (\gamma_j^{(3)})^2 (y_j^T p_j), \qquad j = 1, \ldots, k. \end{aligned} \tag{3.26}$$

*Then, if $M_1 \succ 0$ the matrix $M_{k+1}$ in* (3.19) *is positive definite.*

*Proof.* From (3.19), we have

$$M_{k+1} = M_k + \Delta_k$$

where

$$\Delta_k = \gamma_k^{(1)} v_k v_k^T + \gamma_k^{(2)} \frac{p_k p_k^T}{y_k^T p_k} + \gamma_k^{(3)} (p_k v_k^T + v_k p_k^T).$$

Hence $\Delta_k$ can be rewritten in the form

$$\begin{pmatrix} p_k & \vdots & v_k \end{pmatrix} \begin{pmatrix} \frac{\gamma_k^{(2)}}{y_k^T p_k} & & \gamma_k^{(3)} \\ & & \\ \gamma_k^{(3)} & & \gamma_k^{(1)} \end{pmatrix} \begin{pmatrix} p_k^T \\ \dots \\ v_k^T \end{pmatrix}.$$

Therefore $M_{k+1}$ is positive definite if $M_k \succ 0$ and the following inequalities hold:

$$\begin{aligned} \frac{\gamma_k^{(2)}}{y_k^T p_k} &> 0 \\ \gamma_k^{(1)} \left( \frac{\gamma_k^{(2)}}{y_k^T p_k} \right) - (\gamma_k^{(3)})^2 &> 0. \end{aligned} \tag{3.27}$$

Recalling that $y_k^T s_k > 0$ (as a consequence of the Wolfe conditions), (3.27) are equivalent to

$$\begin{aligned} \gamma_k^{(2)} &> 0 \\ \gamma_k^{(1)} \gamma_k^{(2)} &> (\gamma_k^{(3)})^2 (y_k^T p_k). \end{aligned}$$

□

Note that, by considering (3.26), thanks to the first inequalities we have that $\gamma_k^{(2)}$ is greater than zero. From the second inequalities we have again that $\gamma_k^{(1)}$ is greater than zero and $\gamma_k^{(3)}$ must be non-zero.

From the previous Proposition 3.3.5, we could use the matrix $M_{k+1}$ as an approximate inverse of $\nabla^2 f(x)$, in a preconditioning framework, provided that $M_{k+1} \succ 0$. Observe that a consideration similar to the one in the end of Section 3.3.1 holds, so that the Step While in PNCG scheme might be suitably modified.

### 3.3.3 A Symmetric Rank-2 update based on modified weak secant equation

In the previous section we have introduced a novel SR2 update. From Proposition 3.3.5 we have simple conditions to ensure that our preconditioner is positive definite. However, considering the expression of $\gamma_k^{(2)}$ in (3.23), we can see that the first inequalities in (3.26) could be not satisfied and, for this reason, the preconditioner could be not again positive definite (as in Section 3.3.1). On this guidelines,

in this section we study a quasi-Newton updating formula, by considering the properties of a parameter dependent SR1 update of the inverse of the Hessian matrix. Our quasi-Newton update $M_{k+1}$, which approximates $[\nabla^2 f(x)]^{-1}$ in some senses, draws its inspiration from the *weak secant equation* (see e.g. Dennis-Wolkowicz [41]). However, reasoning as in Sections 3.3.1-3.3.2, after some computation we soon realize that it might be difficult to iteratively assess a positive definite preconditioner, which satisfies the weak secant equations at the previous iterates. Thus, we can consider the following *modified weak secant equations*

$$y_j^T M_{k+1} y_j = \delta_j y_j^T s_j, \qquad \text{for all} \qquad j \leq k. \tag{3.28}$$

Observe that the latter appealing properties (as well as the secant equations (3.8)) are again satisfied by all the updates of the Broyden class, for any $j \geq 1$, provided that the linesearch adopted is exact (see e.g. [95]). We would like to recover the motivation underlying the latter class of updates, and by using rank-1 updates we would like to define a preconditioner for PNCG.

On this guideline, in order to build an approximate inverse of the Hessian matrix, drawing inspiration from (3.9), we consider the update

$$M_{k+1}(\gamma_{k+1}) = \delta_k M_k(\gamma_k) + \Delta_k, \qquad \Delta_k \in \mathbb{R}^{n\times n}, \text{ symmetric, } \delta_k > 0, \tag{3.29}$$

where the sequence $\{M(\gamma_k)\}$ depends on the parameter $\gamma_k$ and provides our quasi-Newton updates of $[\nabla^2 f(x)]^{-1}$.

Assuming that $M_k = M(\gamma_k)$ is given, we consider the relation (3.29) where now we set

$$\Delta_k = \gamma_k^{(1)} \frac{p_k p_k^T}{y_k^T p_k}, \qquad \gamma_k^{(1)} \in \mathbb{R} \setminus \{0\},$$

and $p_k$ is generated at the $k-$th iteration of the (unpreconditioned) NCG method. Thus, we will have the new update

$$M_{k+1} = \delta_k M_k + \gamma_k^{(1)} \frac{p_k p_k^T}{y_k^T p_k}, \qquad \gamma_k^{(1)} \in \mathbb{R} \setminus \{0\}, \ \delta_k > 0, \tag{3.30}$$

and in order to satisfy the modified weak secant equation $y_k^T M_{k+1} y_k = \delta_k y_k^T s_k$ the following equality must hold

$$y_k^T \left( \delta_k M_k y_k + \gamma_k^{(1)} \frac{p_k p_k^T}{y_k^T p_k} \right) y_k = \delta_k y_k^T s_k,$$

that is

$$\delta_k y_k^T M_k y_k + \gamma_k^{(1)} y_k^T p_k = \delta_k y_k^T s_k. \tag{3.31}$$

The latter relation is satisfied by taking

$$\gamma_k^{(1)} = \frac{\delta_k (y_k^T s_k - y_k^T M_k y_k)}{y_k^T p_k} \tag{3.32}$$

for any $\delta_k > 0$.

Now, in the next proposition we first consider the case of quadratic functions, and prove that the update (3.30) satisfies the modified weak secant equation, along all previous directions.

**Proposition 3.3.6.** *Assume that $f$ is the quadratic function $f(x) = \frac{1}{2}x^T Ax + b^T x$, where $A \in \mathbb{R}^{n\times n}$ is symmetric and $b \in \mathbb{R}^n$. Suppose that $k$ steps of the (unpreconditioned) CG are performed, in order to detect the stationary point (if any) of the function $f$, and that the vectors $p_1, \ldots, p_k$ are generated. Then, the matrix $M_{k+1}$ in* (3.30) *satisfies the secant equations*

$$y_j^T M_{k+1} y_j = \delta_j s_j^T y_j, \qquad j = 1, \ldots, k, \tag{3.33}$$

*provided that the nonzero coefficients $\gamma_j^{(1)}$, $j = 1, \ldots, k$ are computed, for any $\delta_j > 0$, such that*

$$\gamma_j^{(1)} = \frac{\delta_j (y_j^T s_j - y_j^T M_j y_j)}{y_j^T p_j}, \quad j = 1, \ldots, k. \tag{3.34}$$

*Proof.* The proof proceeds by induction. Equations (3.33) hold for $k = 1$, that is $y_1^T M_2 y_1 = \delta_1 y_1^T s_1$, as long as

$$\delta_1 y_1^T s_1 = y_1^T \left( \delta_1 M_1 + \gamma_1^{(1)} \frac{p_1 p_1^T}{y_1^T p_1} \right) y_1,$$

or equivalently

$$\delta_1 y_1^T s_1 = \delta_1 y_1^T M_1 y_1 + \gamma_1^{(1)} y_1^T p_1,$$

which is satisfied selecting $\gamma_1^{(1)}$ and $\delta_1$ according with (3.34).
Now, suppose that the relations (3.33) hold for the index $k-1$. To complete the induction we need to prove that the relations (3.33) hold for the index $k$. Firstly, note that $y_k^T M_{k+1} y_k = \delta_k y_k^T s_k$ holds. In fact

$$\delta_k y_k^T s_k = y_k^T \left( \delta_k M_k + \gamma_k^{(1)} \frac{p_k p_k^T}{y_k^T p_k} \right) y_k$$

holds if and only if

$$\delta_k y_k^T s_k = \delta_k y_k^T M_k y_k + \gamma_k^{(1)} y_k^T p_k,$$

and the latter holds from (3.34) with $j = k$. Now, we have to prove that (3.33) hold for any $j < k$. For $j < k$ we have

$$y_j^T M_{k+1} y_j = \delta_j y_j^T M_k y_j + y_j^T \left( \gamma_k^{(1)} \frac{p_k p_k^T}{y_k^T p_k} \right) y_j,$$

where $\delta_j y_j^T M_k y_j = \delta_j y_j^T s_j$ by the inductive hypothesis. Finally,

$$y_j^T (\gamma_k^{(1)} p_k p_k^T) y_j = \gamma_k^{(1)} (A s_j)^T (p_k p_k^T) A s_j = \gamma_k^{(1)} \alpha_j^2 p_j^T A (p_k p_k^T) A p_j = 0,$$

which follows from the conjugacy of the directions $\{p_1, \ldots, p_k\}$ generated by the CG. Thus, (3.33) hold for any $j \leq k$ and the induction is complete. □

From (3.34) we know that, for any $\delta_k > 0$, we must have $\gamma_k^{(1)} = \dfrac{\delta_k (y_k^T s_k - y_k^T M_k y_k)}{y_k^T p_k}$.
From a numerical point of view, $y_k^T s_k - y_k^T M_k y_k$ could be negative; in this case we

have the matrix $\Delta_k$ negative definite. In the next proposition we prove that if the parameter $y_k^T s_k - y_k^T M_k y_k$ is negative, we can set the parameter $\delta_k$ such that the matrix $M_{k+1}$ is positive definite. From (3.30) $M_{k+1}$ can be rewritten in the form

$$M_{k+1} = \delta_k M_k + \Delta_k, \tag{3.35}$$

where

$$\Delta_k = \gamma_k^{(1)} \frac{p_k p_k^T}{y_k^T p_k}. \tag{3.36}$$

So, the matrix (3.36) is negative semidefinite if and only if the scalar $\dfrac{\gamma_k^{(1)}}{y_k^T p_k}$ is negative; that is $\gamma_k^{(1)} < 0$ (by definition $y_k^T p_k$ is positive). Since $y_k^T p_k > 0$ and $\delta_k > 0$, $\gamma_k^{(1)}$ is negative if and only if $y_k^T s_k - y_k^T M_k y_k < 0$, with $y_k^T s_k > 0$ and $y_k^T M_k y_k > 0$.

**Proposition 3.3.7.** *Let $f$ be a nonlinear twice continuously differentiable function. Suppose that the (unpreconditioned) NCG method is used to minimize the function $f$. Suppose that* (3.34) *is satisfied, there exits a sequence of positive values $\{\hat{\lambda}_k\}$ such that if $M_1 \succ 0$ with*

$$\lambda_m(M_1) \geq \hat{\lambda}_1 > 0,$$

*and the sequence $\{\delta_k\}$ in* (3.35) *satisfies*

$$\begin{cases} \delta_k > 0 & \text{if } \lambda_m(\Delta_k) \geq 0, \\ \delta_k \geq \dfrac{\hat{\lambda}_{k+1} - \lambda_m(\Delta_k)}{\lambda_m(H_k)} & \text{if } \lambda_m(\Delta_k) < 0, \end{cases} \tag{3.37}$$

*the matrix $M_{k+1}$ in* (3.30) *is positive definite, for any $k \geq 1$, with*

$$\lambda_m(M_{k+1}) \geq \hat{\lambda}_{k+1}.$$

*Proof.* We prove the result by complete induction. Since by hypothesis $\lambda_m(M_1) \geq \hat{\lambda}_1 > 0$, then a sufficient condition to have $\lambda_m(M_2) \geq \hat{\lambda}_2 > 0$ is that $\omega_1 \geq 0$, so that $\lambda_m(M_2) \geq \delta_1 \hat{\lambda}_1 = \hat{\lambda}_2$ for any $\delta_1 > 0$. On the other hand, if $\omega_1 < 0$ then we can set $\hat{\lambda}_2$ such that

$$\lambda_m(M_2) \stackrel{\text{def}}{=} \lambda_m(\delta_1 M_1 + \Delta_1) \geq \delta_1 \lambda_m(M_1) + \lambda_m(\Delta_1) \geq \hat{\lambda}_2 > 0,$$

which is satisfied provided that

$$\delta_1 \geq \frac{\hat{\lambda}_2 - \lambda_m(\Delta_1)}{\lambda_m(M_1)}.$$

Now assume that by induction $\lambda_m(M_k) \geq \hat{\lambda}_k$, we want to prove the result for step $k+1$. Then, using

$$\lambda_m(\Delta_k) = \min\left\{ \gamma_k^{(1)} \frac{\|p_k\|^2}{y_k^T p_k}, 0 \right\}$$

and considering the cases $\gamma_k^{(1)} \leq 0$ and $\gamma_k^{(1)} > 0$, the choice (3.37) immediately yields $\lambda_m(M_{k+1}) \geq \hat{\lambda}_{k+1}$. □

From the previous Proposition 3.3.7, we ensure that the matrix $M_{k+1}$ is positive definite. This fills the gap with preconditioners in Sections 3.3.1-3.3.2, where conditions (3.15), (3.16), (3.26) and (3.23) did not suffice to ensure $M_{k+1}$ positive definite.

### 3.3.4 A preconditioner using a BFGS-like quasi-Newton update

In the previous section we have introduced a positive definite preconditioner: both in case of $\gamma_k^{(1)} \leq 0$ and $\gamma_k^{(1)} > 0$, where $\gamma_k^{(1)}$ is given by (3.32), Proposition 3.3.7 ensures that the matrix $M_{k+1}$ is positive definite. However, the main drawback is that conditions (3.37) may be difficult to guarantee. On this guideline, in this section we introduce a new class of preconditioners which are iteratively constructed by using information from NCG iterations, and satisfy the properties *(1)-(5)* in Section 3.3. On this purpose, in order to comply with properties *(4)* and *(5)*, the preconditioners in our proposal satisfy two prerequisites. First they are conceived around the rightmost term (3.6) in (3.5), in order to possibly approximate the inverse Hessian matrix; then, they satisfy the secant equation at the current iterate, and not necessarily at all the previous iterates. This is a weak theoretical requirement, with respect to other quasi-Newton updates, however numerical results yet confirm its efficiency and robustness.

Now, in order to introduce a class of preconditioners for the NCG, suppose we have performed $k$ iterations of the (unpreconditioned) NCG, so that the directions $p_1, \ldots, p_k$ are generated. To this aim, assuming that $M_k = M(\gamma_k)$ is given, we consider the relation (3.9) where now we set

$$\begin{aligned} M_k &= \gamma_k^{(1)} C_k \\ \Delta_k &= \gamma_k^{(2)} v_k v_k^T + \gamma_k^{(3)} \sum_{j=k-m}^{k} \alpha_j \frac{p_j p_j^T}{p_j^T \nabla^2 f(x_j) p_j} \end{aligned}$$

with $0 \leq m \leq k-1$, $\gamma_k^{(1)} > 0$, $\gamma_k^{(2)}, \gamma_k^{(3)} \geq 0$, $C_k \in \mathbb{R}^{n \times n}$ is *symmetric positive definite* and $v_k \in \mathbb{R}^n$.

Thus we will have the new update

$$M_{k+1} = \gamma_k^{(1)} C_k + \gamma_k^{(2)} v_k v_k^T + \gamma_k^{(3)} \sum_{j=k-m}^{k} \alpha_j \frac{p_j p_j^T}{p_j^T \nabla^2 f(x_j) p_j}, \tag{3.38}$$

where $0 \leq m \leq k-1$, $\gamma_k^{(1)} > 0$, $\gamma_k^{(2)}, \gamma_k^{(3)} \geq 0$, $C_k \in \mathbb{R}^{n \times n}$ is *symmetric positive definite* and $v_k \in \mathbb{R}^n$.

Observe that in the expression of $M_{k+1}$ (see (3.38)), $\gamma_k^{(2)} v_k v_k^T$ represents a rank-1 matrix while in view of (3.5)-(3.7), the term

$$\sum_{j=k-m}^{k} \frac{p_j p_j^T}{p_j^T \nabla^2 f(x_j) p_j} \tag{3.39}$$

is aimed at building, in some sense, an approximate inverse of the Hessian matrix on a specific subspace. The next proposition better justifies the last statement.

**Proposition 3.3.8.** *Let $f(x) = 1/2x^T Ax + b^T x$, with $A \succ 0$. Let $p_1, \dots, p_n \in \mathbb{R}^n \setminus \{0\}$, with $p_i^T A p_j = 0$, $1 \le i \ne j \le n$. Then, for any $0 \le m \le \min\{n-1, k-1\}$,*

$$\left[ \sum_{j=k-m}^{k} \frac{p_j p_j^T}{p_j^T \nabla^2 f(x_j) p_j} \right] Av = v, \qquad \text{for all} \quad v \in span\{p_{k-m}, \dots, p_k\}.$$

*Moreover, when $m = n-1$ then* $\displaystyle\sum_{j=k-m}^{k} \frac{p_j p_j^T}{p_j^T \nabla^2 f(x_j) p_j} = A^{-1}$.

*Proof.* Let $v = \sum_{i=k-m}^{k} \mu_i p_i$, $\mu_i \in \mathbb{R}$; then, since $\nabla^2 f(x) = A$, for any $x \in \mathbb{R}^n$, we have

$$\begin{aligned} \left[ \sum_{j=k-m}^{k} \frac{p_j p_j^T}{p_j^T \nabla^2 f(x_j) p_j} \right] Av &= \left[ \sum_{j=k-m}^{k} \frac{p_j p_j^T}{p_j^T A p_j} \right] Av \\ &= \sum_{j=k-m}^{k} \sum_{i=k-m}^{k} \mu_i \frac{p_j p_j^T}{p_j^T A p_j} A p_i = \sum_{i=k-m}^{k} \mu_i p_i = v. \end{aligned}$$

In case $m = n-1$, since the vectors $\{p_j\}$ are also linearly independent, we directly obtain the inverse matrix $A^{-1}$. □

Thus, in case $f(x)$ is quadratic, then (3.39) behaves as an inverse of the Hessian matrix on the subspace spanned by the linearly independent vectors $p_{k-m}, \dots, p_k$. The integer $m$ can be viewed as a "limited memory" parameter, similarly to the L-BFGS method. Moreover, we can set the matrix $C_k$, the vector $v_k$ and the parameters $\gamma_k^{(1)}, \gamma_k^{(2)}, \gamma_k^{(3)}$ such that the class of preconditioners $\{M_k\}$ satisfies, for any $k$, the secant equation at the current iterate

$$M_{k+1} y_k = s_k. \tag{3.40}$$

along with a *modified secant equation* at some previous iterates, as described in the next proposition.

**Proposition 3.3.9.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable. Suppose that $k$ iterations of NCG are performed, using a strong Wolfe linesearch procedure. Let $M_{k+1} \in \mathbb{R}^{n \times n}$ be defined as in* (3.38)*, with $0 \le m \le k-1$, $\gamma_k^{(1)} > 0$, $\gamma_k^{(2)}, \gamma_k^{(3)} \ge 0$.*

*(i) Let $C_k \in \mathbb{R}^{n \times n}$ be symmetric positive definite, then there exist values of $\gamma_k^{(1)}, \gamma_k^{(2)}, \gamma_k^{(3)}$ such that $M_{k+1} \succ 0$ and* (3.40) *holds.*

*(ii) Let $C_k \in \mathbb{R}^{n \times n}$ be symmetric positive definite and $f(x) = 1/2x^T Ax + b^T x$. Then, $M_{k+1} \succ 0$,* (3.40) *holds and $M_{k+1}$ reduces to*

$$M_{k+1} = \gamma_k^{(1)} C_k + \gamma_k^{(2)} v_k v_k^T + \gamma_k^{(3)} \sum_{j=k-m}^{k} \frac{s_j s_j^T}{y_j^T s_j}, \tag{3.41}$$

*with $v_k = \sigma_k(s_k - \gamma_k^{(1)} C_k y_k - \gamma_k^{(3)} s_k)$, $\sigma_k \in \{-1, +1\}$.*

*(iii) Let $f$ be the quadratic function $f(x) = 1/2x^T Ax + b^T x$, with $A \succ 0$ and suppose $k \geq 2$ iterations of the NCG algorithm are performed, using an exact linesearch. Then, there exist values of $\gamma_k^{(1)}, \gamma_k^{(2)}, \gamma_k^{(3)}$, and a positive semidefinite matrix $C_k$, such that $M_{k+1} \succ 0$, (3.40) holds and the following modified secant conditions*

$$M_{k+1} y_i = \gamma_k^{(3)} s_i, \qquad i = k-m, \ldots, k-1, \tag{3.42}$$

*are satisfied.*

*Proof.* From (3.38) imposing relation (3.40) we have

$$\gamma_k^{(1)} C_k y_k + \gamma_k^{(2)} (v_k^T y_k) v_k + \gamma_k^{(3)} \sum_{j=k-m}^{k} \alpha_j \frac{p_j^T y_k}{p_j^T \nabla^2 f(x_j) p_j} p_j = s_k;$$

hence, assuming $\gamma_k^{(2)}(v_k^T y_k) \neq 0$ (which may be straightforwardly guaranteed by a suitable choice of $\gamma_k^{(1)}, \gamma_k^{(2)}, \gamma_k^{(3)}$),

$$v_k = \sigma_k \left[ s_k - \gamma_k^{(1)} C_k y_k - \gamma_k^{(3)} \sum_{j=k-m}^{k} \alpha_j \frac{p_j^T y_k}{p_j^T \nabla^2 f(x_j) p_j} p_j \right], \tag{3.43}$$

for some $\sigma_k \in \mathbb{R}$. Replacing (3.43) in (3.40) we obtain the equation

$$\begin{aligned} \gamma_k^{(2)} \sigma_k^2 \left[ s_k^T y_k - \gamma_k^{(1)} y_k^T C_k y_k - \gamma_k^{(3)} \sum_{j=k-m}^{k} \frac{(p_j^T y_k)^2}{p_j^T \nabla^2 f(x_j) p_j} \right] \\ \cdot \left[ s_k - \gamma_k^{(1)} C_k y_k - \gamma_k^{(3)} \sum_{j=k-m}^{k} \frac{p_j^T y_k}{p_j^T \nabla^2 f(x_j) p_j} p_j \right] = \\ s_k - \gamma_k^{(1)} C_k y_k - \gamma_k^{(3)} \sum_{j=k-m}^{k} \frac{p_j^T y_k}{p_j^T \nabla^2 f(x_j) p_j} p_j. \end{aligned}$$

Thus, the following relation among the parameters $\sigma_k, \gamma_k^{(1)}, \gamma_k^{(2)}, \gamma_k^{(3)}$ has to be satisfied

$$\gamma_k^{(2)} \sigma_k^2 = \frac{1}{s_k^T y_k - \gamma_k^{(1)} y_k^T C_k y_k - \gamma_k^{(3)} \sum_{j=k-m}^{k} \frac{(p_j^T y_k)^2}{p_j^T \nabla^2 f(x_j) p_j}}, \tag{3.44}$$

where, without loss of generality, we can set $\sigma_k \in \{-1, +1\}$. Then, we remark that the condition (3.44) guarantees the matrix $M_{k+1}$ in (3.40) to satisfy the secant equation only at the $k$-th iteration (even for quadratic functions), and possibly not at the previous iterates. To complete the proof of item *(i)*, observe that the Wolfe conditions used in the linesearch procedure for computing the steplength $\alpha_k$ ensure that (2.15) holds, i.e. $s_k^T y_k > 0$. Thus, for $\gamma_k^{(1)} > 0$ and $\gamma_k^{(3)} \geq 0$ sufficiently small in (3.44) we obtain that $\gamma_k^{(2)} > 0$, and the matrix $M_{k+1}$ is positive definite. To prove item *(ii)*, by the Mean Value Theorem we have

$$\int_0^1 s_j^T \nabla^2 f[x_j + \zeta(x_{j+1} - x_j)] s_j \, d\zeta = s_j^T y_j,$$

and using relation $s_j = \alpha_j p_j$, in case $f(x)$ is a quadratic function, then we have

$$p_j^T A p_j = p_j^T \nabla^2 f(x_j) p_j = \int_0^1 p_j^T \nabla^2 f[x_j + \zeta(x_{j+1} - x_j)] p_j \, d\zeta = \frac{p_j^T y_j}{\alpha_j}, \qquad (3.45)$$

which can be replaced in (3.38) to obtain (3.41). Since the Wolfe conditions are used in the linesearch procedure, then (2.15) holds, still implying that

$$\sum_{j=k-m}^{k} \frac{s_j s_j^T}{y_j^T s_j} \succeq 0.$$

In addition, since $y_k = A s_k$, now the expression of $v_k$ in (3.43) reduces to $v_k = \sigma_k(s_k - \gamma_k^{(1)} C_k y_k - \gamma_k^{(3)} s_k)$.

Finally, as regards *(iii)*, let us define

$$C_k = V_k^T V_{k-1}^T \cdots V_{k-m}^T V_{k-m} \cdots V_{k-1} V_k. \qquad (3.46)$$

Even if $C_k$ now is not positive definite, similarly to the proof of *(i)*, we can obtain (3.43) and (3.44). Now, since $y_k = A s_k$, we have $M_{k+1} y_k = s_k$, $v_k = \sigma_k(s_k - \gamma_k^{(1)} C_k y_k - \gamma_k^{(3)} s_k)$ and

$$\gamma_k^{(2)} \sigma_k^2 = \frac{1}{(1 - \gamma_k^{(3)}) s_k^T y_k - \gamma_k^{(1)} y_k^T C_k y_k}. \qquad (3.47)$$

We prove that the matrix $M_{k+1}$ (which is now the sum of positive semidefinite matrices) is positive definite. Indeed, let $s_1, \ldots, s_n$ be $n$ conjugate (hence linearly independent) directions with respect to matrix $A \succ 0$. Then, recalling that the exact linesearch along with the conjugacy among $\{s_j\}$ yield $\nabla f(x_{j+1}) = \nabla f(x_j) + A s_j$ and

$$(A s_i)^T (A s_j) = 0, \qquad \text{for all} \quad |i - j| > 1, \qquad (3.48)$$

by (3.3) and for any $\gamma_k^{(1)} \neq 0$, $\gamma_k^{(3)} \neq 0$ it results

$$\left[ \gamma_k^{(1)} C_k + \gamma_k^{(3)} \sum_{j=k-m}^{k} \frac{s_j s_j^T}{y_j^T s_j} \right] A s_i \neq 0, \qquad i = 1, \ldots, n.$$

Indeed, the latter result trivially holds for any $i \neq k-m-1$; moreover, for $i = k-m-1$ it also holds, using the relation $V_{k-m}^T (A s_{k-m-1}) = A s_{k-m-1} \neq 0$. This implies that the matrix $\gamma_k^{(1)} C_k + \gamma_k^{(3)} \sum_{j=k-m}^{k} s_j s_j^T / y_j^T s_j$ (and consequently $M_{k+1}$) is nonsingular. Moreover, since $f(x)$ is quadratic, by (3.41) we obtain for $i \in \{k-m, \ldots, k\}$

$$\begin{aligned} M_{k+1} y_i &= \left[ \gamma_k^{(1)} C_k + \gamma_k^{(2)} v_k v_k^T + \gamma_k^{(3)} \sum_{j=k-m}^{k} \frac{s_j s_j^T}{y_j^T s_j} \right] y_i \\ &= \left[ \gamma_k^{(1)} C_k + \gamma_k^{(2)} v_k v_k^T \right] A s_i + \gamma_k^{(3)} s_i. \end{aligned}$$

Now, since $v_k = \sigma_k(s_k - \gamma_k^{(1)} C_k y_k - \gamma_k^{(3)} s_k)$, then we obtain for $i \in \{k-m, \ldots, k-1\}$ that $v_k^T A s_i = 0$. Furthermore, by a direct computation we also have for $i \in \{k-m, \ldots, k-1\}$

$$C_k A s_i = V_k^T V_{k-1}^T \cdots V_{k-m}^T V_{k-m} \cdots V_{k-1} V_k A s_i = 0;$$

thus, we finally obtain

$$M_{k+1}y_i = \gamma_k^{(1)} C_k A s_i + \gamma_k^{(3)} s_i = \gamma_k^{(3)} s_i, \qquad i \in \{k-m, \dots, k-1\}.$$

□

In the next proposition we give some properties about the clustering of the eigenvalues of the preconditioner $M_{k+1}$.

**Proposition 3.3.10.** *Let $f(x) = 1/2 x^T A x + b^T x$, with $A \succ 0$, and suppose $k \geq 2$ iterations of the NCG algorithm are performed, using an exact linesearch. Consider the matrix $C_k$ in* (3.46) *and $M_{k+1}$ in* (3.41)*. Then, $M_{k+1}$ has at least $n-(m+2)$ eigenvalues equal to $\gamma_k^{(1)}$.*

*Proof.* After some computations, we obtain the relation $V_{k-m}^T(As_{k-m-1}) = As_{k-m-1}$, and by the hypotheses (see also (3.43)), it results $v_k = \sigma_k(s_k - \gamma_k^{(1)} C_k y_k - \gamma_k^{(3)} s_k)$. Then, recalling (3.48), we have

$$M_{k+1}As_i = \gamma_k^{(1)} As_i, \qquad \text{for } \; i \leq k-m-1 \;\text{ and }\; k+2 \leq i \leq n,$$

so that $[k-m-1] + [n-(k+2)+1] = n-(m+2)$ eigenvalues of $M_{k+1}$ are equal to $\gamma_k^{(1)}$. □

Observe that the different choices for the parameters $\gamma_k^{(1)}$ and $\gamma_k^{(3)}$ in (3.44) provide a different scaling of the matrices $C_k$ and

$$\sum_{j=k-m}^{k} \frac{p_j p_j^T}{p_j^T \nabla^2 f(x_j) p_j}$$

in the preconditioners.

As regards the specific choice of $\gamma_k^{(3)}$, $\gamma_k^{(1)}$ and $C_k$ in (3.41), observe that by (3.42), the choice $\gamma_k^{(3)} = 1$ and $C_k$ given by (3.46) seems appealing when $f(x)$ is quadratic. However, with $\gamma_k^{(3)} = 1$ in (3.47) $\gamma_k^{(2)}$ might not be well defined or possibly negative. Also observe that

$$rk(C_k) = rk\left[V_k^T V_{k-1}^T \cdots V_{k-m}^T V_{k-m} \cdots V_{k-1} V_k\right] \leq n-1,$$

so that $C_k$ is consequently singular, and when $f(x)$ is non-quadratic the preconditioner $M_{k+1}$ might be singular. To avoid the latter drawback, and possibly reduce the computational burden, while preserving a certain level of efficiency, an obvious choice could be $\gamma_k^{(3)} \neq 1$ and

$$C_k = \varepsilon_k I, \qquad \varepsilon_k \in \mathbb{R}.$$

The parameter $\varepsilon_k$ may be computed as the least squares solution of the equation $(\varepsilon I)y_k - s_k = 0$, i.e. $\varepsilon_k$ solves

$$\min_{\varepsilon} \|(\varepsilon I)y_k - s_k\|^2.$$

Hence,

$$\varepsilon_k = \frac{s_k^T y_k}{\|y_k\|^2}$$

so that since $s_k^T y_k > 0$ by the Wolfe conditions, the matrix

$$C_k = \frac{s_k^T y_k}{\|y_k\|^2} I \tag{3.49}$$

is positive definite. It is not difficult to verify that the choice (3.49), for $C_k$, also satisfies the weak secant equation $y_k^T C_k y_k = y_k^T s_k$ (see [41]), at current iterate $x_k$.

For the sake of clarity we report here the overall resulting expression of our class of preconditioners (3.38), including the choice (3.49) and $\sigma_k = 1$:

$$M_{k+1} = \gamma_k^{(1)} \frac{s_k^T y_k}{\|y_k\|^2} I + \gamma_k^{(2)} v_k v_k^T + \gamma_k^{(3)} \sum_{j=k-m}^{k} \frac{s_j s_j^T}{y_j^T s_j}, \tag{3.50}$$

where

$$v_k = s_k - \gamma_k^{(1)} \frac{s_k^T y_k}{\|y_k\|^2} y_k - \gamma_k^{(3)} \sum_{j=k-m}^{k} \frac{s_j^T y_k}{y_j^T s_j} s_j, \tag{3.51}$$

$$\gamma_k^{(2)} = \frac{1}{(1-\gamma_k^{(1)}) s_k^T y_k - \gamma_k^{(3)} \sum_{j=k-m}^{k} \frac{(s_j^T y_k)^2}{y_j^T s_j}}. \tag{3.52}$$

The reader may conjecture that since $M_{k+1}$ merely satisfies, in the convex quadratic case, the interpolation (say secant) conditions (3.40) and (3.42), then its theoretical properties with respect to BFGS are definitely poor. This seems indeed a partially correct conclusion. However, since in practice L-BFGS often performs better than BFGS, we warn the reader that on nonconvex problems the good performance of our proposal in Section 3.3.4.1 might not be so surprising. In fact, likewise L-BFGS we retain information from a limited number of previous iterates, mainly relying on the role of the rightmost term in (3.50), as detailed in Proposition 3.3.8.

We conclude this section by highlighting that, interestingly enough, we can also construct a class of preconditioners based on DFP-like quasi-Newton updates. Indeed, we can iteratively build the matrices $B(\gamma_{k+1})$, approximating $\nabla^2 f(x)$ instead of its inverse. Then, by the Sherman-Morrison-Woodbury formula applied to $B(\gamma_{k+1})$ we can compute a class of preconditioners alternative to $M(\gamma_{k+1})$. However, following the current literature which privileges the use of BFGS in place of DFP [95], here we have proposed the class described in (3.50)-(3.52), which performed successfully in practice.

#### 3.3.4.1 Numerical experience

In order to investigate the reliability of the class of preconditioners we have introduced, we performed a wide numerical testing using the preconditioners defined in (3.50).

To this purpose, we embedded the preconditioners (3.50) within the standard CG+ code (see [51]), from the literature, available at J. Nocedal's web page. As regards the stopping criterion we adopt the standard one given by (see e.g. [78], [83], [93])

$$\|g_k\| \leq 10^{-5} \max\{1, \|x_k\|\}.$$

For a fair comparison we used the same linesearch used by default in CG+ code. It is the Moré-Thuente linesearch [84] with a slight modification (we refer the reader to [51] for a complete description of the algorithm). In particular, we tested the standard Polak and Ribière (PR) version of the PNCG method (see [97]). As regards the test problems, we selected all the large scale unconstrained test problems in the CUTEst collection [56]. The dimension of the test problems is between $n = 1000$ and $n = 10000$ (we considered 112 resulting problems). The parameters of the preconditioners (3.50) have been chosen as follows:

$$m = 4, \quad \gamma_k^{(1)} = \frac{\frac{1}{2} s_k^T y_k}{y_k^T C_k y_k + \sum_{j=k-m}^{k} \frac{(s_j^T y_k)^2}{s_j^T y_j}}, \quad \gamma_k^{(2)} = \frac{2}{s_k^T y_k}, \quad \gamma_k^{(3)} = \gamma_k^{(1)},$$

where $C_k$ is given by (3.49), for all $k$ (this choice ensures that, by Wolfe conditions, the denominator of $\gamma_k^{(2)}$ in (3.52) is positive). As preliminary investigation, we considered the results in terms of the number of iterations and the number of function evaluations, comparing three alternatives:

- $M_{k+1}$ in (3.50), namely *OUR PREC*;
- $M_{k+1} = I$ (unpreconditioned case), namely *UNPREC*;
- $M_{k+1}$ coincident with the L-BFGS update $H_{k+1}$ in (2.64), using a memory of $m = 4$, namely *PREC-LBFGS*.

The overall comparison is reported by using performance profiles [42]. For a fair comparison, we have excluded from each profile all the test problems where the three alternatives do not converge to the same stationary point. Moreover, for $k < 4$ (i.e. in the first three PNCG iterations) we have coherently set $m = \min\{4, k\}$.

We strongly highlight that our proposal (3.50) is built using a dual standpoint with respect to *PREC-LBFGS*. Indeed, our proposal starts by first considering the third matrix in the right hand side of (3.50), in the light of approximating (in the quadratic case) the inverse of the Hessian matrix, as in (3.7). Then, the other two matrices, on the right hand side of (3.50), make our proposal $M_{k+1}$ nonsingular and consistent with a current interpolation condition at iterate $k$. On the contrary, *PREC-LBFGS* update starts from imposing multiple interpolation conditions at previous iterates (i.e. the secant equations). Then, as by-product it also proves to yield in the quadratic case, after $n$ iterations, the inverse Hessian.

The choice $m = 4$ was in our experience the best compromise over the chosen test set. This should not be surprising if compared with the results in [48, 49, 83], where the best choice for the memory parameter is either $m = 7$ or $m = 8$. In fact, in the latter papers the preconditioner is built using the CG (or L-BFGS for quadratics) in

place of the NCG, which allows to fully exploit the mutual conjugacy among the search directions. On the contrary, in this work the NCG is unable to guarantee the latter property, so that the information at iterations $k-m-1, k-m-2, \dots$ for large $m$ risks to be unreliable.

In Figures 3.1-3.2 we report the comparison among the three algorithms. These profiles show that our proposal definitely outperforms the competitors, both in terms of number of iterations and number of function evaluations.

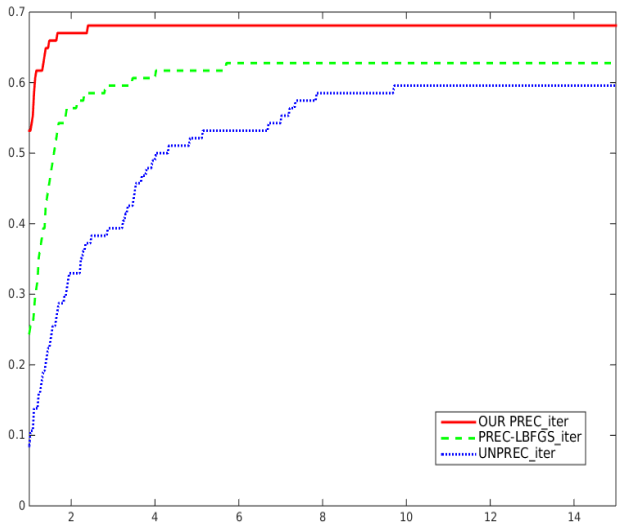


**Figure 3.1.** Comparison among *OUR PREC* (solid line), *PREC-LBFGS* (dashed line) and *UNPREC* (dotted line), in terms of number of *iterations*.

Finally, we guess that in place of (3.49), a more sophisticated choice of the matrix $C_k$ might be conceived, which possibly summarizes more information on the function at the previous iterates.

### 3.3.5 A Symmetric Rank-2 update based on modified secant equations

In the previous section we have introduced new quasi-Newton updates to iteratively construct a preconditioner for PNCG method such that:

- it is positive definite;
- it satisfies the secant equation at the current iteration.

However, since the search direction we compute does not seem yet well scaled (see Section 5 in [32]), in this section we propose a novel quasi-Newton updating formula, by considering the properties of a parameter dependent symmetric rank-2 (SR2) update of the inverse Hessian matrix, used as a possible preconditioner for the NCG (see, for example, preconditioner in Section 3.3.4). We claim that our quasi-Newton

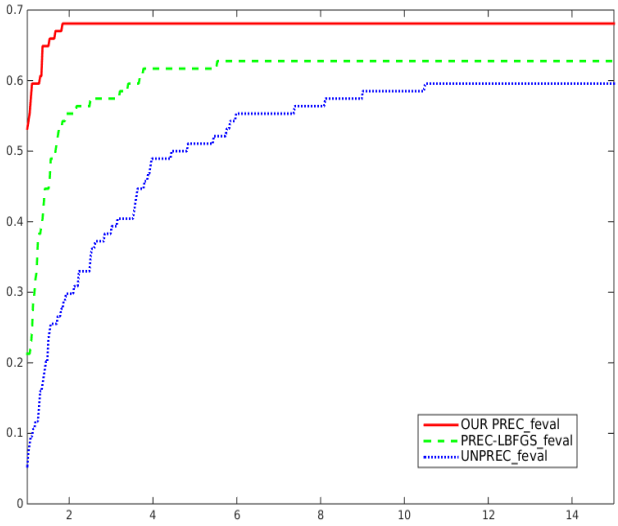


**Figure 3.2.** Comparison among *OUR PREC* (solid line), *PREC-LBFGS* (dashed line) and *UNPREC* (dotted line), in terms of number of *function evaluations*.

update $M_{k+1}$, which aims at approximating $[\nabla^2 f(x)]^{-1}$ in some sense, satisfies the following *modified* secant equation along all previous directions; namely it results

$$M_{k+1}y_j = \delta_j s_j \qquad \text{with} \qquad \begin{cases} \delta_j > 0, & \text{for} \quad j < k, \\ \\ \delta_j = 1, & \text{for} \quad j = k. \end{cases} \tag{3.53}$$

The satisfaction of (3.53) is a distinguishing property of our proposal in this study, and though (3.53) imposes weaker conditions with respect to the satisfaction of the secant equation at any step $j < k$, numerical performance seems to confirm its effectiveness and efficiency.

On this guideline, in order to build an approximate inverse of the Hessian matrix and explicitly indicating the parameters it depends on, in place of (3.9) we consider here the update

$$M_{k+1} = \delta_k M_k + \Delta_k, \qquad \Delta_k \in \mathbb{R}^{n\times n} \text{ symmetric}, \tag{3.54}$$

where the sequence $\{M_k\}$ explicitly depends on the real parameters $\delta_k$, $\gamma_k^{(1)}$, $\gamma_k^{(2)}$. Considering the relation (3.54) we set now more explicitly

$$\Delta_k = \gamma_k^{(1)} v_k v_k^T + \gamma_k^{(2)} \frac{p_k p_k^T}{y_k^T p_k}, \qquad \gamma_k^{(1)}, \gamma_k^{(2)} \in \mathbb{R} \setminus \{0\}, \qquad v_k \in \mathbb{R}^n,$$

where $p_k$ is generated at the $k$-th iteration of the NCG method. Thus, we have the novel update

$$M_{k+1} = \delta_k M_k + \gamma_k^{(1)} v_k v_k^T + \gamma_k^{(2)} \frac{p_k p_k^T}{y_k^T p_k}, \qquad \gamma_k^{(1)}, \gamma_k^{(2)} \in \mathbb{R} \setminus \{0\}, \qquad v_k \in \mathbb{R}^n. \tag{3.55}$$

We immediately remark that the main difference between (3.55) and the proposal in Section 3.3.4 relies on the following fact. In (3.50) the rightmost contribution aims at possibly computing an approximate inverse Hessian matrix. Then, the coefficients $\gamma_k^{(1)}$, $\gamma_k^{(2)}$ and $\gamma_k^{(3)}$ are set so that $M_{k+1}$ fulfills the secant equation. On the contrary, in (3.55) the rightmost term of $M_{k+1}$ plays a role similar to that of the term $\rho_k s_k s_k^T$ in (2.64), and thus it does not contain the contribution from any previous step.

In order to comply with (3.53), $M_{k+1}$ must satisfy the secant equation $M_{k+1}y_k = s_k$, which implies

$$\delta_k M_k y_k + \gamma_k^{(1)}(v_k^T y_k)v_k + \gamma_k^{(2)}\frac{p_k p_k^T}{y_k^T p_k}y_k = s_k,$$

that is

$$\gamma_k^{(1)}(v_k^T y_k)v_k = s_k - \delta_k M_k y_k - \gamma_k^{(2)}p_k. \tag{3.56}$$

Therefore it results explicitly

$$v_k = \sigma_k\left(s_k - \delta_k M_k y_k - \gamma_k^{(3)}p_k\right) \tag{3.57}$$

for some scalar $\sigma_k \in \mathbb{R}$. By replacing the expression (3.57) of $v_k$ in (3.56) we have

$$\gamma_k^{(1)}\sigma_k^2\left[y_k^T(s_k - \delta_k M_k y_k - \gamma_k^{(2)}p_k)\right](s_k - \delta_k M_k y_k - \gamma_k^{(2)}p_k) = s_k - \delta_k M_k y_k - \gamma_k^{(2)}p_k.$$

Thus, the following relation among the parameters $\sigma_k$, $\gamma_k^{(1)}$ and $\gamma_k^{(2)}$ must hold

$$\gamma_k^{(1)}\sigma_k^2 = \frac{1}{s_k^T y_k - \delta_k y_k^T M_k y_k - \gamma_k^{(2)}p_k^T y_k}. \tag{3.58}$$

By the arbitrariness of $\gamma_k^{(1)}$, without loss of generality, we can set $\sigma_k \in \{-1, 1\}$. Now, in the next proposition we first consider the case of quadratic functions, and prove that under mild assumptions the update (3.55) satisfies the modified secant equation (3.53), along all previous directions.

**Proposition 3.3.11.** *Let $f(x) = \frac{1}{2}x^T Ax - b^T x$, where $A \in \mathbb{R}^{n\times n}$ is symmetric and $b \in \mathbb{R}^n$. Suppose that $k$ steps of the CG are performed, in order to detect the stationary point (if any) of the function $f$, and that the vectors $p_1, \ldots, p_k$ are generated. Then, the matrix $M_{k+1}$ in* (3.55) *satisfies the modified secant equations* (3.53)*, that is*

$$M_{k+1}y_j = \delta_j s_j \qquad \textit{with} \qquad \begin{cases} \delta_j > 0, & \textit{for} \quad j < k, \\ \delta_j = 1, & \textit{for} \quad j = k. \end{cases}$$

*provided that the nonzero coefficients $\gamma_j^{(1)}$, $\gamma_j^{(2)}$, $j = 1, \ldots, k$ are chosen such that*

$$\begin{cases} \gamma_j^{(1)} = \dfrac{1}{s_j^T y_j - \delta_j y_j^T M_j y_j - \gamma_j^{(2)}p_j^T y_j}, & j = 1, \ldots, k, \\ \gamma_j^{(2)} \neq \dfrac{s_j^T y_j - \delta_j y_j^T M_j y_j}{p_j^T y_j}, & j = 1, \ldots, k. \end{cases} \tag{3.59}$$

*Proof.* The proof proceeds by induction. Equations (3.53) hold for $k = 1$, that is $M_2 y_1 = s_1$, as long as

$$s_1 = \left[\delta_1 M_1 + \gamma_1^{(1)}\sigma_1^2(s_1 - \delta_1 M_1 y_1 - \gamma_1^{(2)} p_1)(s_1 - \delta_1 M_1 y_1 - \gamma_1^{(2)} p_1)^T + \gamma_1^{(2)} \frac{p_1 p_1^T}{y_1^T p_1}\right] y_1,$$

or equivalently

$$s_1 - \delta_1 M_1 y_1 - \gamma_1^{(2)} p_1 = \gamma_1^{(1)}(s_1^T y_1 - \delta_1 y_1^T M_1 y_1 - \gamma_1^{(2)} p_1^T y_1)\left[s_1 - \delta_1 M_1 y_1 - \gamma_1^{(2)} p_1\right],$$

which is satisfied selecting $\gamma_1^{(1)}$ and $\gamma_1^{(2)}$ according with (3.59).

Now, suppose that the relations (3.53) hold for the index $k-1$. To complete the induction we need to prove that the relations (3.53) hold for the index $k$. Firstly, note that $M_{k+1} y_k = s_k$ holds. In fact, relation

$$\begin{aligned} s_k &= \left[\delta_k M_k + \gamma_k^{(1)}\sigma_k^2(s_k - \delta_k M_k y_k - \gamma_k^{(2)} p_k)(s_k - \delta_k M_k y_k - \gamma_k^{(2)} p_k)^T\right] y_k \\ &+ \left[\gamma_k^{(2)} \frac{p_k p_k^T}{y_k^T p_k}\right] y_k, \end{aligned}$$

holds if and only if

$$s_k - \delta_k M_k y_k - \gamma_k^{(2)} p_k = \gamma_k^{(1)}(s_k^T y_k - \delta_k y_k^T M_k y_k - \gamma_k^{(2)} p_k^T y_k)(s_k - \delta_k M_k y_k - \gamma_k^{(2)} p_k),$$

which is satisfied using (3.59) with $j = k$. Now, we have to prove that (3.53) hold for any $j < k$. For $j < k$ the definition of $M_{k+1}$ yields

$$\begin{aligned} M_{k+1} y_j &= \delta_k M_k y_j + \gamma_k^{(1)}\sigma_k^2(s_k - \delta_k M_k y_k - \gamma_k^{(2)} p_k)(s_k - \delta_k M_k y_k - \gamma_k^{(2)} p_k)^T y_j \\ &+ \gamma_k^{(2)} \frac{p_k^T y_j}{y_k^T p_k} p_k, \end{aligned}$$

where $M_k y_j = \delta_j s_j$, $j = 1, \ldots, k-2$, and $M_k y_{k-1} = s_{k-1}$ by the inductive hypothesis. Moreover,

$$(s_k - \delta_k M_k y_k)^T y_j = s_k^T y_j - \delta_k y_k^T M_k y_j =$$

$$\begin{cases} s_k^T y_j - \delta_k \delta_j y_k^T s_j = s_k^T A s_j - \delta_k \delta_j s_k^T A s_j = 0, & j < k-1, \\ s_k^T y_{k-1} - \delta_k y_k^T s_{k-1} = s_k^T A s_{k-1} - \delta_k (A s_k)^T s_{k-1} = 0, & j = k-1, \end{cases}$$

where the third equality holds since $y_j = A s_j$, for any $j \in \{1, \ldots, k-1\}$. Moreover, the fourth equality holds since $s_j = \alpha_j p_j$, $j = 1, \ldots, k-1$, and $p_j$ are conjugate directions. Finally,

$$\gamma_k^{(2)} p_k^T y_j = \gamma_k^{(2)} p_k^T A s_j = \gamma_k^{(2)} \alpha_j p_k^T A p_j = 0,$$

which again follows from the conjugacy of the directions $\{p_1, \ldots, p_k\}$. Thus, (3.53) hold for any $j \leq k$ and the induction is complete. □

As an immediate consequence of the previous proposition, we give now a finite termination property for quadratic functions. More specifically, we prove that after at most $n$ steps, the recursion (3.55) provides the matrix $M_{n+1}$, which is, in some sense, a *modified inverse* Hessian.

**Corollary 3.3.12.** *Let $f(x) = \frac{1}{2}x^T Ax - b^T x$, where $A \in \mathbb{R}^{n\times n}$ is symmetric nonsingular and $b \in \mathbb{R}^n$. Suppose that $n$ steps of the CG are performed, in order to detect the stationary point of the function $f$, and that the vectors $p_1, \ldots, p_n$ are generated.*

- *If (3.53)-(3.59) hold, we have $M_{n+1}A = (s_1 \cdots s_n)D(s_1 \cdots s_n)^{-1}$, with $D = diag\{\delta_1, \ldots, \delta_{n-1}, 1\}$.*
- *It results*

$$\lambda_m(M_{n+1}A) = \lambda_m(D), \qquad \lambda_M(M_{n+1}A) = \lambda_M(D). \tag{3.60}$$

*Proof.* By Proposition 3.3.11, we have that relations (3.53) hold for $k = n$, i.e.

$$\begin{cases} M_{n+1}y_j = \delta_j s_j, & j = 1, \ldots, n-1, \\ \\ M_{n+1}y_n = s_n. \end{cases}$$

Since $f$ is quadratic then $y_j = As_j$, for any $j$, i.e.

$$\begin{cases} M_{n+1}As_j = \delta_j s_j, & j = 1, \ldots, n-1, \\ \\ M_{n+1}As_n = s_n. \end{cases}$$

Now, since $s_j = \alpha_j p_j$, $j = 1, \ldots, n$, the conjugacy among the vectors $\{p_1, \ldots, p_n\}$ implies that

$$M_{n+1}A(s_1 \cdots s_n) = (s_1 \cdots s_n)D, \tag{3.61}$$

where $D = diag\{\delta_1, \ldots, \delta_{n-1}, 1\}$. By (3.61) we have

$$M_{n+1}A = (s_1 \cdots s_n)D(s_1 \cdots s_n)^{-1}$$

and therefore the eigenvalues of $M_{n+1}A$ coincide with those of $D$. □

We highlight that, whenever $k = n$, Corollary 3.3.12 justifies item *(5)* in Section 3.3. Moreover, later on in this study we show that for $k < n$, the update in (3.55) can be suitably used to provide a preconditioner. The next corollary details further properties of $M_{k+1}$ in (3.55), again when the function $f(x)$ is quadratic, in the light of possibly approximating the inverse Hessian $A^{-1}$.

**Corollary 3.3.13.** *Let $f(x) = \frac{1}{2}x^T Ax - b^T x$, where $A \in \mathbb{R}^{n\times n}$ is symmetric nonsingular and $b \in \mathbb{R}^n$. Suppose that $n$ steps of the CG are performed, in order to detect the stationary point of the function $f$, and that the vectors $p_1, \ldots, p_n$ are generated. If relations (3.53)-(3.59) hold for any $k = 1, \ldots, n$, then we have*

*1)* $\displaystyle\sum_{j=1}^{k} \delta_j s_j = (M_{k+1}A)(x_{k+1} - x_1),$

*2)* $AM_{k+1}\left[\sum_{j=1}^{k}\frac{g_{j+1}-g_j}{\delta_j}\right]=g_{k+1}-g_1$,

*3) in the case* $k=n$ *then*

$$\sum_{j=1}^{n}\left[(s_1\cdots s_n)D(s_1\cdots s_n)^{-1}-\delta_j I\right]s_j=0,$$

*with* $D=diag\{\delta_1,\ldots,\delta_{n-1},1\}$.

*Proof.* By (3.53) and adding with respect to index $j$ we obtain the two relations

$$M_{k+1}\left[g_{k+1}-g_1\right]=\sum_{j=1}^{k}\delta_j s_j \tag{3.62}$$

$$M_{k+1}\left[\sum_{j=1}^{k}\frac{g_{j+1}-g_j}{\delta_j}\right]=x_{k+1}-x_1. \tag{3.63}$$

By (3.62) and recalling that $f(x)$ is quadratic, we have

$$M_{k+1}A(x_{k+1}-x_1)=\sum_{j=1}^{k}\delta_j s_j, \tag{3.64}$$

which yields *1)*. By (3.63) we immediately infer *2)*. Finally, by Corollary 3.3.12

$$M_{n+1}A=(s_1\cdots s_n)D(s_1\cdots s_n)^{-1}$$

and using (3.64) with $k=n$ we have

$$(s_1\cdots s_n)D(s_1\cdots s_n)^{-1}\sum_{j=1}^{n}s_j=\sum_{j=1}^{n}\delta_j s_j,$$

i.e.

$$\sum_{j=1}^{n}\left[(s_1\cdots s_n)D(s_1\cdots s_n)^{-1}-\delta_j I\right]s_j=0,$$

which is relation *3)*. □

After analyzing the case of $f(x)$ quadratic, we turn now to the general case of a nonlinear continuously differentiable function. In particular, since we are interested in using the matrix $M_{k+1}$ in (3.55) as a preconditioner, we need to investigate if there exists a suitable setting of the parameters $\delta_k$, $\gamma_k^{(1)}$ and $\gamma_k^{(2)}$ such that $M_{k+1}$ is positive definite, provided that (3.53)-(3.59) are satisfied.

In the next lemma we provide a technical result which will be used to prove the latter purpose.

**Lemma 3.3.14.** *Let* $u,v\in\mathbb{R}^n$ *and* $a,b,c\in\mathbb{R}$*, with* $u,v$ *linearly independent and* $ac-b^2\neq 0$*. Then, the symmetric matrix* $\mathcal{H}\in\mathbb{R}^{n\times n}$ *given by*

$$\mathcal{H}=(v\vdots u)\begin{pmatrix}a & b\\ b & c\end{pmatrix}(v\vdots u)^T$$

*has $n-2$ eigenvalues equal to 0, and the two real eigenvalues*

$$\lambda_{n-1} = \frac{\delta+\beta-\sqrt{(\delta-\beta)^2+4\alpha\gamma}}{2}, \qquad \lambda_n = \frac{\delta+\beta+\sqrt{(\delta-\beta)^2+4\alpha\gamma}}{2}, \tag{3.65}$$

*where*

$$\alpha = av^Tu + b\|u\|^2, \quad \beta = a\|v\|^2 + bv^Tu, \quad \gamma = b\|v\|^2 + cv^Tu, \quad \delta = bv^Tu + c\|u\|^2.$$

*Proof.* First note that since $\mathcal{H}$ is symmetric then its $n$ eigenvalues are real. Now, let $\{z_i\}$, $i = 1, \ldots, n$, be an independent set of $n$-real vectors. Let $\{z_i\}$, $i = 1, \ldots, n-2$, be orthogonal to vectors $v, u$. Then, for any $i \in \{1, \ldots, n-2\}$ the vector $z_i$ is trivially an eigenvector of $\mathcal{H}$, associated with the zero eigenvalue. Thus, the only two eigenvectors $z_{n-1}$ and $z_n$ of $\mathcal{H}$, associated with the nonzero eigenvalues $\lambda_{n-1}$ and $\lambda_n$, must satisfy the relation

$$z_{n-1}, z_n \in span\{v, u\}.$$

Now, using

$$\begin{cases} z_{n-1} = \mu_{1,1}v + \mu_{1,2}u \\ \\ z_n = \mu_{2,1}v + \mu_{2,2}u, \end{cases}$$

with $\mu_{1,1}, \mu_{1,2}, \mu_{2,1}, \mu_{2,2} \in \mathbb{R}$, and imposing the conditions $\mathcal{H}z_h = \lambda_h z_h$, for $h \in \{n-1, n\}$, we obtain the couple of relations

$$\left(avv^T + buv^T + bvu^T + cuu^T\right)(\mu_{1,1}v + \mu_{1,2}u) = \lambda_{n-1}(\mu_{1,1}v + \mu_{1,2}u)$$

$$\left(avv^T + buv^T + bvu^T + cuu^T\right)(\mu_{2,1}v + \mu_{2,2}u) = \lambda_n(\mu_{2,1}v + \mu_{2,2}u).$$

Note that after an easy computation, $av^Tu + b\|u\|^2 \neq 0$ implies that $\mu_{1,1} \neq 0$ (indeed, $\mu_{1,1} = 0$ yields $z_{n-1} = \mu_{1,2}u$, but there is no real value for $\lambda$ such that $\mathcal{H}\mu_{1,2}u = \lambda\mu_{1,2}u$). Thus, since the latter relations must hold for any choice of vectors $v$ and $u$, setting $\sigma_{n-1} = \mu_{1,2}/\mu_{1,1}$ in the first of them (a similar analysis holds also for the second relation), we obtain the couple of equalities

$$\left[av^Tu + b\|u\|^2\right]\sigma_{n-1} + \left[a\|v\|^2 + bu^Tv\right] = \lambda_{n-1},$$

$$\left[b\|v\|^2 + cu^Tv\right] + \left[bv^Tu + c\|u\|^2\right]\sigma_{n-1} = \lambda_{n-1}\sigma_{n-1},$$

or equivalently

$$\begin{cases} \alpha\sigma_{n-1} + \beta = \lambda_{n-1}, \\ \\ \gamma + \delta\sigma_{n-1} = \lambda_{n-1}\sigma_{n-1}, \end{cases}$$

which give

$$\sigma_{n-1} = \frac{(\delta-\beta) \pm \sqrt{(\delta-\beta)^2+4\alpha\gamma}}{2\alpha}$$

and

$$\lambda_{n-1} = \alpha\sigma_{n-1} + \beta.$$

A similar analysis holds for the computation of $\lambda_n$, which completes the proof. □

In order to prove the next result, we highlight that by replacing (3.57) in (3.55), and recalling that $\sigma_k^2 = 1$, we obtain

$$\begin{aligned} M_{k+1} &= \delta_k M_k + \gamma_k^{(1)} \left[\delta_k^2 \left(M_k y_k\right)\left(M_k y_k\right)^T\right] + \gamma_k^{(2)} \frac{p_k p_k^T}{y_k^T p_k} \\ &\quad + \gamma_k^{(1)} \left[\left(\alpha_k - \gamma_k^{(2)}\right)^2 p_k p_k^T - \delta_k \left(\alpha_k - \gamma_k^{(2)}\right)\left(\left(M_k y_k\right) p_k^T + p_k \left(M_k y_k\right)^T\right)\right]. \end{aligned}$$

Hence, $M_{k+1}$ can be rewritten in the form (3.54), that is

$$M_{k+1} = \delta_k M_k + \Delta_k,$$

with

$$\Delta_k = \left(p_k \vdots M_k y_k\right) \begin{pmatrix} \gamma_k^{(1)}(\alpha_k - \gamma_k^{(2)})^2 + \dfrac{\gamma_k^{(2)}}{y_k^T p_k} & -\delta_k \gamma_k^{(1)}(\alpha_k - \gamma_k^{(2)}) \\ -\delta_k \gamma_k^{(1)}(\alpha_k - \gamma_k^{(2)}) & \delta_k^2 \gamma_k^{(2)} \end{pmatrix} \begin{pmatrix} p_k^T \\ \cdots \\ (M_k y_k)^T \end{pmatrix}.$$

**Proposition 3.3.15.** *Let $f$ be a continuously differentiable function. Suppose that the NCG method is used to minimize the function $f$. Suppose that* (3.53)-(3.59) *are satisfied and let us set the arbitrary sequence of positive values $\{\hat{\lambda}_k\}$. If $M_1 \succ 0$ with*

$$\lambda_m(M_1) \geq \hat{\lambda}_1 > 0,$$

*for any $k$ the vectors $p_k$ and $M_k y_k$ are not parallel, and the sequence $\{\delta_k\}$ in* (3.54) *satisfies*

$$\begin{cases} \delta_k \geq \dfrac{\hat{\lambda}_{k+1}}{\lambda_m(M_k)} & \text{if } \lambda_m(\Delta_k) \geq 0, \\ \delta_k \geq \dfrac{\hat{\lambda}_{k+1} - \lambda_m(\Delta_k)}{\lambda_m(M_k)} & \text{if } \lambda_m(\Delta_k) < 0, \end{cases} \tag{3.66}$$

*then the matrix $M_{k+1}$ in* (3.55) *is positive definite, for any $k \geq 1$, with*

$$\lambda_m(M_{k+1}) \geq \hat{\lambda}_{k+1}.$$

*Proof.* We prove the result by induction. Since by hypothesis $\lambda_m(M_1) \geq \hat{\lambda}_1 > 0$, then a sufficient condition to have $\lambda_m(M_2) \geq \hat{\lambda}_2 > 0$ is that $\lambda_m(\Delta_1) \geq 0$, so that relation $\lambda_m(M_2) \geq \delta_1 \lambda_m(M_1) \geq \hat{\lambda}_2$ is fulfilled choosing $\delta_1 \geq \hat{\lambda}_2 / \lambda_m(M_1)$. On the other hand, if $\lambda_m(\Delta_1) < 0$ we can always set $\hat{\lambda}_2$ such that

$$\lambda_m(M_2) = \lambda_m(\delta_1 M_1 + \Delta_1) \geq \delta_1 \lambda_m(M_1) + \lambda_m(\Delta_1) \geq \hat{\lambda}_2 > 0,$$

which is satisfied provided that

$$\delta_1 \geq \frac{\hat{\lambda}_2 - \lambda_m(\Delta_1)}{\lambda_m(M_1)}.$$

Note that by (3.65) of Lemma 3.3.14, regardless of the choice of $\gamma_1^{(1)}$ and $\gamma_1^{(2)}$ in (3.59), the quantity $\lambda_m(\Delta_1)$ can be obtained directly as

$$\lambda_m(\Delta_1) = \min\{\lambda_{n-1}, \lambda_n, 0\}, \tag{3.67}$$

where $\lambda_{n-1}$ and $\lambda_n$ are defined in (3.65), setting in Lemma 3.3.14

$$v = p_1, \quad u = M_1 y_1,$$

$$a = \gamma_1^{(1)}(\alpha_1 - \gamma_1^{(2)})^2 + \frac{\gamma_1^{(2)}}{y_1^T p_1}, \quad b = -\delta_1 \gamma_1^{(1)}(\alpha_1 - \gamma_1^{(2)}, \quad c = \delta_1^2 \gamma_1^{(1)}.$$

Now assume that by induction $\lambda_m(M_k) \geq \hat{\lambda}_k$, we want to prove the result for step $k+1$. To this purpose, recalling again that by Lemma 3.3.14 we can compute similarly to (3.67)

$$\lambda_m(\Delta_k) = \min\{\lambda_{n-1}, \lambda_n, 0\},$$

the choice (3.66) immediately yields $\lambda_m(M_{k+1}) \geq \hat{\lambda}_{k+1}$. □

The result in Proposition 3.3.15 gives a characterization of the spectrum of $M_{k+1}$, but possibly it does not indicate a procedure to set the parameters affecting the formula of $M_{k+1}$. Moreover, the hypothesis that for any $k$ the vectors $p_k$ and $M_k y_k$ are not parallel may be difficult to be guaranteed. Thus, in order to fill the latter gap and provide a set of parameters $\delta_k$, $\gamma_k^{(1)}$ and $\gamma_k^{(2)}$, such that conditions (3.59) are satisfied (i.e. equivalently (3.53) hold) and the preconditioner $M_k$ is positive definite, the following proposition may represent an operative tool. In particular, observe that unlike Proposition 3.3.15, the next result neither requires to compute $\lambda_m(\Delta_k)$ nor it needs to introduce the sequence $\{\hat{\lambda}_k\}$.

**Proposition 3.3.16.** *Let $f$ be a continuously differentiable function. Suppose that the NCG method is used to minimize the function $f$. Suppose that at current step $k$ the linesearch procedure satisfies $s_k^T y_k > 0$. Moreover, let $M_k \succ 0$ in (3.55), and set $\varepsilon_k \in (0,1)$, with*

$$0 < \delta_k = (1-\varepsilon_k)\frac{s_k^T y_k}{y_k^T M_k y_k}, \tag{3.68}$$

$$0 < \gamma_k^{(2)} < \varepsilon_k \alpha_k, \tag{3.69}$$

$$0 < \gamma_k^{(1)} = \frac{1}{(\varepsilon_k \alpha_k - \gamma_k^{(2)}) p_k^T y_k}. \tag{3.70}$$

*Then, conditions (3.53)-(3.59) hold and $M_{k+1} \succ 0$ in (3.55).*

*Proof.* By (3.68) and recalling that $\varepsilon_k \in (0,1)$, with $M_k \succ 0$, we obtain

$$0 < \delta_k < \frac{s_k^T y_k}{y_k^T M_k y_k},$$

which implies also

$$s_k^T y_k - \delta_k y_k^T M_k y_k > 0. \tag{3.71}$$

Now, observe that by the first relation (3.59), $\gamma_k^{(1)} > 0$ if and only if $s_k^T y_k - \delta_k y_k^T M_k y_k - \gamma_k^{(2)} p_k^T y_k > 0$, i.e.

$$\gamma_k^{(2)} < \frac{s_k^T y_k - \delta_k y_k^T M_k y_k}{p_k^T y_k},$$

which can be satisfied using (3.68) as

$$0 < \gamma_k^{(2)} < \frac{s_k^T y_k - \delta_k y_k^T M_k y_k}{p_k^T y_k} = \varepsilon_k \alpha_k.$$

The latter relation is indeed the condition (3.69), and satisfies also the second relation (3.59). Finally, by (3.68) the first relation (3.59) is equivalent to

$$\gamma_k^{(1)} = \frac{1}{s_k^T y_k - (1-\varepsilon_k) s_k^T y_k - \gamma_k^{(2)} p_k^T y_k} = \frac{1}{(\varepsilon_k \alpha_k - \gamma_k^{(2)}) p_k^T y_k},$$

with $(\varepsilon_k \alpha_k - \gamma_k^{(2)}) p_k^T y_k > 0$. Then, (3.68)-(3.70) yield exactly (3.53) and (3.59), along with $M_{k+1} \succ 0$. □

#### 3.3.5.1 Issues on ill-conditioning

The previous proposition ensures that properly choosing the parameters $\delta_k$, $\gamma_k^{(1)}$ and $\gamma_k^{(2)}$ the preconditioner $M_{k+1}$ is well-posed and positive definite. However, we should take into account that the search direction $p_k$ we compute at iteration $k$ of NCG could be not well scaled, which may introduce some ill-conditioning when applying the PNCG. Following the rationale behind the BFGS updates, a possible remedy to the latter drawback can be represented by reducing the *distance* between $M_{k+1}$ and $M_k$, i.e. minimizing the Frobenious norm $\|M_{k+1} - M_k\|_F$. In this regard, as well known we have

$$\|M_{k+1} - M_k\|_F = \sqrt{tr[(M_{k+1} - M_k)^T (M_{k+1} - M_k)]} = \sqrt{tr[(M_{k+1} - M_k)^2]}. \quad (3.72)$$

By the properties of the trace of matrices (see e.g. [22]), we have

$$\sqrt{tr[(M_{k+1} - M_k)^2]} \leq \sqrt{[tr(M_{k+1} - M_k)]^2} = |tr(M_{k+1} - M_k)|. \quad (3.73)$$

Thus, a possible upper bound for $\|M_{k+1} - M_k\|_F$ is given by

$$\|M_{k+1} - M_k\|_F \leq |tr(M_{k+1} - M_k)|. \quad (3.74)$$

We recall that by Proposition 3.3.16, $\{\gamma_k^{(1)}\}$ and $\{\gamma_k^{(2)}\}$ are positive sequences, so that using (3.55) in (3.74) we have

$$\begin{aligned} |tr(M_{k+1} - M_k)| &= \left| \delta_k tr(M_k) + \gamma_k^{(1)} \|v_k\|^2 + \gamma_k^{(2)} \frac{\|p_k\|^2}{y_k^T p_k} - tr(M_k) \right| \\ &\leq |\delta_k - 1| tr(M_k) + \gamma_k^{(1)} \|v_k\|^2 + \gamma_k^{(2)} \frac{\|p_k\|^2}{y_k^T p_k}. \end{aligned} \quad (3.75)$$

From (3.69) and (3.70) we observe that, after setting the arbitrary parameter $\varepsilon_k$, then $\gamma_k^{(1)}$ still depends on $\gamma_k^{(2)}$. Thus, following the rationale behind the BFGS update, in the following proposition we investigate possible values for the bound (3.75) on $tr(M_{k+1} - M_k)$, when $\gamma_k^{(2)}$ changes.

**Proposition 3.3.17.** *Let $f$ be a continuously differentiable function. Suppose that the NCG method is used to minimize $f$. Suppose that at current step $k$ we have $s_k^T y_k > 0$, $M_k \succ 0$, $\varepsilon_k \in (0,1)$, and let (3.53), (3.59), (3.68)-(3.70) be satisfied. Consider the function of $\gamma_k^{(2)}$*

$$\phi(\gamma_k^{(2)}) = |\delta_k - 1| tr(M_k) + \gamma_k^{(1)} \|v_k\|^2 + \gamma_k^{(2)} \frac{\|p_k\|^2}{y_k^T p_k}. \tag{3.76}$$

*Then $\phi(\gamma_k^{(2)})$ is monotone non decreasing with respect to $\gamma_k^{(2)}$, and $\gamma_k^{(2)} = 0$ minimizes it.*

*Proof.* After setting $\varepsilon_k \in (0,1)$, by (3.70) we note that $\gamma_k^{(1)}$ depends on $\gamma_k^{(2)}$. Thus, the function $\phi(\gamma_k^{(2)})$ in (3.76) uniquely depends on $\gamma_k^{(2)}$. Now we have for $\phi'(\gamma_k^{(2)})$

$$\phi'(\gamma_k^{(2)}) = \frac{d(\gamma_k^{(1)}\|v_k\|^2)}{d\gamma_k^{(2)}} + \frac{\|p_k\|^2}{y_k^T p_k} = \frac{d\gamma_k^{(1)}}{d\gamma_k^{(2)}}\|v_k\|^2 + \gamma_k^{(1)} \frac{d\left(\|v_k\|^2\right)}{d\gamma_k^{(2)}} + \frac{\|p_k\|^2}{y_k^T p_k}. \tag{3.77}$$

From (3.57) and $\sigma_k \in \{-1, +1\}$ we have

$$\|v_k\|^2 = \|p_k\|^2(\gamma_k^{(2)})^2 + 2(\delta_k p_k^T M_k y_k - \alpha_k \|p_k\|^2)\gamma_k^{(2)} + \|s_k - \delta_k M_k y_k\|^2, \tag{3.78}$$

so that, using (3.70) and (3.78) in (3.77) we obtain

$$\begin{aligned} \phi'(\gamma_k^{(2)}) \;=\; & \frac{\|p_k\|^2(\gamma_k^{(2)})^2 + 2(\delta_k p_k^T M_k y_k - \alpha_k\|p_k\|^2)\gamma_k^{(2)} + \|s_k - \delta_k M_k y_k\|^2}{(\varepsilon_k\alpha_k - \gamma_k^{(2)})^2 p_k^T y_k} \\ & + \frac{2\|p_k\|^2\gamma_k^{(2)} + 2(\delta_k p_k^T M_k y_k - \alpha_k\|p_k\|^2)}{(\varepsilon_k\alpha_k - \gamma_k^{(2)}) p_k^T y_k} + \frac{\|p_k\|^2}{p_k^T y_k}. \end{aligned} \tag{3.79}$$

Setting for the sake of clarity

$$A = \|p_k\|^2, \qquad B = \delta_k p_k^T M_k y_k - \alpha_k \|p_k\|^2, \qquad C = \|s_k - \delta_k M_k y_k\|^2, \qquad D = p_k^T y_k,$$

in (3.79), we obtain after some computation for $\phi'(\gamma_k^{(2)})$

$$\phi'(\gamma_k^{(2)}) = \frac{C + 2B\varepsilon_k\alpha_k + A\varepsilon_k^2\alpha_k^2}{D(\varepsilon_k\alpha_k - \gamma_k^{(2)})^2}. \tag{3.80}$$

Now, by (3.69) and $s_k^T y_k > 0$ the quantity $D(\varepsilon_k\alpha_k - \gamma_k^{(2)})^2$ is strictly positive, for any $\gamma_k^{(2)}$. Moreover, recalling that $s_k = \alpha_k p_k$ we have

$$\begin{aligned} & C + 2B\varepsilon_k\alpha_k + A\varepsilon_k^2\alpha_k^2 \;= \\ & = \|s_k\|^2 - 2\delta_k s_k^T M_k y_k + \delta_k^2\|M_k y_k\|^2 + 2(\delta_k p_k^T M_k y_k - \alpha_k\|p_k\|^2)\alpha_k\varepsilon_k + \|p_k\|^2\alpha_k^2\varepsilon_k^2 \\ & = \|s_k\|^2 - 2\delta_k s_k^T M_k y_k + \delta_k^2\|M_k y_k\|^2 + 2\varepsilon_k\delta_k s_k^T M_k y_k - 2\varepsilon_k\|s_k\|^2 + \varepsilon_k^2\|s_k\|^2 \\ & = (1-\varepsilon_k)^2\|s_k\|^2 - 2(1-\varepsilon_k)\delta_k s_k^T M_k y_k + \delta_k^2\|M_k y_k\|^2. \end{aligned} \tag{3.81}$$

Now, replacing $1 - \varepsilon_k = \theta$ in (3.81), we can introduce the function

$$\zeta(\theta) = C + 2B\varepsilon_k\alpha_k + A\varepsilon_k^2\alpha_k^2 = \theta^2\|s_k\|^2 - 2\theta\delta_k s_k^T M_k y_k + \delta_k^2\|M_k y_k\|^2, \qquad (3.82)$$

being after a simple computation $\zeta(\theta) \geq 0$, for any $\theta \in \mathbb{R}$. Hence, $\phi'(\gamma_k^{(2)}) \geq 0$ in (3.80) and $\phi(\gamma_k^{(2)})$ in (3.76) is monotone non decreasing. Finally, by (3.69) $\gamma_k^{(2)} = 0$ minimizes $\phi(\gamma_k^{(2)})$. □

The latter proposition gives some guidelines for the choice of the parameter $\varepsilon_k$ in (3.68)-(3.70), indicating that small positive values for $\varepsilon_k$ tend to reduce the value of $\gamma_k^{(2)}$ and can possibly control ill-conditioning of the matrix in (3.55).

#### 3.3.5.2 Numerical experience

Here we report the results of an extensive numerical experience, in order to validate the analysis and the theoretical achievements of the previous sections. We implement our proposal (3.55) and, analogously to preconditioner in Section 3.3.4, we embed it in a standard implementation of NCG, namely the code CG+ by Gilbert and Nocedal [51]. We select the Polak and Ribière method and, as regards the linesearch procedure, for a fair comparison, we used the default one in CG+ code. As concerns the parameters of the algorithm we use all the default values of CG+. According with Proposition 3.3.16, for the parameters affecting our proposal in (3.55) we used the settings

$$M_1 = I, \qquad \varepsilon_k = \frac{1}{2}, \qquad \delta_k = (1 - \varepsilon_k)\frac{s_k^T y_k}{y_k^T M_k y_k},$$

$$\gamma_k^{(2)} = \frac{1}{2}\varepsilon_k\alpha_k, \qquad \gamma_k^{(1)} = \frac{\alpha_k}{(\varepsilon_k\alpha_k - \gamma_k^{(2)})s_k^T y_k}.$$

In order to limit the computational burden as long as the storage requirement at iteration $k$, we preliminarily investigated the possibility to implement the preconditioner in (3.55) neglecting the information at iterations older than iteration $(k-m)$-th, playing $m$ the role of a "memory" parameter. The latter choice resembles the setting of the preconditioner proposed in Section 3.3.4. Not surprisingly, our numerical experience highlighted that this simplification does not deteriorate the performance. Indeed, this choice matches the rationale of the L-BFGS method and a value of $m = 4$ seems to provide the best compromise.

Analogously to the numerical results in Section 3.3.4 the stopping criterion we adopt is the standard one given by (see e.g. [78], [83], [93]), that is

$$\|g_k\| \leq 10^{-5}\max\{1, \|x_k\|\}.$$

As a test set for our numerical experience, we select all the large scale unconstrained test problems in CUTEst collection [56]. We consider those test problems whose dimension is in the range $n = 1000$ and $n = 10000$, and whenever a variable-dimension problem is used, two different sizes are included. As in the previous Section, this sums up to 112 resulting problems. The results are reported in terms of number of iterations and number of function evaluations. As usual, when comparing

two algorithms we exclude all the test problems where the compared alternatives do not converge to the same stationary point. In Figures 3.3-3.4 we report the comparison between the two algorithms.

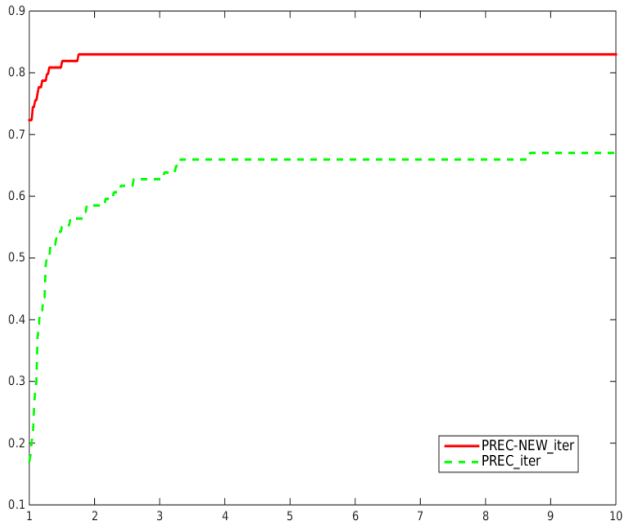


**Figure 3.3.** Comparison between our proposal in (3.55) (namely PREC-NEW, *solid line*) and the proposal of preconditioner in Section 3.3.4 (namely PREC, *dotted line*), in terms of number of *iterations*.

We can easily observe that our proposal definitely outperforms the one in Section 3.3.4 (both in terms of iterations and function evaluations). This should be due to the fact that the preconditioner in (3.55) seems to better exploit the information collected in the history of the overall algorithm, imposing a modified quasi-Newton equation. The profiles reveal an appreciable improvement of the efficiency as long as the robustness.

## 3.4 Conclusions

In this chapter we have focused on how to improve NCG methods by using preconditioning techniques. In particular we have proposed new preconditioners drawing inspiration from quasi-Newton updates. In Sections 3.3.1-3.3.3 we have introduced three classes of quasi-Newton updates. Their drawback is that these preconditioners could be not positive definite. To overcome this drawback, in Section 3.3.4 we have proposed new quasi-Newton updates to iteratively construct a preconditioner for PNCG method such that:

- it is positive definite;
- it satisfies the secant equation at the current iteration.

The results obtained showed that the preconditioners we have proposed are definitely much efficient and robust in optimization frameworks. However the search direction

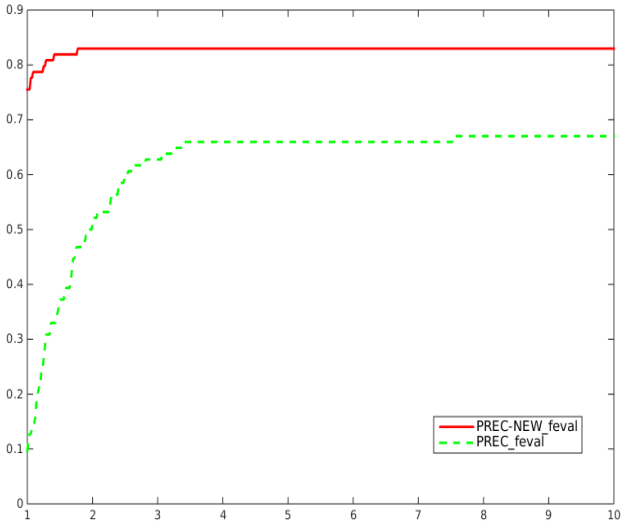


**Figure 3.4.** Comparison between our proposal in (3.55) (namely PREC-NEW, *solid line*) and the proposal of preconditioner in Section 3.3.4 (namely PREC, *dotted line*), in terms of number of *function evaluations*.

we conpute does not seem yet well scaled. On this purpose, in Section 3.3.5 we have introduced a new preconditioner such that:

- it is positive definite;
- it satisfies the modified secant equation at each iteration.

Numerical results showed an improvement of the preconditioner in Section 3.3.4 but the search direction possibly needs a better scaling.

# Chapter 4

# Damped techniques for NCG methods

In this chapter we propose the use of damped techniques within NCG methods. Damped techniques were introduced by Powell and recently reproposed by Al-Baali and, to the best of our knowledge, only applied in the framework of quasi-Newton methods. We extend their use to NCG methods in large scale unconstrained optimization, aiming at possibly improving the efficiency and the robustness of the latter methods, especially when solving difficult problems. In this chapter we embed damped techniques within PNCG method which makes use of preconditioners based on quasi-Newton updates, proposed in Section 3.3.4. Most of the material of this chapter is contained in [4], [5].

## 4.1 Introduction

In this chapter we aim at extending the use of damped techniques to both NCG and PNCG methods. To this purpose, the following possibilities can be considered:

- *Modified* methods. In this case a damped technique is only used to modify the scalar (usually denoted by $\beta_k$) which characterizes the different NCG methods. The search direction is therefore modified, hence the necessity to ensure the global convergence of the resulting novel NCG method, the damped one.

- *Preconditioned* methods. They are obtained without modifying the original expression of the scalar $\beta_k$. Here, the damped techniques are only used for constructing a preconditioner based on quasi-Newton updates. In this case we do not obtain a novel NCG algorithm or focus on a particular NCG method. On the contrary, we have a new methodology for defining preconditioning strategies, to be possibly used within any NCG method for improving its performance.

- *Modified preconditioned* methods. In this case, a damped technique is used both to modify the scalar $\beta_k$ and to construct a suitable preconditioner for NCG schemes.

We deal with all the three items above, even if the main focus is actually on the second one. Indeed, we believe that, since damped techniques were conceived in the framework of quasi-Newton methods, we expect to inherit their good features when building a preconditioner based on quasi-Newton updates. To this aim, we introduce two different damping strategies, which seem to be suited for our purposes. In particular, we focus on Polak and Ribière (PR) method, proving that under reasonable assumptions, the damped and preconditioned version of this method (denoted by D-PR-PNCG) to some extent retains the convergence properties of the (undamped and unpreconditioned) PR method.

We propose to combine damped techniques with preconditioning strategies, aiming at making the resulting D-PR-PNCG method able to efficiently tackle also difficult problems. To this aim, in order to perform extensive numerical results, we consider the class of preconditioners based on quasi-Newton updates, which has been introduced in Section 3.3.4. The rationale behind the idea of adopting a damped strategy in defining preconditioners for NCG methods, relies on the fact that an approximation of the (inverse of the) Hessian matrix by means of a positive definite matrix is required. Therefore, modifying the quasi-Newton updates used for building a preconditioner for NCG methods, in order to prevent the lack of positive definiteness of the Hessian matrix, sounds meaningful. An extensive numerical experience confirmed this fact, showing the possible fruitful use of the damped techniques in constructing preconditioners for NCG, based on quasi-Newton updates. For the sake of completeness, we also report results obtained by using the *modified* methods, which do not seem to produce noticeable improvement in terms of efficiency and robustness.

## 4.2 Novel damped strategies for NCG preconditioning

In this section we introduce two novel damped strategies, to be considered within NCG methods, along with an adaptive criterion for deciding if it is worth to replace the undamped vector with the damped one. In the sequel, whenever we consider the preconditioned case, we refer to a positive definite preconditioner based on quasi-Newton updates, which will be denoted by $P_k(y_k, s_k)$, to evidence the current pair $(y_k, s_k)$ used for constructing the quasi-Newton update.

Drawing inspiration from the Al-Baali-Powell proposals briefly described in Section 2.1.2.1, now we aim at defining modifications of the vector $y_k$ which should lead to obtain more efficient and/or robust NCG methods. Once a damped vector $\widehat{y}_k$ is defined, it can be used:

*(i)* in the definition of $\beta_k$, replacing $y_k$ with $\widehat{y}_k$ (*modified method*);

*(ii)* in the definition of the preconditioner replacing $P_k(y_k, s_k)$ with $P_k(\widehat{y}_k, s_k)$.

In order to clearly evaluate the effect of the damped techniques, we study the cases *(i)* and *(ii)* separately. Furthermore, we also investigate the joint modification of both $\beta_k$ and $P_k(y_k, s_k)$, by means of the damped vector $\widehat{y}_k$ (*modified preconditioned method*). Note that in the unpreconditioned case, a damped strategy obviously may affect the definition of $\beta_k$ only when $y_k$ explicitly appears in the formula of $\beta_k$.

Broadly speaking, in extending the definition of the damped vector $\widehat{y}_k$ introduced in (2.61), that is

$$\widehat{y}_k = \varphi_k y_k + (1-\varphi_k) B_k s_k,$$

our aim is to define a vector $\widehat{y}_k$ as a combination of the original vector $y_k$ and an appropriate vector $z_k$, namely

$$\widehat{y}_k = \varphi_k y_k + (1-\varphi_k) z_k, \tag{4.1}$$

such that $s_k^T \widehat{y}_k$ is sufficiently positive for suited values of $\varphi_k \in (0,1]$. Of course, a key point of this approach is an appropriate choice of $z_k$, both in terms of certain gained information and in terms of a good relative scaling of $\widehat{y}_k$. Note that the choice (4.1) is reduced to (2.61) if $z_k = B_k s_k$, which cannot be computed explicitly in the NCG context, being $B_k$ unavailable. In the following, two proposals will be provided.

### 4.2.1 Our first proposal

In this proposal, we set $z_k = \eta_k s_k$, where $\eta_k$ is a positive scalar, based on approximating $B_k$ by $\eta_k I$. It originates from the idea of using $z_k = A_{k+1} y_k$ in (4.1), where $A_{k+1}$ is a positive definitive approximation of the inverse Hessian, satisfying the *modified secant equation*

$$A_{k+1} y_k = \eta_k s_k.$$

Hence, by using the latter equation, we can define the damped formula

$$\widehat{y}_k^{(1)} \;=\; \varphi_k y_k + (1-\varphi_k)\eta_k s_k, \tag{4.2}$$

which does not require the explicit knowledge of the approximate inverse $A_{k+1}$ of the Hessian matrix. Since (4.2) follows from (2.61) with $B_k s_k$ replaced by $\eta_k s_k$, we use the same replacement in (2.62) to obtain the following formula

$$\varphi_k = \begin{cases} \dfrac{\sigma \eta_k \|s_k\|^2}{\eta_k \|s_k\|^2 - s_k^T y_k} & \text{if} \quad s_k^T y_k < (1-\sigma)\eta_k \|s_k\|^2 \\ \\ 1, & \text{otherwise,} \end{cases} \tag{4.3}$$

where $\eta_k \geq 1$.

Then, in order to set $\varphi_k \neq 1$ only whenever $s_k^T y_k$ is sufficiently small, we modify (4.3) as

$$\varphi_k = \begin{cases} \dfrac{\sigma \eta_k \|s_k\|^2}{\eta_k \|s_k\|^2 - s_k^T y_k} & \text{if} \quad s_k^T y_k < (1-\sigma)\|s_k\|^2 \\ \\ 1, & \text{otherwise.} \end{cases} \tag{4.4}$$

Note that on some iterations, the former formula may modify $y_k$, because the condition in the former formula, i.e.

$$s_k^T y_k < (1-\sigma)\eta_k \|s_k\|^2, \tag{4.5}$$

can be satisfied for sufficiently large values of $\eta_k$. For certain choices of $\eta_k$, our numerical experience (which we will describe in Section 4.3) was carried on adopting

(4.4), which showed favourable results avoiding the dependence on the product $(1-\sigma)\eta_k$. Nevertheless, in our opinion, the numerical impact of (4.5) deserves further investigations.

We now give an alternative motivation for choice (4.2) which, in practice, represents a combination of $y_k$ and $s_k$ with the scalar $\eta_k$. Moreover, we can derive the novel adaptive criterion used in (4.4) starting from a geometric interpretation of the curvature condition (2.15), that is,

$$s_k^T y_k > 0.$$

As already mentioned, (see also [25]) if $f$ is strongly convex, the curvature condition (2.15) holds. Roughly speaking, $f$ strongly convex means that its curvature is positive and not too close to zero. Hence, motivated by the former idea of Powell in [99], we intend to define a criterion based on the (local) strong convexity of the function for deciding if a damped vector $\widehat{y}_k$ must be used in place of $y_k$.

It is well known that if $f$ is strongly convex on a convex set $S \subseteq \mathbb{R}^n$, then there exists $\theta > 0$ such that

$$[\nabla f(y) - \nabla f(x)]^T (y-x) \geq \theta \|y-x\|^2, \tag{4.6}$$

for all $x$ and $y$ belonging to $S$. For $\theta = 0$, we recover the basic inequality characterizing the convexity, namely the curvature condition, provided by the Wolfe line search procedure. For $\theta > 0$, we obtain a strong lower bound in (4.6). Hence, given $\theta > 0$, if we adopt (4.6) as selection criterion, we actually obtain the one used in (4.4) with $\theta = 1-\sigma$. Therefore, the rationale behind this criterion is the following: whenever $s_k^T y_k \geq (1-\sigma)\|s_k\|^2 > 0$ and hence the curvature is "sufficiently positive", there is no need to modify the vector $y_k$; otherwise the damped vector $\widehat{y}_k$ is considered.

Now, we remark that we are interested in obtaining the vector $\widehat{y}_k$ such that $s_k^T \widehat{y}_k$ is sufficiently positive, and that an improvement in the curvature condition is obtained, namely

$$s_k^T \widehat{y}_k^{(1)} \geq s_k^T y_k. \tag{4.7}$$

Recalling that we are considering in (4.4) the case $s_k^T y_k < (1-\sigma)\|s_k\|^2$, by substituting the value of $\varphi_k$ in (4.2), by simple computation we obtain

$$s_k^T \widehat{y}_k^{(1)} = (1-\sigma)\eta_k \|s_k\|^2. \tag{4.8}$$

Therefore $s_k^T \widehat{y}_k^{(1)}$ is sufficiently positive for suited values of the parameters $\sigma$ and $\eta_k$. Moreover, we can guarantee that $\widehat{y}_k^{(1)}$ satisfies (4.7) by setting $\eta_k > 1$ whenever $s_k^T y_k > 0$. On the other hand, if $s_k^T y_k$ is negative, (4.7) is trivially satisfied by the choice $\widehat{y}_k^{(1)}$.

### 4.2.2 Our second proposal

In this proposal we set $z_k = -\alpha_k g_k$ in (4.2) to obtain the damped vector

$$\widehat{y}_k^{(2)} \quad = \quad \varphi_k y_k - (1-\varphi_k)\alpha_k g_k \tag{4.9}$$

which arises from the following observation: if $B_k \succ 0$ is an approximation of the Hessian and we consider as search direction $-B_k^{-1}g_k$, it immediately follows that $s_k = x_{k+1} - x_k = -\alpha_k B_k^{-1} g_k$ which implies

$$B_k s_k = -\alpha_k g_k.$$

This allows us to consider the original damped vector (2.61), without computing $B_k$ explicitly, i.e. by replacing $B_k s_k$ with $-\alpha_k g_k$, as defined in (4.9). In this case adapting the Powell's rule in (2.62) (replacing $B_k s_k$ with $-\alpha_k g_k$), it follows that

$$\varphi_k = \begin{cases} \dfrac{\sigma \alpha_k s_k^T g_k}{\alpha_k s_k^T g_k + s_k^T y_k}, & \text{if } s_k^T y_k < -(1-\sigma)\alpha_k s_k^T g_k, \\ \\ 1, & \text{otherwise.} \end{cases} \tag{4.10}$$

Substituting the value of $\varphi_k$ from the first case (i.e. $\varphi_k \neq 1$) into (4.9), we obtain

$$s_k^T \widehat{y}_k^{(2)} = -\alpha_k (1-\sigma) s_k^T g_k = -\alpha_k^2 (1-\sigma) p_k^T g_k > 0,$$

where the last inequality follows since $p_k$ is a descent direction at $x_k$. Moreover, here we also have that the final steplength computed by the line search procedure plays a keynote role. Following guidelines adopted to obtain (4.4), formula (4.10) can be changed to define

$$\varphi_k = \begin{cases} \dfrac{\sigma \eta_k \alpha_k s_k^T g_k}{\eta_k \alpha_k s_k^T g_k + s_k^T y_k}, & \text{if } s_k^T y_k < -(1-\sigma)\alpha_k s_k^T g_k, \\ \\ 1, & \text{otherwise,} \end{cases} \tag{4.11}$$

where $\eta_k \geq 1$. Furthermore, similar formulae with the three cases in (2.63) can be also defined. Finally, observe that in our first proposal the conditions (4.4)-(4.8) omit the dependency on any considerations regarding the global convergence of the final damped techniques. In this regard, a further study on the latter issue (see also [2], [3], [11]) seems to be necessary, which will be the object of future research.

## 4.3 Numerical experience

In this section we consider the use of the damped vectors defined in Sections 4.2.1-4.2.2, for constructing a preconditioner based on quasi-Newton updates. Therefore, according to the taxonomy in Section 4.1, here we consider *unmodified* PNCG methods, where the use of damped techniques only affects the preconditioning strategies and not the value of $\beta_k$. Our aim is to perform a numerical assessment when adopting damped techniques within a PNCG method. On the other hand, note that as regards the convergence (and the order of convergence) of PNCG methods, an interesting theoretical result has been proved in [6]. However, it considers the use of an exact linesearch and a strong assumption on the preconditioner is required, namely the preconditioner is assumed to be a "strongly consistent approximation" of

the Hessian matrix at the solution. Therefore this result risks to be seldom applied in practice.

The preconditioner we use is the approximate inverse preconditioner belonging to the class proposed in Section 3.3.4. Since it is iteratively constructed, it is quite simple to introduce an adaptive rule and to choose, at each iteration $k$, if it is convenient to replace $y_k$ with a damped vector $\widehat{y}_k$. If so, the resulting preconditioner $P_k(\widehat{y}_k, s_k)$ is then used in place of $P_k(y_k, s_k)$.

We embedded the latter strategy in the implementation of the PNCG described in Section 3.3.4. Note that this implementation is based on the standard CG+ code (see [51]), where the preconditioner reported is included, and the linesearch technique is the the default one in CG+ code.

In particular, we focused on the *unmodified* preconditioned Polak and Ribière method and performed an extensive numerical testing by considering all the large scale problems available in the CUTEst collection [56], namely 112 problems whose dimension ranges from 1000 to 10000, analogously to the ones used previously in this thesis. The stopping criterion is the standard one (see e.g. [83]) which is given by

$$\|g_k\| \leq 10^{-5} \max\{1, \|x_k\|\}.$$

In the sequel our numerical results are reported by using performance profiles [42], both in terms of number of iterations and number of function and gradient evaluations.

We started by considering our first proposal in Section 4.2.1, namely the use of the damped vector $\widehat{y}_k^{(1)}$ in (4.2) combined with the adaptive rule in (4.4). First of all, we needed to tune the choice of the two parameters $\eta_k$ in (4.2) and $\sigma$ in (4.4). In Figures 4.1-4.4 we report the results obtained for different choices of $\eta_k \in \{2, 3, 4, 5\}$ and by setting $\sigma = 0.8$. Conversely, in Figures 4.5-4.8 we report the results obtained for different choices of $\sigma \in \{0.8, 0.6, 0.4, 0.2\}$ and by setting $\eta_k = 4$. By observing the profiles, the values $\eta_k = 4$ and $\sigma = 0.8$ seem to be the best ones, based on our experiments on the above mentioned set of test problems. These latter have been used in the sequel of this section as default values of $\eta_k$ and $\sigma$.

Figures 4.9-4.12 report the results of the comparison between the *unmodified preconditioned* PR method, whose preconditioner is damped according to the formula (4.2), and the standard preconditioned PR method. These profiles clearly evidence the fruitful use of the first damped strategy both in terms of efficiency and in terms of robustness.

Then we turned to our second proposal in Section 4.2.2, namely the use of the damped vector $\widehat{y}_k^{(2)}$ in (4.9) combined with the original rule in (2.62) for choosing $\varphi_k$ (with $B_k s_k$ replaced by $-\alpha_k g_k$) with $\sigma = 0.8$. In the Figures 4.13-4.16 the comparison between the *unmodified preconditioned* PR method, whose preconditioner is damped according to formula (4.9), and the standard preconditioned PR method is reported. Also in this case the adoption of the damped strategy for computing the preconditioner is very useful.

Now, since the preconditioner in Section 3.3.5 outperforms the one in Section 3.3.4 (both in terms of iterations and function evaluations), in Figures 4.17-4.20 we report the results of the comparison between the *unmodified preconditioned* PR method, whose preconditioner is damped according to the formula (4.2) in the first proposal

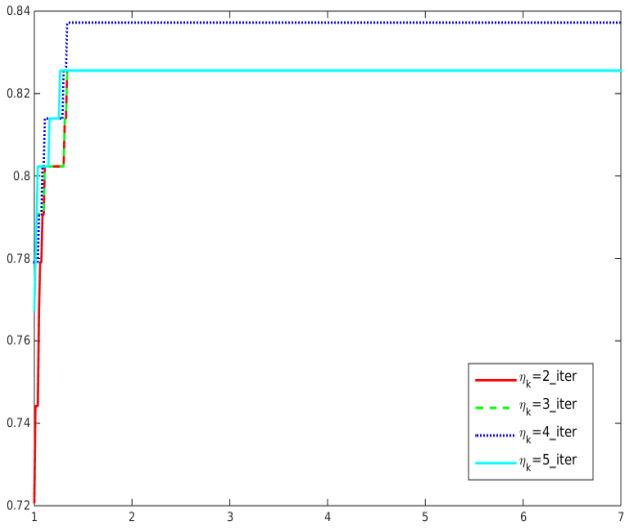


**Figure 4.1.** Comparison among different choices of $\eta_k$ in (4.2), setting $\sigma = 0.8$ in (4.4). Profile in terms of number of *iterations*.

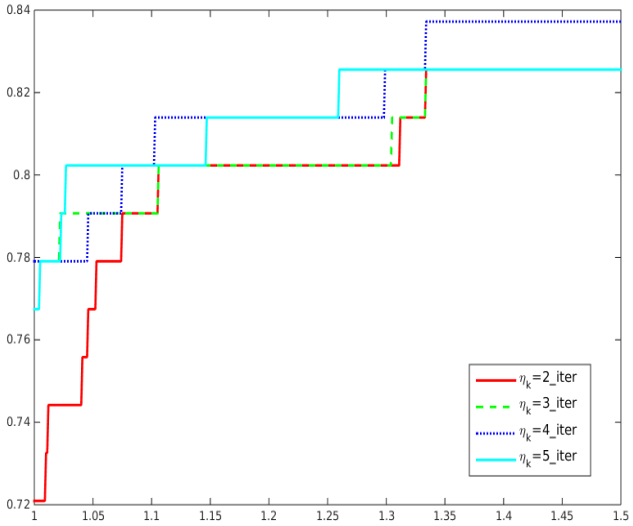


**Figure 4.2.** Comparison among different choices of $\eta_k$ in (4.2), setting $\sigma = 0.8$ in (4.4). Detailed profile in terms of number of *iterations*.

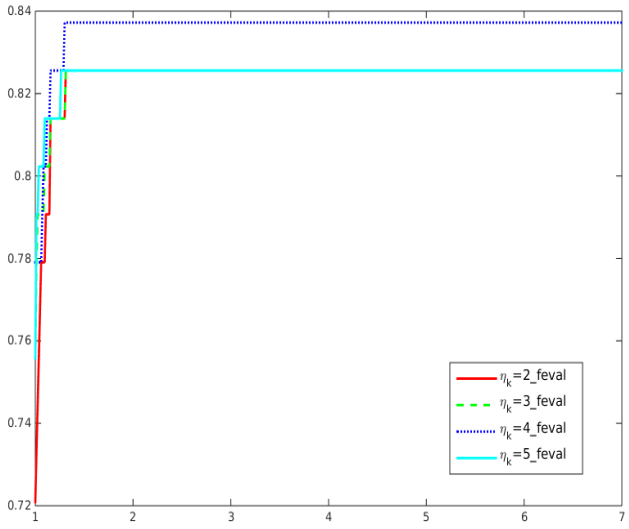


**Figure 4.3.** Comparison among different choices of $\eta_k$ in (4.2), setting $\sigma = 0.8$ in (4.4). Profile in terms of number of *function evaluations*.

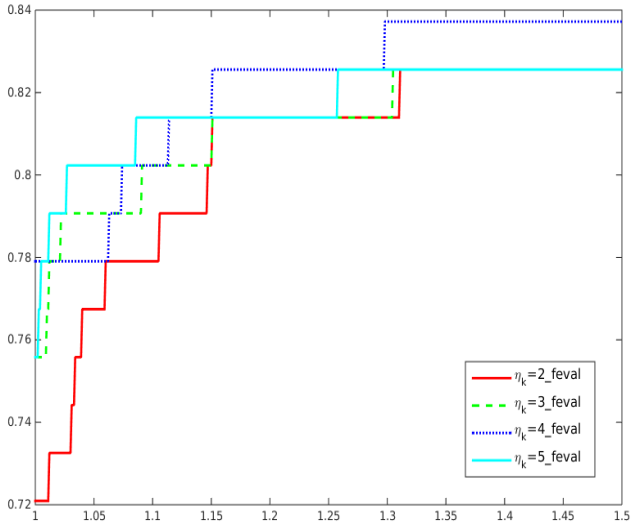


**Figure 4.4.** Comparison among different choices of $\eta_k$ in (4.2), setting $\sigma = 0.8$ in (4.4). Detailed profile in terms of number of *function evaluations*.

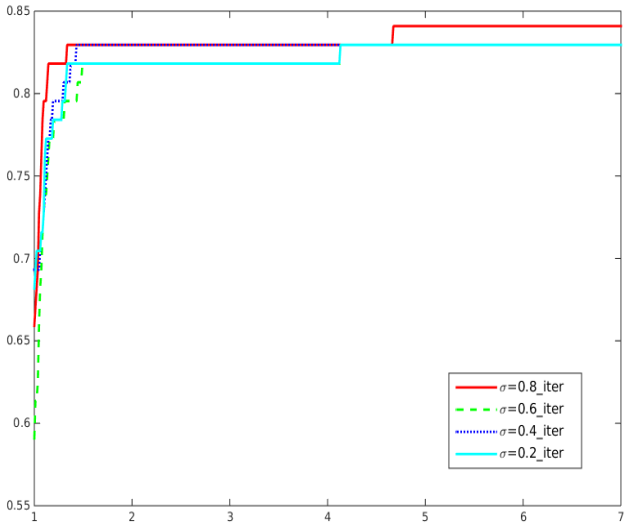


**Figure 4.5.** Comparison among different choices of $\sigma$ in (4.4), setting $\eta_k = 4$ in (4.2). Profile in terms of number of *iterations*.

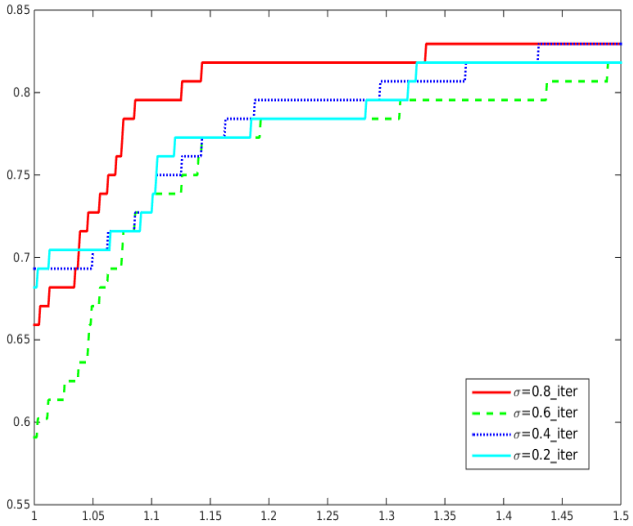


**Figure 4.6.** Comparison among different choices of $\sigma$ in (4.4), setting $\eta_k = 4$ in (4.2). Detailed profile in terms of number of *iterations*.

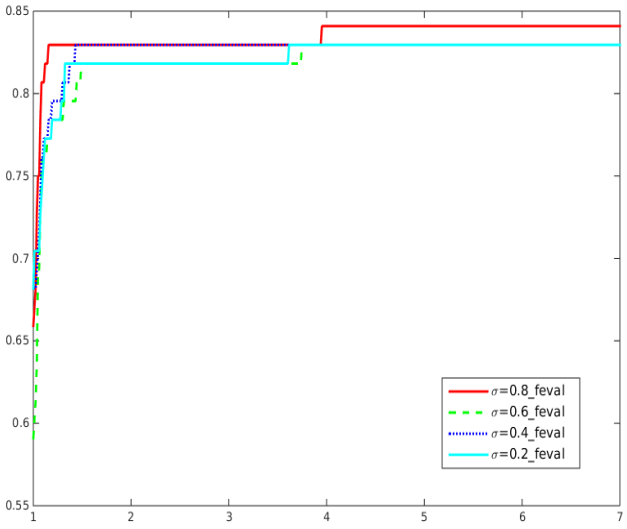


**Figure 4.7.** Comparison among different choices of $\sigma$ in (4.4), setting $\eta_k = 4$ in (4.2). Profile in terms of number of *function evaluations*.

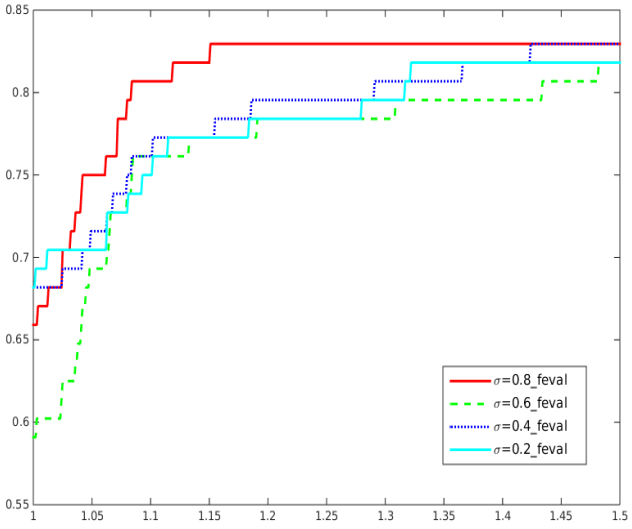


**Figure 4.8.** Comparison among different choices of $\sigma$ in (4.4), setting $\eta_k = 4$ in (4.2). Detailed profile in terms of number of *function evaluations*.

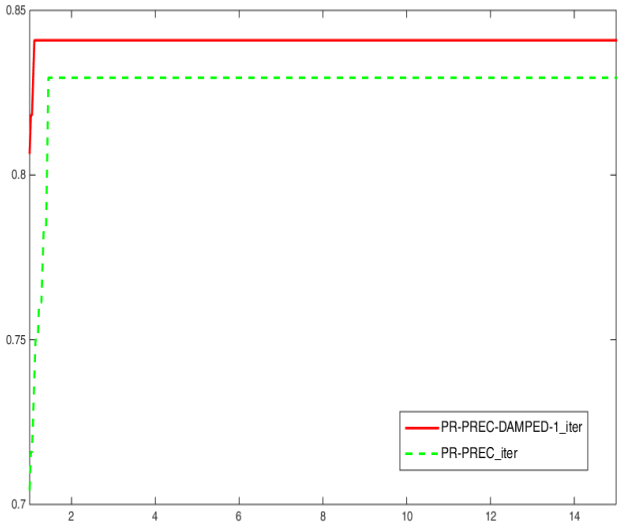


**Figure 4.9.** Comparison between *unmodified preconditioned* PR damped according to (4.2) and the standard preconditioned PR (undamped). Profile in terms of number of *iterations*.

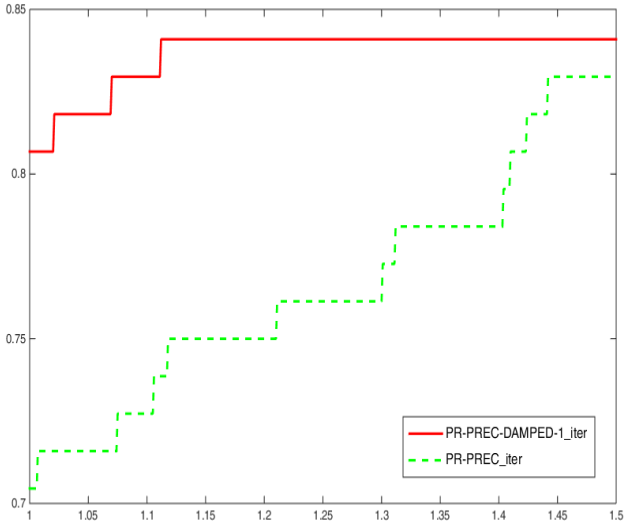


**Figure 4.10.** Comparison between *unmodified preconditioned* PR damped according to (4.2) and the standard preconditioned PR (undamped). Detailed profile in terms of number of *iterations*.

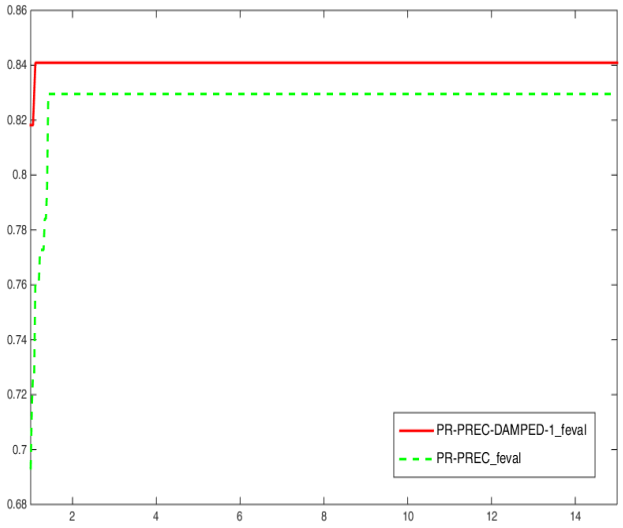


**Figure 4.11.** Comparison between *unmodified preconditioned* PR damped according to (4.2) and the standard preconditioned PR (undamped). Profile in terms of number of *function evaluations*.

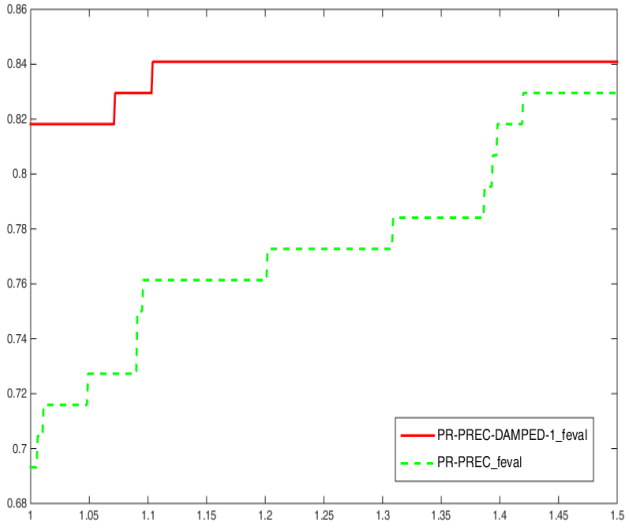


**Figure 4.12.** Comparison between *unmodified preconditioned* PR damped according to (4.2) and the standard preconditioned PR (undamped). Detailed profile in terms of number of *function evaluations*.

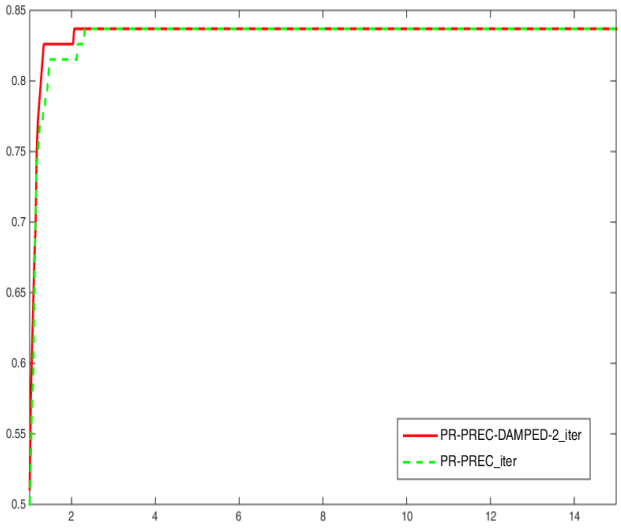


**Figure 4.13.** Comparison between *unmodified preconditioned* PR damped according to (4.9) and the standard preconditioned PR (undamped). Profile in terms of number of *iterations*.

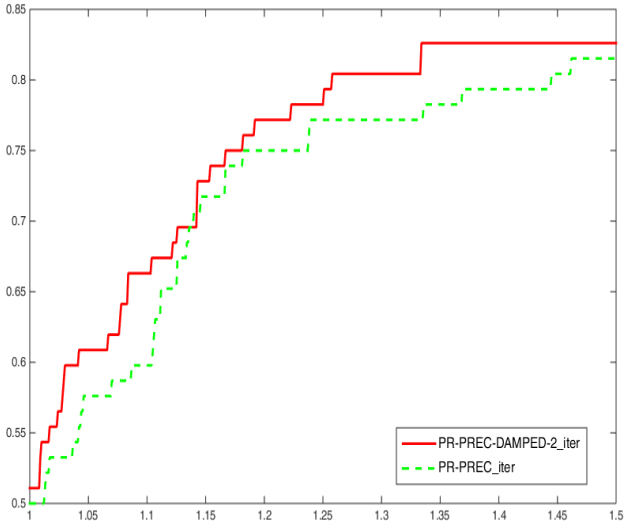


**Figure 4.14.** Comparison between *unmodified preconditioned* PR damped according to (4.9) and the standard preconditioned PR (undamped). Detailed profile in terms of number of *iterations*.

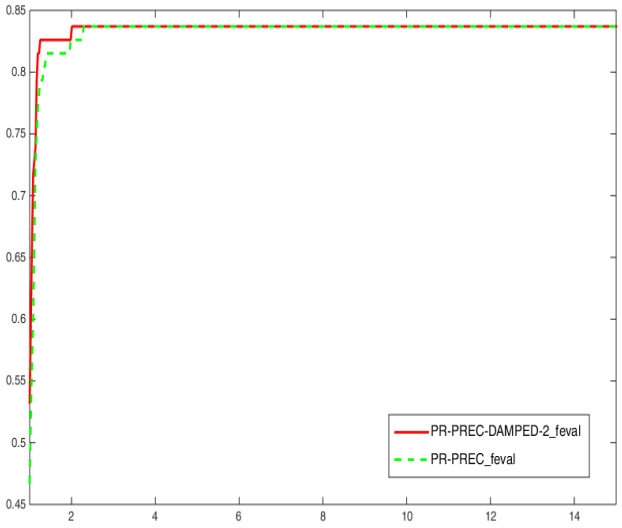


**Figure 4.15.** Comparison between *unmodified preconditioned* PR damped according to (4.9) and the standard preconditioned PR (undamped). Profile in terms of number of *function evaluations*.

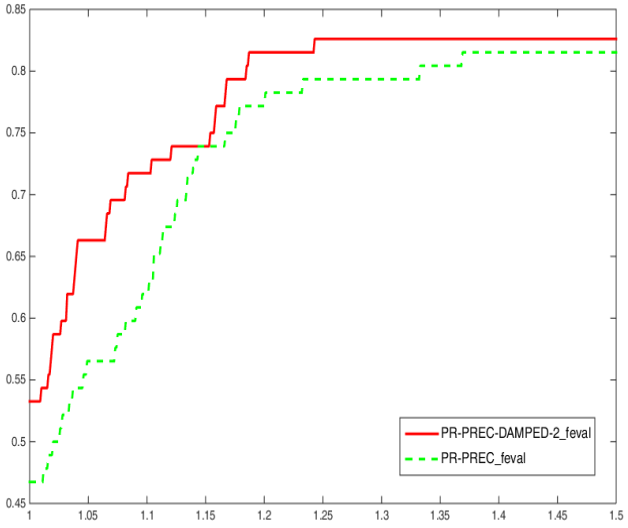


**Figure 4.16.** Comparison between *unmodified preconditioned* PR damped according to (4.9) and the standard preconditioned PR (undamped). Detailed profile in terms of number of *function evaluations*.

(see Section 4.2.1), and the preconditioned PR method (according to Section 3.3.5). These profiles clearly evidence again the fruitful use of the first damped strategy both in terms of efficiency and in terms of robustness.

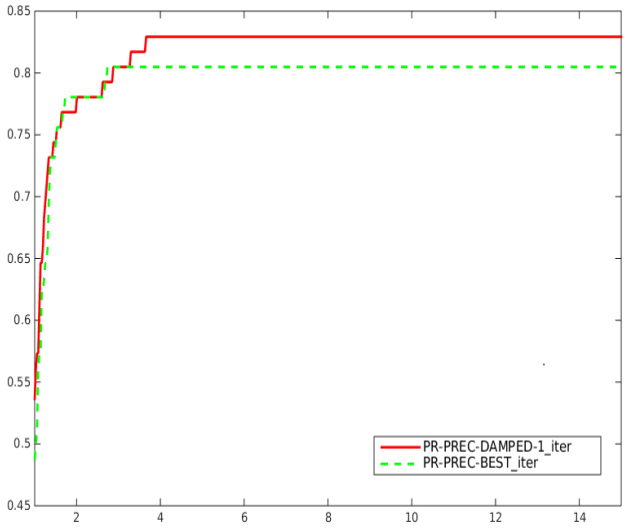


**Figure 4.17.** Comparison between *unmodified preconditioned* PR damped according to (4.2) and the preconditioned PR (undamped) according to Section 3.3.5. Profile in terms of number of *iterations*.

Then we again turned to our second proposal (see Section 4.2.2), namely the use of the damped vector $\widehat{y}_k^{(2)}$ in (4.9) combined with the original rule in (2.62) for choosing $\varphi_k$ (with $B_k s_k$ replaced by $-\alpha_k g_k$) with $\sigma = 0.8$. In the Figures 4.21-4.24 the comparison between the *unmodified preconditioned* PR method, whose preconditioner is damped according to formula (4.9), and the preconditioned PR method (according to Section 3.3.5). These profiles show that, both considering the number of iterations and the number of function evaluations, the use of the second damped strategy seems to be slightly worse in terms of efficiency but slightly better in terms of robustness.

On this guideline, we compared the two damped strategies proposed in this Chapter. The results of this comparison are reported in Figures 4.25-4.28. By observing these profiles, the adoption of the first damped strategy (see Section 4.2.1) seems to be slightly preferable.

In order to verify the effectiveness of our first proposal in Section 4.2.1, we perform a comparison between our first damped strategy in (4.2) and the benchmark algorithm given by the L-BFGS method (see [78], [93]). To this aim, we use the L-BFGS code available at the J. Nocedal web page. Observe that, as reported in the L-BFGS code, the linesearch procedure used therein slightly differs from the original one in [84], as (quoting from the Fortran code) *"Moré's code has been modified so that at least one new function value is computed during the line search (enforcing at least one interpolation is not easy, since the code may override an interpolation)"*. In this comparison, we also adopt this modified linesearch procedure within our PNCG

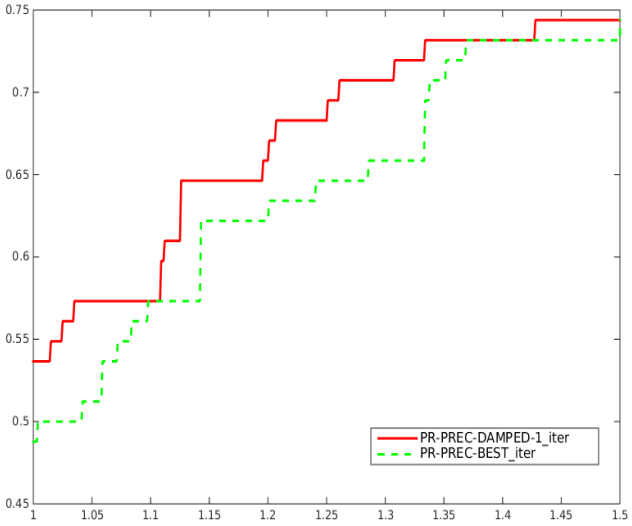


**Figure 4.18.** Comparison between *unmodified preconditioned* PR damped according to (4.2) and the preconditioned PR (undamped) according to Section 3.3.5. Detailed profile in terms of number of *iterations*.

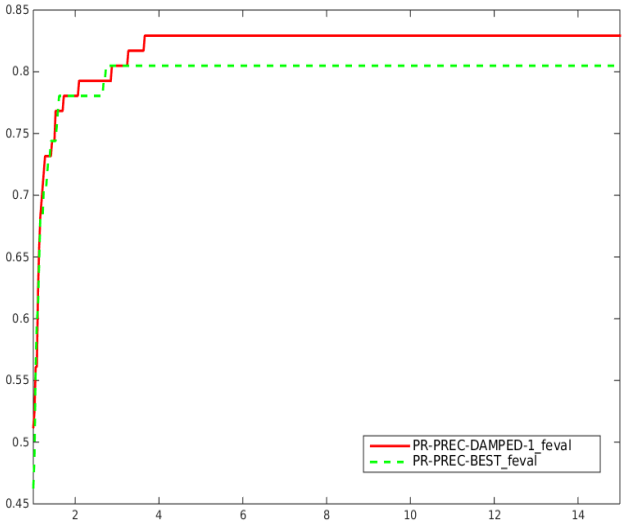


**Figure 4.19.** Comparison between *unmodified preconditioned* PR damped according to (4.2) and the preconditioned PR (undamped) according to Section 3.3.5. Profile in terms of number of *function evaluations*.

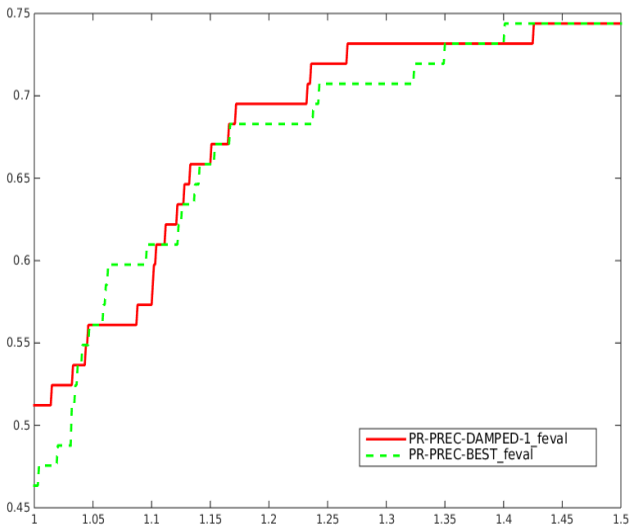


**Figure 4.20.** Comparison between *unmodified preconditioned* PR damped according to (4.2) and the preconditioned PR (undamped) according to Section 3.3.5. Detailed profile in terms of number of *function evaluations*.

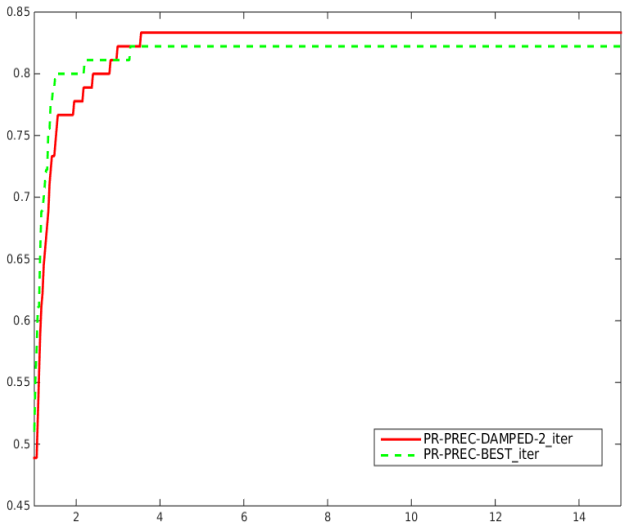


**Figure 4.21.** Comparison between *unmodified preconditioned* PR damped according to (4.9) and the preconditioned PR (undamped) according to Section 3.3.5. Profile in terms of number of *iterations*.

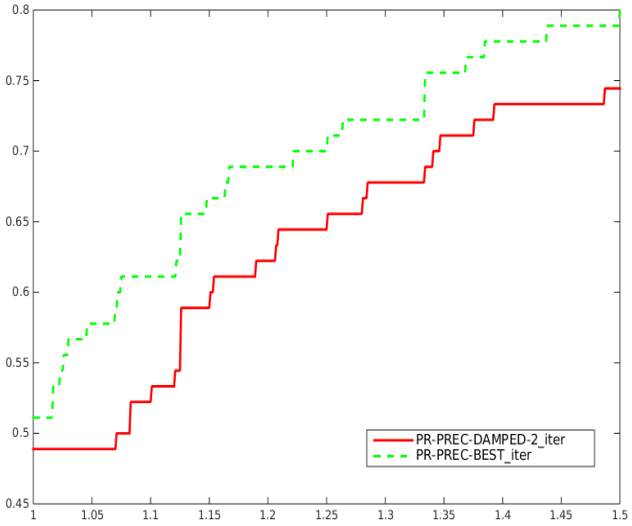


**Figure 4.22.** Comparison between *unmodified preconditioned* PR damped according to (4.9) and the preconditioned PR (undamped) according to Section 3.3.5. Detailed profile in terms of number of *iterations*.

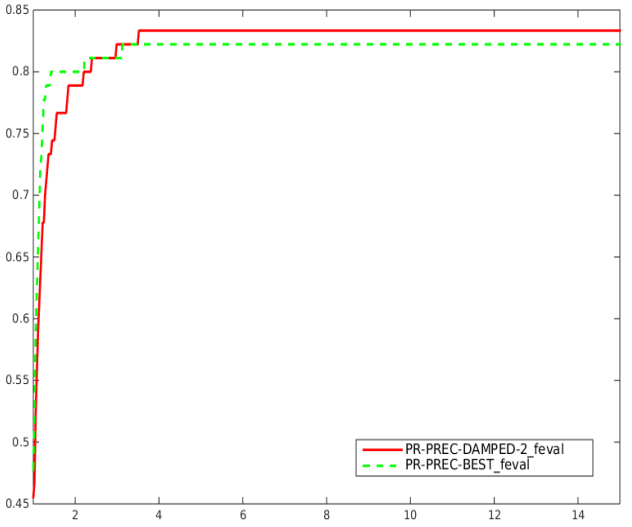


**Figure 4.23.** Comparison between *unmodified preconditioned* PR damped according to (4.9) and the preconditioned PR (undamped) according to Section 3.3.5. Profile in terms of number of *function evaluations*.

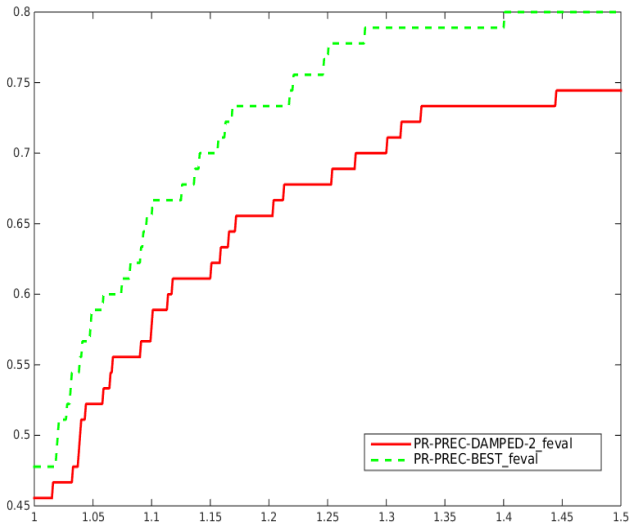


**Figure 4.24.** Comparison between *unmodified preconditioned* PR damped according to (4.9) and the preconditioned PR (undamped) according to Section 3.3.5. Detailed profile in terms of number of *function evaluations*.

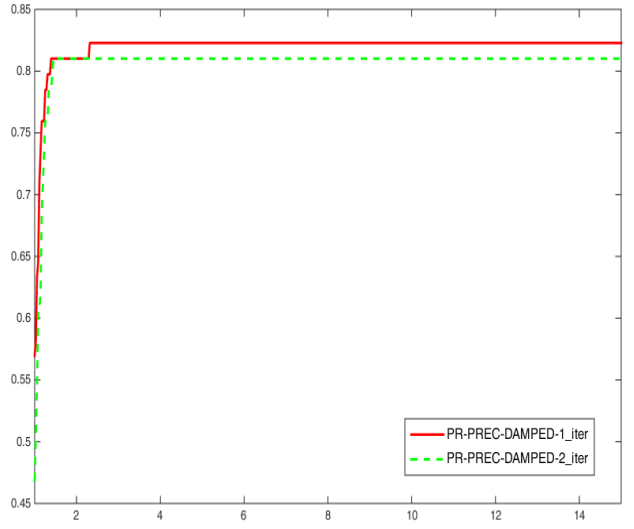


**Figure 4.25.** Comparison between the adoption of the two damped strategies in (4.2) and in (4.9). Profile in terms of number of *iterations*.

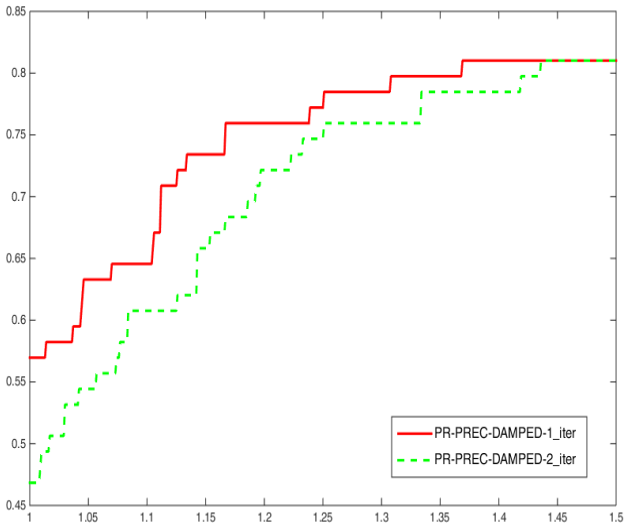


**Figure 4.26.** Comparison between the adoption of the two damped strategies in (4.2) and in (4.9). Detailed profile in terms of number of *iterations*.

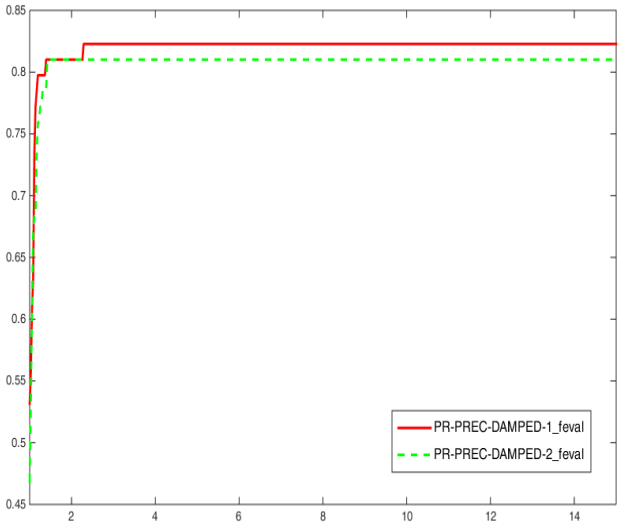


**Figure 4.27.** Comparison between the adoption of the two damped strategies in (4.2) and in (4.9). Profile in terms of number of *function evaluations*.

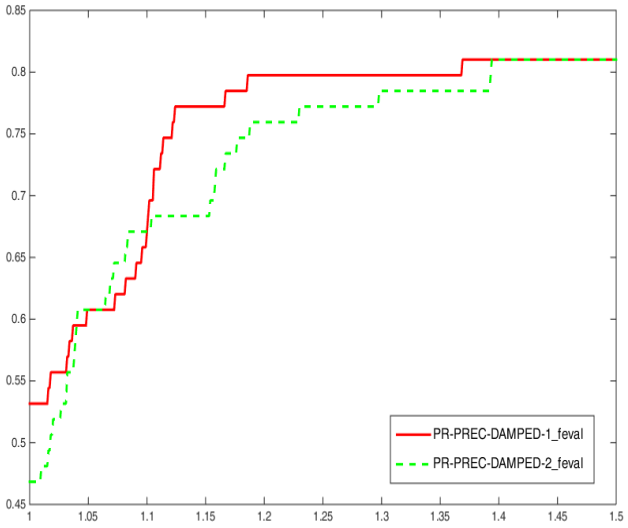


**Figure 4.28.** Comparison between the adoption of the two damped strategies in (4.2) and in (4.9). Detailed profile in terms of number of *function evaluations*.

code, for the sake of correctness.

The results of this comparison, also including the unpreconditioned NCG case, are reported in Figures 4.29-4.30. The results w.r.t. number of iterations show that, to some extent, our first damped strategy in (4.2) can be also competitive with L-BFGS, in terms of efficiency. On the other hand, L-BFGS seems to confirm in any case its robustness, with respect to both a standard (unpreconditioned) NCG scheme and a preconditioned scheme including our damped strategy in (4.2). Finally, on the overall, the results highlight that our first damped strategy in (4.2) provides a good (say efficient) search direction, but still needs a better scaling. In order to confirm this trend, we enclose in Table 4.1 the detailed results (number of iterations (*it*) and number of function evaluations (*nf*)) for those problems where our proposal compares favourably vs. L-BFGS, at least in terms of number of iterations.

Table 4.1 reveals that on several test problems, our approach requires a larger amount of function evaluations w.r.t. L-BFGS, even in presence of a reduced number of iterations. This is due to a couple of facts affecting the Moré-Thuente linesearch procedure [84] used, i.e.:

- the linesearch procedure seems to be well tuned when search directions computed by quasi-Newton methods are adopted, hence the efficiency of L-BFGS;
- observe that, in most of the iterations, the linesearch procedure provides a unit steplength for L-BFGS, while the choice of the stepsize for PNCG is distributed on a larger interval. In Figure 4.31 a typical behaviour of the latter fact is reported (the plot refers to FMINSURF 1024 test problem). This is also motivated by a different selection of the initial stepsize in the linesearch procedure. In particular, a unit initial stepsize is used for L-BFGS, while the Shanno-Phua's formula (default choice for CG+) is adopted as the initial trial

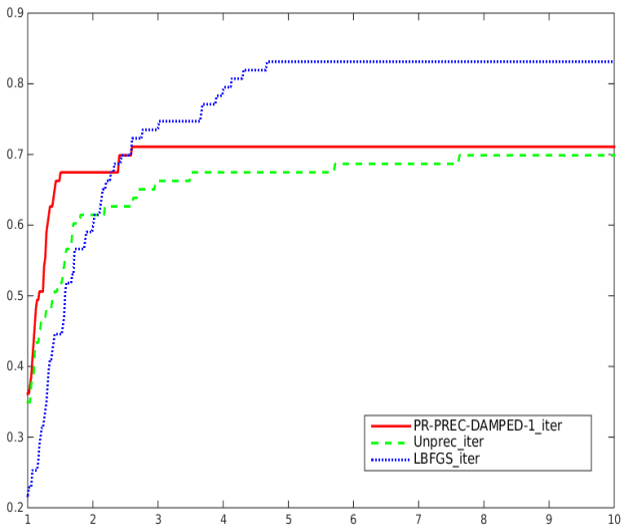


**Figure 4.29.** Comparison among L-BFGS (*dotted line*), our first damped strategy in (4.2) (*solid line*) and the Unpreconditioned NCG method (*dashed line*). Profile in terms of number of *iterations*.

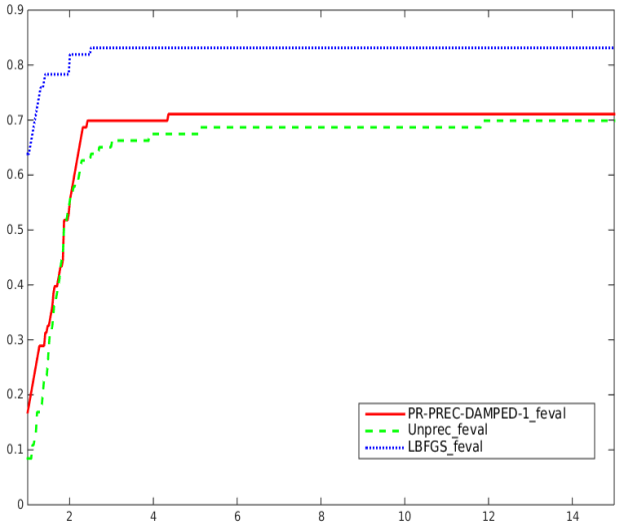


**Figure 4.30.** Comparison among L-BFGS (*dotted line*), our first damped strategy in (4.2) (*solid line*) and the Unpreconditioned NCG method (*dashed line*). Profile in terms of number of *function evaluations*.

| | | *Damped PNCG* | | *L-BFGS* | |
|---|---|---|---|---|---|
| *PROBLEM* | $n$ | *it* | *nf* | *it* | *nf* |
| ARWHEAD | 1000 | 4 | 14 | 11 | 13 |
| ARWHEAD | 10000 | 4 | 13 | 11 | 14 |
| BDQRTIC | 1000 | 315 | 685 | - | - |
| BRYBND | 10000 | 24 | 57 | 41 | 50 |
| CRAGGLVY | 1000 | 69 | 144 | 69 | 79 |
| DIXMAANA | 1500 | 8 | 24 | 11 | 13 |
| DIXMAANA | 3000 | 7 | 22 | 12 | 14 |
| DIXMAANB | 1500 | 7 | 24 | 11 | 13 |
| DIXMAANB | 3000 | 9 | 30 | 11 | 13 |
| DIXMAANC | 1500 | 7 | 26 | 12 | 14 |
| DIXMAANC | 3000 | 9 | 29 | 2 | 14 |
| DIXMAAND | 1500 | 8 | 27 | 5 | 17 |
| DIXMAAND | 3000 | 10 | 32 | 4 | 16 |
| DIXMAANF | 1500 | 90 | 163 | 181 | 190 |
| DIXMAANF | 3000 | 126 | 219 | 236 | 241 |
| DIXMAANG | 3000 | 119 | 214 | 226 | 236 |
| DIXMAANH | 3000 | 98 | 180 | 223 | 229 |
| DQDRTIC | 10000 | 12 | 28 | 13 | 21 |
| EDENSCH | 1000 | 23 | 64 | 25 | 29 |
| EDENSCH | 10000 | 20 | 63 | 25 | 31 |
| FLETCBV3 | 1000 | 262 | 297 | 370 | 384 |
| FMINSURF | 1024 | 170 | 280 | 223 | 224 |
| FMINSURF | 5625 | 468 | 673 | 614 | 632 |
| GENHUMPS | 10000 | 974 | 2344 | - | - |
| LIARWHD | 1000 | 12 | 32 | 20 | 25 |
| MOREBV | 1000 | 10 | 18 | 43 | 45 |
| MOREBV | 10000 | 2 | 4 | 6 | 8 |
| MSQRTBLS | 1024 | 1499 | 2649 | 1811 | 1874 |
| PENALTY1 | 10000 | 18 | 83 | 70 | 84 |
| SCHMVETT | 1000 | 33 | 66 | 39 | 45 |
| SCHMVETT | 10000 | 29 | 64 | 34 | 41 |
| SINQUAD | 1000 | 21 | 61 | 26 | 38 |
| SPARSINE | 1000 | 2592 | 4583 | 6029 | 6307 |
| SPARSQUR | 1000 | 21 | 64 | 27 | 28 |
| SPARSQUR | 10000 | 27 | 84 | 34 | 39 |
| TESTQUAD | 1000 | 2680 | 4774 | 4081 | 4222 |
| TOINTGSS | 1000 | 3 | 17 | 14 | 20 |
| TOINTGSS | 10000 | 4 | 22 | 16 | 23 |
| TQUARTIC | 1000 | 10 | 32 | 21 | 27 |
| VAREIGVL | 1000 | 44 | 90 | 171 | 179 |
| WOODS | 1000 | 26 | 63 | 95 | 125 |

**Table 4.1.** Detailed results for those problems where our first damped strategy in (4.2) compares favourably vs. L-BFGS.

step for our first damped strategy in (4.2).

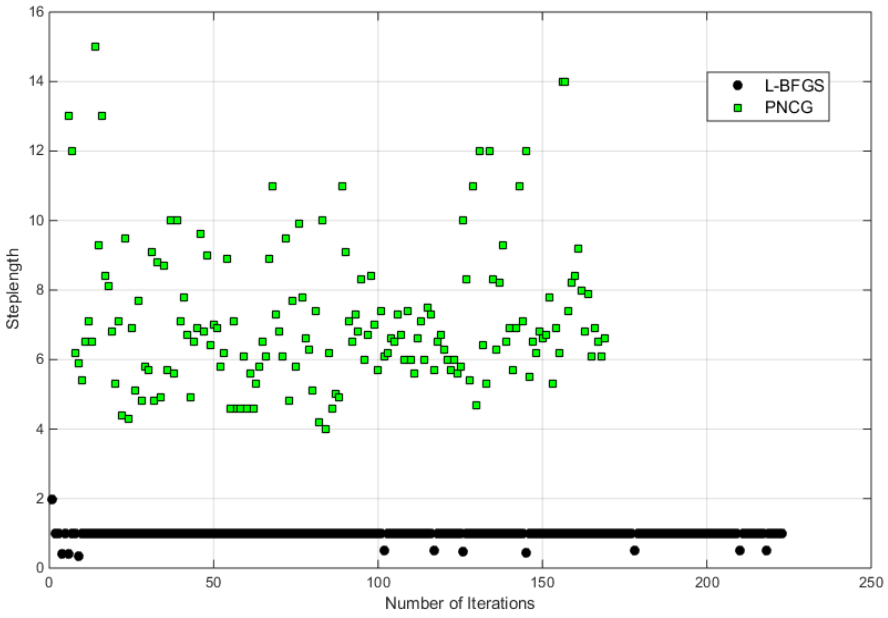


**Figure 4.31.** The complete sequences of steplengths generated by the linesearch procedure, when coupled to L-BFGS (*filled circles*) and to our first damped strategy in (4.2) (*empty squares*).

On the base of the above considerations, it might be the case to investigate modifications to the linesearch procedure, to be paired with our proposal (see for instance [68]). As regards L-CG_DESCENT, the most recent version available in the W. Hager's web page is the L-CG_DESCENT 6.8 code. It is written in $C$, uses an hybrid version of $\beta_k$ coefficient and a different linesearch expressly designed by the authors (see [65]), more efficient and accurate than the Moré-Thuente one. At present, this possibly makes unfair any comparison between our codes and L-CG_DESCENT. Anyway, embedding our preconditioner in L-CG_DESCENT 6.8 would be an interesting further numerical experiment.

It is also worth to highlight that from the detailed complete numerical results we obtained (not all reported in this thesis), as expected the damped strategy occurs in few cases. In particular, when it takes place it enhances either the robustness or the efficiency of the algorithm. In other words, in the case of test problems without "pathologies", correctly the damped strategy is not invoked by the adaptive rule.

On the overall, the results of the numerical experiences reported indicate that the use of a damped strategy can definitely improve the performance of the PR algorithm, at least on the CUTEst problems considered.

## 4.4 Conclusions

In this chapter we proposed the introduction of damped techniques within the framework of the NCG methods. We drew our inspiration from the damped quasi-Newton methods proposed by Al-Baali and Powell. In particular, by referring to the PR method, we investigated the use of a damped vector in the unmodified preconditioned NCG method. On this purpose, we used the damped strategy for constructing a preconditioner based on quasi-Newton updates to be used in the PNCG method. The results obtained clearly highlighted the potentialities of this approach.

Of course several other aspects of interest on damped PNCG were not treated in this study. They range from (but are not limited to) the use of damped techniques to possibility enhance some global convergence properties of the NCG methods, to their more sophisticated use in the construction of a preconditioner (for instance, by introducing a dependence on the iteration $k$ of the parameter $\sigma = \sigma_k$ and a dependence of $\sigma_k$ and $\eta_k$ on $\|g_k\|$ or the number of iterations). Considering self-scaling quasi-Newton methods, it might be also useful to consider the choice of $\bar{\eta}_k = \dfrac{s_k^T B_k s_k}{s_k^T y_k} = \dfrac{-\alpha_k s_k^T g_k}{s_k^T y_k}$ and to use $\eta_k = \max(\bar{\eta}_k, 2)$ in the numerical experiences. Moreover, the combined use of damped strategies with other linesearch procedures (different from the standard Wolfe method) is surely of great interest, too. Finally, adopting the test (4.5) in place of the one in (4.4) can be a possible alternative to explore, in order to improve performance.

# Chapter 5

# Global convergence for Preconditioned Polak-Ribière method

In this chapter we investigate how the use of preconditioning techniques does not prejudice the global convergence of a particular NCG scheme. In Section 5.1 we consider a theoretical analysis, where preconditioning is embedded in a strong convergence framework of an NCG method from the literature. Mild conditions to be satisfied by the preconditioners are defined, in order to preserve NCG convergence. In Section 5.2 we study the global convergence of one *modified preconditioned* method, the damped Polak-Ribière method (D-PR-PNCG) (see Chapter 4).

On this guideline, we are going to report some theoretical results. Most of the material of this chapter is contained in [4], [5], [33].

## 5.1 Global convergence for an effective PNCG method

Several preconditioned NCG schemes were proposed in the literature, with a strong focus on efficiency (see e.g. [101]). The latter schemes also include algorithms where the Nonlinear Conjugate Gradient method is often integrated with an *ad hoc* preconditioner, and is coupled with a linesearch procedure to compute the steplength $\alpha_k$. An example of recent methods where this approach was used is given by CG-DESCENT and L-CG-DESCENT algorithms [65, 66, 68], which include a proper linesearch procedure, along with a formula for the parameter $\beta_k$ specifically defined, to ensure both global convergence and efficiency of the overall algorithm.

In this section we aim at using a globally convergent NCG scheme from the literature, endowed with strong convergence properties, and studying how embedding a positive definite preconditioner in order to preserve the global convergence. This approach on one hand provides an overall preconditioned NCG scheme with strong convergence properties; on the other hand, it gives general clear guidelines to build fruitful preconditioners.

On this purpose, we selected the NCG scheme in [62], since the authors prove rather strong and appealing convergence results for it (further results can also be found in [2] and [6]). We remark that the proposal in [62] (here addressed as PR-

NCG since Polak and Ribière method is considered) also adopts a simple linesearch procedure (which devises results from the literature of derivative-free optimization methods), whose implementation is, to some extent, simpler than the use of standard Wolfe conditions. Then, in this section we show how introducing a very general preconditioner in PR-NCG, still maintaining its global convergence.

In order to prove our theoretical results, we need the following (standard) assumption.

**Assumption 5.1.1** (**see also [62]**).

*a) Given the vector* $x_1 \in \mathbb{R}^n$ *and the function* $f \in C^1(\mathbb{R}^n)$*, the level set* $\mathcal{L}_1 = \{x \in \mathbb{R}^n : f(x) \leq f_1\}$ *is compact.*

*b) There exists an open ball* $\mathcal{B}_r := \{x \in \mathbb{R}^n : \|x\| < r\}$ *containing* $\mathcal{L}_1$ *where* $f(x)$ *is continuously differentiable and its gradient* $g(x)$ *is Lipschitz continuous. In particular, there exists* $L > 0$ *such that*

$$\|g(x) - g(y)\| \leq L\|x - y\| \qquad \text{for all} \qquad x, y \in \mathcal{B}_r. \tag{5.1}$$

*c) There exist* $\delta_1 > 0$, $\delta_2 > 0$ *such that the preconditioner* $M(x)$*, for any* $x \in \mathcal{B}_r$*, is positive definite with eigenvalues satisfying*

$$0 < \delta_1 < \lambda_m[M(x)] \leq \lambda_M[M(x)] < \delta_2.$$

Note that by Assumption 5.1.1 there exists a value $\Omega$ (say $\Omega \geq 1$) such that

$$\|g(x)\| \leq \Omega, \qquad \text{for all} \qquad x \in \mathcal{L}_1. \tag{5.2}$$

Moreover, due to technical reasons we assume that the radius $r$ of $\mathcal{B}_r$ is large enough to satisfy relation

$$r > \sup_{x \in \mathcal{L}_1} \|x\| + \frac{\rho_2}{\sigma}\Omega, \tag{5.3}$$

where $\sigma \in (0, 1)$ and $\rho_2 > 0$ (being the latter parameters used in the next Algorithm PR-NCG_M).

Now we report the algorithm PR-NCG_M, which represents our preconditioned version of the algorithm PR-NCG in [62]. Then, we are going to prove that, under Assumption 5.1.1, PR-NCG_M maintains the same global convergence properties of PR-NCG. For the sake of simplicity we indicate $M(x_k)$ with $M_k$, being $x_k \in \mathcal{B}_r$.

**PR-NCG_M algorithm**

***Data***: Choose $\rho_2 > \rho_1 > 0$, $\gamma > 0$, $\sigma \in (0,1)$. Set $k = 1$, $x_1 \in \mathbb{R}^n$.
For any $k \geq 1$ choose $M_k$ such that $0 < \delta_1 < \lambda_m(M_k) \leq \lambda_M(M_k) < \delta_2$.

***Step 0***: Set $p_1 = -M_1 g_1$.

***Step 1***: If $g_k = 0$ STOP.

***Step 2***: Set $\tau_k = \dfrac{|g_k^T p_k|}{\|p_k\|^2}$ and choose $\Delta_k \in [\rho_1 \tau_k, \rho_2 \tau_k]$.

***Step 3***: Compute $\alpha_k = \max\{\sigma^j \Delta_k, j = 0, 1, \ldots\}$ such that the vectors
$x_{k+1} = x_k + \alpha_k p_k$ and $p_{k+1} = -M_{k+1} g_{k+1} + \beta_{k+1} p_k$,
with $\beta_{k+1} = \dfrac{(g_{k+1} - g_k)^T M_{k+1} g_{k+1}}{g_k^T M_k g_k}$, satisfy the conditions:
***(C1)*** $f_{k+1} \leq f_k - \gamma \alpha_k^2 \|p_k\|^2$
***(C2)*** $-\delta_2 \|g_{k+1}\|^2 \leq g_{k+1}^T p_{k+1} \leq -\delta_1 \|g_{k+1}\|^2$.
Set $k = k + 1$ and go to *Step 1*.

**Remark 5.1.2.** Observe that here the parameters $\delta_1$ and $\delta_2$ do not have to satisfy the condition $\delta_1 < 1 < \delta_2$ as in [62]. This additional generality of our proposal relies on the freedom to choose the preconditioner. In this regard, since $\delta_1$ and $\delta_2$ are no more related to the unit value, no significant bound on the condition number of the preconditioner is imposed (see *(c)* of Assumption 5.1.1).

Now, we prove first that conditions ***(C1)*** and ***(C2)*** and the entire *Step 3* of PR-NCG_M are well defined.

**Proposition 5.1.3.** *Suppose that $g_k \neq 0$ and $\|p_k\| < +\infty$ for all $k$. Let Assumption 5.1.1 hold. Then, for every $k \geq 1$ there exists a finite index $j_k$ such that $\alpha_k = \sigma^{j_k} \Delta_k$ satisfies conditions* **(C1)** *and* **(C2)** *at* Step 3.

*Proof.* First observe that as $g_1^T p_1 < 0$, using ***(C2)*** we have by induction that $g_k^T p_k < 0$ for all $k$. Now, by assuming $g_k^T p_k < 0$, we prove that the number $\alpha_k = \Delta_k \sigma^j$ satisfies conditions ***(C1)*** and ***(C2)*** for all sufficiently large $j$.

By contradiction, assume first that there exists an infinite set $\mathcal{J}$ of the index $j$ such that condition ***(C1)*** is violated, i.e. for every $j \in \mathcal{J}$ we have:

$$\frac{f(y^{(j)}) - f_k}{\sigma^j \Delta_k} > -\gamma \sigma^j \Delta_k \|p_k\|^2,$$

where $y^{(j)} := x_k + \sigma^j \Delta_k p_k$. Then, taking the limit for $j \in \mathcal{J}, j \longrightarrow +\infty$, we have $g_k^T p_k \geq 0$ which contradicts the assumption $g_k^T p_k < 0$. Suppose now that by

contradiction there exists an infinite set, say it again $\mathcal{J}$, such that for $j \in \mathcal{J}$ condition ***(C2)*** is violated. This implies that by the boundedness of $\|p_k\|$, for all $j \in \mathcal{J}$ at least one of the following conditions holds (the subscript in $M_j$ denotes the fact that the preconditioner possibly depends on the point $y^{(j)}$):

$$g(y^{(j)})^T \left( -M_j g(y^{(j)}) + \frac{(g(y^{(j)}) - g_k)^T M_j g(y^{(j)})}{g_k^T M_k g_k} p_k \right) > -\delta_1 \|g(y^{(j)})\|^2,$$

$$g(y^{(j)})^T \left( -M_j g(y^{(j)}) + \frac{(g(y^{(j)}) - g_k)^T M_j g(y^{(j)})}{g_k^T M_k g_k} p_k \right) < -\delta_2 \|g(y^{(j)})\|^2.$$

Then, taking limits for $j \in \mathcal{J}, j \longrightarrow +\infty$, we obtain that at least one of the two inequalities $-g_k^T M_k g_k \geq -\delta_1 \|g_k\|^2$ and $-g_k^T M_k g_k \leq -\delta_2 \|g_k\|^2$ must be valid. But in both cases we get a contradiction to the assumptions $g_k \neq 0$ and $0 < \delta_1 < \lambda_m(M_k) \leq \lambda_M(M_k) < \delta_2$.

Therefore, under the assumption $g_k^T p_k < 0$, we can conclude that *Step 3* is well defined, by taking $j_k$ as the largest index for which both conditions ***(C1)*** and ***(C2)*** are satisfied, and setting the parameters as $\alpha_k = \sigma^{j_k} \Delta_k$. □

The main properties of the sequence of iterates produced by Algorithm PR-NCG_M, which are at the basis of our convergence result, are stated in the next proposition.

**Proposition 5.1.4.** *Let Assumption 5.1.1 hold. Suppose that $g_k \neq 0$ and $\|p_k\| < \infty$ for all $k \geq 1$. Then we have:*

*(i)* $x_k \in \mathcal{L}_1$ *for all* $k \geq 1$;

*(ii) the sequence* $\{f_k\}$ *has a limit;*

*(iii)* $\lim_{k \to +\infty} \alpha_k \|p_k\| = 0$;

*(iv)* $\alpha_k \|p_k\|^2 \leq \rho_2 \delta_2 \Omega^2$, *for all* $k \geq 1$;

*(v) for every* $k$ *there exists a positive number* $\theta$ *such that* $\alpha_k \geq \theta \dfrac{|g_k^T p_k|}{\|p_k\|^2}$.

*Proof.* Condition ***(C1)*** at *Step 3* implies

$$f_k - f_{k+1} \geq \gamma \alpha_k^2 \|p_k\|^2. \tag{5.4}$$

From (5.4) we have $x_k \in \mathcal{L}_1$ for all $k$, which establishes $(i)$; then $(ii)$ follows from (5.4) and the compactness of $\mathcal{L}_1$. Recalling the expression of $\alpha_k$ and taking limits in (5.4) for $j \longrightarrow +\infty$, then $(ii)$ yields $(iii)$. Now, the instructions at *Step 2* and *Step 3* imply

$$\alpha_k \|p_k\|^2 \leq \Delta_k \|p_k\|^2 \leq \rho_2 |g_k^T p_k| \leq \rho_2 \delta_2 \|g_k\|^2 \leq \rho_2 \delta_2 \Omega^2,$$

so that we get $(iv)$. In order to establish $(v)$ we distinguish the two cases: $\alpha_k = \Delta_k$ and $\alpha_k < \Delta_k$, where $\Delta_k$ is the scalar defined at *Step 2*. In the first case, we have obviously

$$\alpha_k \geq \rho_1 \frac{|g_k^T p_k|}{\|p_k\|^2}. \tag{5.5}$$

Now suppose that $\alpha_k < \Delta_k$, so that $\frac{\alpha_k}{\sigma}$ violates at least one of the conditions at *Step 3*. Recalling (5.3) we have that the point

$$w_k := x_k + \frac{\alpha_k}{\sigma} p_k$$

belongs to the ball $\mathcal{B}_r$ introduced in Assumption 5.1.1, being

$$\left\| \frac{\alpha_k}{\sigma} p_k \right\| \leq \rho_2 \frac{|g_k^T p_k|}{\|p_k\|^2} \frac{\|p_k\|}{\sigma} \leq \frac{\rho_2}{\sigma} \Omega.$$

If ***(C1)*** is violated we can write, using the Mean Value Theorem:

$$f_k + \frac{\alpha_k}{\sigma} g_k^T p_k + \frac{\alpha_k}{\sigma} \left( g(z_k)^T p_k - g_k^T p_k \right) > f_k - \gamma \left( \frac{\alpha_k}{\sigma} \right)^2 \|p_k\|^2, \tag{5.6}$$

where $z_k := x_k + \eta_k \frac{\alpha_k}{\sigma} p_k \in \mathcal{B}_r$ with $\eta_k \in (0,1)$. Recalling (5.1) and by the Cauchy-Schwarz inequality, we get

$$\begin{aligned} g(z_k)^T p_k - g_k^T p_k &\leq |g(z_k)^T p_k - g_k^T p_k| \leq \|g(z_k) - g_k\| \|p_k\| \\ &\leq L \|z_k - x_k\| \|p_k\| \leq L \frac{\alpha_k}{\sigma} \|p_k\|^2. \end{aligned} \tag{5.7}$$

Using (5.7) in (5.6) we get

$$\frac{\alpha_k}{\sigma} g_k^T p_k + \left( \frac{\alpha_k}{\sigma} \right)^2 L \|p_k\|^2 \geq -\gamma \left( \frac{\alpha_k}{\sigma} \right)^2 \|p_k\|^2,$$

whence we obtain

$$\alpha_k \geq \frac{\sigma}{L + \gamma} \frac{|g_k^T p_k|}{\|p_k\|^2}, \tag{5.8}$$

which proves $(v)$. Now assume that $\frac{\alpha_k}{\sigma}$ violates ***(C2)***. Suppose first that the rightmost inequality does not hold, i.e.

$$g(w_k)^T \left( -M_j g(w_k) + \frac{(g(w_k) - g_k)^T M_j g(w_k)}{g_k^T M_k g_k} p_k \right) > -\delta_1 \|g(w_k)\|^2. \tag{5.9}$$

Recalling (5.1) and by the Cauchy-Schwarz inequality we have

$$\begin{aligned} (g(w_k) - g_k)^T M_j g(w_k) &\leq |(g(w_k) - g_k)^T M_j g(w_k)| \leq \|g(w_k) - g_k\| \|M_j g(w_k)\| \\ &\leq L \|w_k - x_k\| \|M_j g(w_k)\| \\ &\leq L \frac{\alpha_k}{\sigma} \lambda_M(M_j) \|g(w_k)\| \|p_k\|. \end{aligned} \tag{5.10}$$

Furthermore we get

$$g(w_k)^T M_j g(w_k) \geq \lambda_m(M_j) \|g(w_k)\|^2; \tag{5.11}$$

thus, using (5.10) and (5.11) in (5.9) we obtain:

$$L\frac{\alpha_k}{\sigma}\lambda_M(M_j)\|g(w_k)\|\|p_k\|g(w_k)^T p_k > (\lambda_m(M_j)-\delta_1)\|g(w_k)\|^2 g_k^T M_k g_k. \tag{5.12}$$

Again, by the Cauchy-Schwarz inequality we have

$$\|g(w_k)\|\|p_k\|g(w_k)^T p_k \leq \|g(w_k)\|^2\|p_k\|^2 \tag{5.13}$$

along with

$$g_k^T M_k g_k \geq \lambda_m(M_k)\|g_k\|^2. \tag{5.14}$$

Finally, using (5.13) and (5.14) in (5.12) we obtain

$$L\frac{\alpha_k}{\sigma}\lambda_M(M_j)\|g(w_k)\|^2\|p_k\|^2 > (\lambda_m(M_j)-\delta_1)\|g(w_k)\|^2\lambda_m(M_k)\|g_k\|^2. \tag{5.15}$$

Taking the limits for $j \longrightarrow +\infty$ and taking into account that the rightmost inequality ***(C2)*** holds at Step $k$, we get

$$\begin{aligned}\alpha_k &\geq \frac{\sigma(\lambda_m(M_k)-\delta_1)}{L\lambda_M(M_k)}\lambda_m(M_k)\frac{\|g_k\|^2}{\|p_k\|^2} \geq \frac{\sigma(\lambda_m(M_k)-\delta_1)}{L\delta_2}\lambda_m(M_k)\frac{\|g_k\|^2}{\|p_k\|^2} \\ &\geq \frac{\sigma(\lambda_m(M_k)-\delta_1)}{\delta_2^2 L}\lambda_m(M_k)\frac{|g_k^T p_k|}{\|p_k\|^2},\end{aligned} \tag{5.16}$$

implying *(v)*. Now suppose that ***(C2)*** is violated because the leftmost inequality is not fulfilled, i.e.

$$g(w_k)^T\left(-M_j g(w_k) + \frac{(g(w_k)-g_k)^T M_j g(w_k)}{g_k^T M_k g_k}p_k\right) < -\delta_2\|g(w_k)\|^2. \tag{5.17}$$

Using a similar reasoning we obtain

$$\begin{aligned}\alpha_k &\geq \frac{\sigma(\delta_2-\lambda_M(M_k))}{L\lambda_M(M_k)}\lambda_m(M_k)\frac{\|g_k\|^2}{\|p_k\|^2} \geq \frac{\sigma(\delta_2-\lambda_M(M_k))}{L\delta_2}\lambda_m(M_k)\frac{\|g_k\|^2}{\|p_k\|^2} \\ &\geq \frac{\sigma(\delta_2-\lambda_M(M_k))}{\delta_2^2 L}\lambda_m(M_k)\frac{|g_k^T p_k|}{\|p_k\|^2}.\end{aligned} \tag{5.18}$$

Thus, from (5.5), (5.8), (5.16) and (5.18) we obtain $(v)$ by taking

$$\theta = \min\left\{\rho_1, \frac{\sigma}{L+\gamma}, \frac{\sigma(\lambda_m(M_k)-\delta_1)}{\delta_2^2 L}\lambda_m(M_k), \frac{\sigma(\delta_2-\lambda_M(M_k))}{\delta_2^2 L}\lambda_m(M_k)\right\}.$$

□

Finally, in order to establish the main global convergence properties of Algorithm PR-NCG_M we can state the following theorem.

**Theorem 5.1.5.** *Let $\{x_k\}$ be the sequence of points generated by Algorithm PR-NCG_M. Then, either there exists an index $\nu \geq 1$ such that $g(x_\nu) = 0$ and the algorithm terminates, or $\{x_k\}$ is an infinite sequence such that:*

*(i)* $\lim_{k\to+\infty} \|g(x_k)\| = 0$;

*(ii) every limit point of* $\{x_k\}$ *is a stationary point of* $f$.

*Proof.* Suppose that Algorithm PR-NCG_M does not terminate in a finite number of iterations and that $(i)$ is false. Then, there exists a subsequence $\{x_k\}_{k\in K} \subseteq \{x_k\}$ and $\varepsilon > 0$ such that

$$\|g_{k-1}\| \geq \varepsilon, \qquad \text{for all} \qquad k \in K, \tag{5.19}$$

and by $(iii)$ of Proposition 5.1.4

$$\lim_{k\to+\infty,\ k\in K} \|x_k - x_{k-1}\| = 0.$$

Now, by the instructions of Algorithm PR-NCG_M, using (5.1), (5.2), (5.19) and $(iv)$ of Proposition 5.1.4 we can write for $k \in K$:

$$\begin{aligned}
\|p_k\| &\leq \|M_k g_k\| + \left(\frac{\|g_k - g_{k-1}\|\|M_k g_k\|}{\|g_{k-1}\|\|M_{k-1} g_{k-1}\|}\|p_{k-1}\|\right) \\
&\leq \lambda_M(M_k)\Omega + \left(\frac{L\|x_k - x_{k-1}\|\lambda_M(M_k)\Omega}{\varepsilon^2\lambda_m(M_{k-1})}\|p_{k-1}\|\right) \\
&= \lambda_M(M_k)\Omega + \left(\frac{L\|\alpha_{k-1}p_{k-1}\|\lambda_M(M_k)\Omega}{\varepsilon^2\lambda_m(M_{k-1})}\|p_{k-1}\|\right) \\
&= \lambda_M(M_k)\Omega + \left(\frac{\lambda_M(M_k)\Omega L\alpha_{k-1}\|p_{k-1}\|^2}{\varepsilon^2\lambda_m(M_{k-1})}\right) \\
&\leq \lambda_M(M_k)\Omega + \left(\frac{\lambda_M(M_k)}{\lambda_m(M_{k-1})}\right)\left(\frac{\Omega^3 L\rho_2\delta_2}{\varepsilon^2}\right).
\end{aligned} \tag{5.20}$$

Therefore, using $(iii)$ of Proposition 5.1.4 we get

$$\lim_{k\to+\infty,\ k\in K} \alpha_k\|p_k\|^2 = 0,$$

and hence by $(v)$ of Proposition 5.1.4 it follows

$$\lim_{k\to+\infty,\ k\in K} |g_k^T p_k| = 0.$$

The latter condition implies, by ***(C2)*** of *Step 3* in algorithm PR-NCG_M

$$\lim_{k\to+\infty,\ k\in K} \|g_k\| = 0,$$

so that $(iii)$ of Proposition 5.1.4 and the Lipschitz continuity of $g$ contradict (5.19). Thus, $(i)$ holds and $(ii)$ follows from the continuity of $f(x)$. □

This theorem shows that for the preconditioned version PR-NCG_M the same global convergence properties of PR-NCG still hold.

## 5.2 Convergence properties for preconditioned damped Polak-Ribière (D-PR-PNCG) method

As already recalled, in paper [3] the author extends global convergence properties of the Broyden family of quasi-Newton methods to the damped version of such methods. In a similar fashion, we aim at proving that some global convergence properties of NCG methods still hold in the general case corresponding to the damped and preconditioned version (*modified* PNCG method). Obviously, results for undamped and/or unmodified methods are straightforwardly obtained as particular cases.

As first step of the convergence analysis, in this section our preliminary focus is on the Polak-Ribière (PR) version of the NCG. In particular, we limit our analysis to consider only the first proposal in (4.2), namely

$$\widehat{y}_k^{(1)} = \varphi_k y_k + (1-\varphi_k)\eta_k s_k,$$

Note that in this regard, developing convergence properties with the choice $\widehat{y}_k^{(2)}$ needs additional analysis, which is not part of this thesis.

Using the damped vector $\widehat{y}_k^{(1)}$ we therefore consider the damped preconditioned PR method (namely D-PR-PNCG method):

$$\hat{\beta}_{k+1}^{\mathrm{PR}} = \frac{(\widehat{y}_k^{(1)})^T M_{k+1} g_{k+1}}{g_k^T M_k g_k}. \tag{5.21}$$

The resulting D-PR-PNCG method actually is a novel *modified* NCG method. Hence the necessity of ensuring its global convergence properties. To this aim, in this section, using Assumption 5.1.1, we prove that, to some extent, the D-PR-PNCG method enjoys the same properties as the standard (undamped and unpreconditioned) PR method (see e.g. [63]). In order to prove our final results, we firstly report a technical result (see [63] and [98]).

**Lemma 5.2.1.** *Let $\{\xi_k\}$ be a sequence of nonnegative real numbers. Let $\Omega > 0$ and $q \in (0,1)$ and suppose that there exists $k_1 \geq 1$ such that*

$$\xi_k \leq \Omega + q\xi_{k-1}, \qquad \textit{for any} \qquad k \geq k_1.$$

*Then,*

$$\xi_k \leq \frac{\Omega}{1-q} + \left(\xi_{k_1} - \frac{\Omega}{1-q}\right) q^{k-k_1}, \qquad \textit{for any} \qquad k \geq k_1.$$

*Proof.* Starting from relation

$$\xi_k \leq \Gamma + q\xi_{k-1}, \qquad \text{for any} \qquad k \geq k_1,$$

considering $k - k_1$ iterations we have

$$\xi_k \leq \Gamma \left( \sum_{i=0}^{k-k_1-1} q^i \right) + q^{k-k_1}\xi_{k_1},$$

and since

$$\sum_{i=0}^{k-k_1-1} q^i = \sum_{i=0}^{+\infty} q^i - \sum_{i=k-k_1}^{+\infty} q^i$$

we get

$$\xi_k \leq \Gamma \left( \sum_{i=0}^{+\infty} q^i - \sum_{i=k-k_1}^{+\infty} q^i \right) + q^{k-k_1} \xi_{k_1}.$$

Recalling that

$$\sum_{i=0}^{+\infty} q^i = \frac{1}{1-q}$$

and

$$\sum_{i=k-k_1}^{+\infty} q^i = q^{k-k_1} \sum_{i=0}^{+\infty} q^i = q^{k-k_1} \left( \frac{1}{1-q} \right),$$

we finally obtain

$$\xi_k \leq \Gamma \left( \frac{1}{1-q} - q^{k-k_1} \left( \frac{1}{1-q} \right) \right) + q^{k-k_1} \xi_{k_1} = \frac{\Gamma}{1-q} + \left( \xi_{k_1} - \frac{\Gamma}{1-q} \right) q^{k-k_1}.$$

□

Now, we can introduce the following result.

**Theorem 5.2.2.** *Let $\{x_k\}$ be an infinite sequence (with $g_k \neq 0$) generated by the D-PR-PNCG method, where the steplength $\alpha_k > 0$ is determined by a linesearch procedure such that, for all $k$, the following conditions hold:*

**(i)** *$x_k \in \mathcal{L}_1$ for all $k$;*

**(ii)** $\lim_{k \to +\infty} \frac{|g_k^T p_k|}{\|p_k\|} = 0$;

**(iii)** $\lim_{k \to +\infty} \alpha_k \|p_k\| = 0$.

*If Assumption 5.1.1 holds, then*

$$\liminf_{k \to +\infty} \|g_k\| = 0$$

*and hence there exists at least a stationary limit point of $\{x_k\}$.*

*Proof.* First observe that by the Lipschitz continuity of $g(x)$ and the compactness of $\mathcal{L}_1$, there exists a number $\Gamma > 0$ such that

$$\|g(x)\| \leq \Gamma, \qquad \text{for all} \qquad x \in \mathcal{L}_1. \tag{5.22}$$

Moreover, from *(i)* and the compactness of $\mathcal{L}_1$, the sequence $\{x_k\}$ admits limit points in $\mathcal{L}_1$. Now, by contradiction, assume that there exist $\varepsilon > 0$ and $\bar{k}$ such that

$$\|g_k\| \geq \varepsilon, \qquad \text{for all} \qquad k > \bar{k}. \tag{5.23}$$

By using (5.21)-(5.23) and *(i)*, and recalling that we are considering D-PR-PNCG, we get for any $k \geq \bar{k}$,

$$\begin{aligned}
\|p_{k+1}\| &= \| - M_{k+1}g_{k+1} + \hat{\beta}^{\mathrm{PR}}_{k+1}p_k\| \\
&= \left\| -M_{k+1}g_{k+1} + \frac{[\varphi_k y_k + (1-\varphi_k)\eta_k s_k]^T M_{k+1}g_{k+1}}{g_k^T M_k g_k} p_k \right\| \\
&\leq \|M_{k+1}g_{k+1}\| + \frac{\|\varphi_k y_k + (1-\varphi_k)\eta_k s_k\|\|M_{k+1}g_{k+1}\|}{\|g_k\|\|M_k g_k\|}\|p_k\| \\
&\leq \Gamma\lambda_M(M_{k+1}) + \Gamma\lambda_M(M_{k+1})\frac{\|\varphi_k y_k + (1-\varphi_k)\eta_k s_k\|}{\varepsilon^2\lambda_m(M_k)}\|p_k\|. \qquad (5.24)
\end{aligned}$$

From (5.24), recalling the Lipschitz continuity of $g(x)$ on $\mathcal{L}_1$, we have

$$\begin{aligned}
\|\varphi_k y_k + (1-\varphi_k)\eta_k s_k\| &= \|\varphi_k(g_{k+1}-g_k) + (1-\varphi_k)\eta_k(x_{k+1}-x_k)\| \\
&\leq \varphi_k L\|x_{k+1}-x_k\| + (1-\varphi_k)\eta_k\|x_{k+1}-x_k\| \\
&= \|\alpha_k p_k\|(\varphi_k L + (1-\varphi_k)\eta_k). \qquad (5.25)
\end{aligned}$$

Hence, by using (5.24) we obtain

$$\|p_{k+1}\| \leq \Gamma\lambda_M(M_{k+1}) + \Gamma\lambda_M(M_{k+1})\left(\frac{\varphi_k L + (1-\varphi_k)\eta_k}{\varepsilon^2\lambda_m(M_k)}\right)\|\alpha_k p_k\|\|p_k\|. \qquad (5.26)$$

Now, by *(iii)*, given $q \in (0,1)$, we can assume there exists $k_1$ sufficiently large such that

$$\Gamma\lambda_M(M_{k+1})\left(\frac{\varphi_k L + (1-\varphi_k)\eta_k}{\varepsilon^2\lambda_m(M_k)}\right)\|\alpha_k p_k\| \leq q < 1, \quad \text{for any} \quad k \geq k_1 > \bar{k}. \qquad (5.27)$$

Thus, by (5.26)-(5.27) we get

$$\|p_{k+1}\| \leq \Gamma\lambda_M(M_{k+1}) + q\|p_k\|, \qquad \text{for any} \qquad k \geq k_1,$$

and by Lemma 5.2.1

$$\|p_{k+1}\| \leq \frac{\Gamma\lambda_M(M_{k+1})}{1-q} + \left(\|p_{k_1}\| - \frac{\Gamma\lambda_M(M_{k+1})}{1-q}\right)q^{(k+1)-k_1}, \qquad \forall k \geq k_1, \qquad (5.28)$$

showing that $\|p_{k+1}\|$ is bounded as $\|p_{k_1}\|$ is bounded. As a consequence, again from *(iii)* we have

$$\lim_{k\to+\infty} \alpha_k\|p_k\|^2 = 0. \qquad (5.29)$$

Furthermore, the boundedness of $\|p_k\|$ and *(ii)* yield

$$\lim_{k\to+\infty} |g_k^T p_k| = 0. \qquad (5.30)$$

Since $M_{k+1}g_{k+1} = \hat{\beta}^{\mathrm{PR}}_{k+1}p_k - p_{k+1}$, by (5.26) it results

$$\begin{aligned} g^T_{k+1}M_{k+1}g_{k+1} &= g^T_{k+1}\hat{\beta}^{\mathrm{PR}}_{k+1}p_k - g^T_{k+1}p_{k+1} \\ &\leq \|g_{k+1}\|\|\hat{\beta}^{\mathrm{PR}}_{k+1}p_k\| + |g^T_{k+1}p_{k+1}| \\ &\leq \frac{\alpha_k\Gamma\lambda_M(M_{k+1})\|g_{k+1}\|\|p_k\|(\varphi_k L + (1-\varphi_k)\eta_k)\|p_k\|}{\|g_k\|\|M_k g_k\|} \\ &\quad + |g^T_{k+1}p_{k+1}| \\ &\leq \frac{\alpha_k\Gamma\lambda_M(M_{k+1})\|g_{k+1}\|\|p_k\|^2(\varphi_k L + (1-\varphi_k)\eta_k)}{\varepsilon^2\lambda_m(M_k)} \\ &\quad + |g^T_{k+1}p_{k+1}|. \end{aligned} \tag{5.31}$$

By (5.29), (5.30) and the compactness of $\mathcal{L}_1$, taking limits in (5.31) as $k \to +\infty$, we obtain

$$\lim_{k\to+\infty} g^T_{k+1}M_{k+1}g_{k+1} = 0.$$

Finally, by (c) of Assumption 5.1.1

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

and this contradicts assumption (5.23). □

### 5.2.1 Numerical experience

On the overall, the results of the numerical experiences reported in Chapter 4 indicate that the use of a damped strategy can definitely improve the performance of the PR algorithm, at least on the CUTEst problems considered.

So far, the damped strategy was experimented in constructing our quasi-Newton based preconditioner, which is our main focus. Now, for the sake of completeness, since the theoretical part in Section 5.2 encompasses the possibility to embed the damped strategy both in the definition of the scalar $\beta_k$ and in the preconditioner, we urge to perform numerical testing also on the use of $\widehat{\beta}^{PR}_k$ in (5.21). In this regard, note that the use of damped strategy was conceived in the context of quasi-Newton updates, and it is not expected to be successfully exploited in the definition of the scalar $\beta_k$ used in a NCG/PNCG method. In the sequel we report results obtained by using the damped vector $\widehat{\beta}^{PR}_k$ confirming this claim. In particular, we first consider the unpreconditioned case and compare the behaviour of the unmodified NCG method with the method which adopts $\widehat{\beta}^{PR}_k$, setting $\widehat{y}_k = \widehat{y}^{(1)}_k$ with the default values of $\sigma = 0.8$ and $\eta_k = 4$. Then, we perform the same comparing in the preconditioned case on the usual selection of test problems from CUTEst collection. Figures 5.1-5.4 report the performance profiles in terms of number of iterations and number of function evaluations corresponding to these comparisons.

As it can be observed from these profiles, the use of the $\widehat{\beta}^{PR}_k$ does not yield a noteworthy improvement neither in terms of iterations or function evaluations. Nevertheless we also observe that the D-PR-PNCG scheme which also uses $\widehat{\beta}^{PR}_k$ reveals to outperform the standard NCG method. Thus, on the overall, the adoption of the damped strategy within PNCG methods seems to be definitely promising.

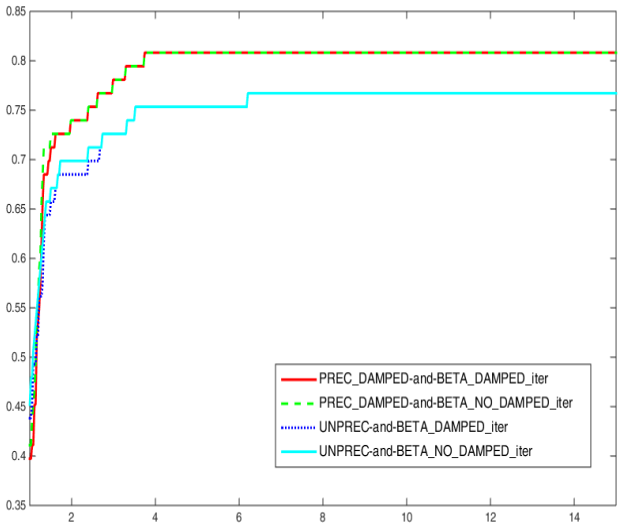


**Figure 5.1.** Comparison between the use $\widehat{\beta}_k^{PR}$ in (5.21) (setting $\widehat{y}_k = \widehat{y}_k^{(1)}$) and $\beta_k^{PR}$ in (2.12), in both preconditioned and unpreconditioned cases. Profile in terms of number of *iterations*.

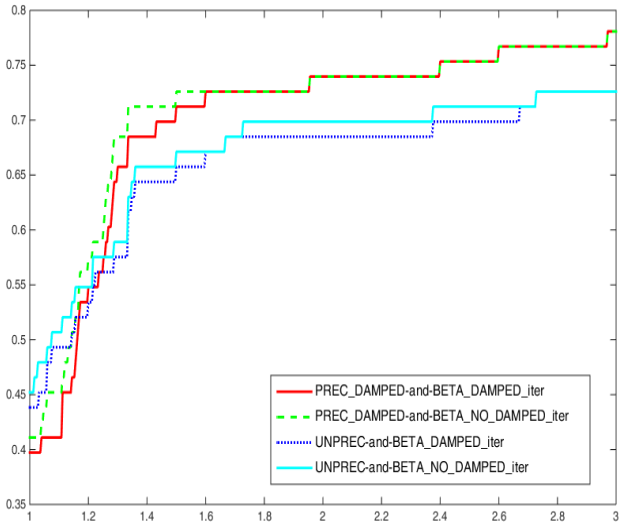


**Figure 5.2.** Comparison between the use $\widehat{\beta}_k^{PR}$ in (5.21) (setting $\widehat{y}_k = \widehat{y}_k^{(1)}$) and $\beta_k^{PR}$ in (2.12), in both preconditioned and unpreconditioned cases. Detailed profile in terms of number of *iterations*.

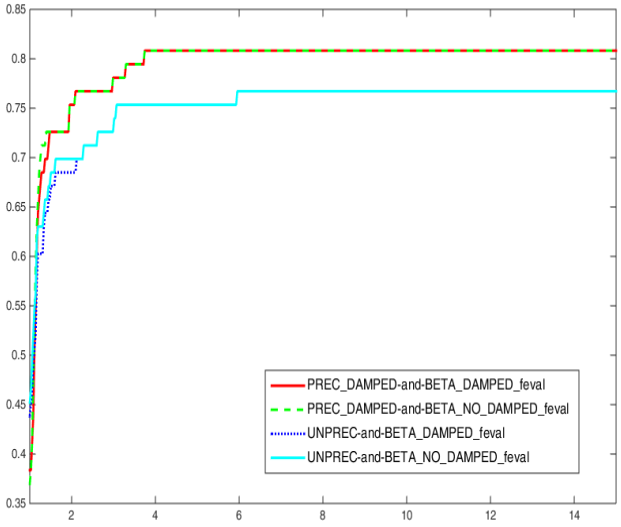


**Figure 5.3.** Comparison between the use $\widehat{\beta}_k^{PR}$ in (5.21) (setting $\widehat{y}_k = \widehat{y}_k^{(1)}$) and $\beta_k^{PR}$ in (2.12), in both preconditioned and unpreconditioned cases. Profile in terms of number of *function evaluations*.

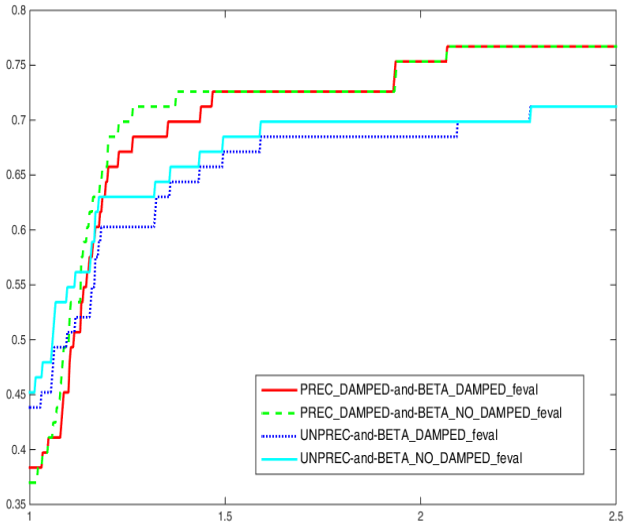


**Figure 5.4.** Comparison between the use $\widehat{\beta}_k^{PR}$ in (5.21) (setting $\widehat{y}_k = \widehat{y}_k^{(1)}$) and $\beta_k^{PR}$ in (2.12), in both preconditioned and unpreconditioned cases. Detailed profile in terms of number of *function evaluations*.

## 5.3 Conclusions

In this chapter we have proposed some conditions in order to preserve the global convergence of a particular NCG schemes, using preconditioning techniques.

# Chapter 6

# An adaptive truncation criterion for Newton-Krylov methods

In this chapter we focus on Newton-Krylov methods for large scale unconstrained optimization problems. In order to increase the efficiency of these methods, the proper choice of a suitable truncation criterion for the inner iterations, along with the necessity to handle the indefinite case and the choice of an effective preconditioning strategy, are three "open questions", listed in the early survey paper [94]. In subsequent years, even if the research activity on these topics was greatly developed, actually no definitive answer has been yet provided.

On this guidelines we focus on the possibility to "enrich" the residual-based criterion (2.75) by conveying, also in this case, information gained from the behaviour of the quadratic model. To this aim, we propose an adaptive rule for deciding the maximum number of inner iterations allowed at each outer iteration. The latter rule combined with the criterion (2.75) should enhance the overall efficiency of a truncated Newton method, by possibly avoiding the over-solving phenomenon. A numerical experience on unconstrained optimization problems highlights a satisfactory effectiveness and robustness of the adaptive criterion proposed, when a residual-based truncation criterion is selected. Most of the material of this chapter is contained in [29], [30].

## 6.1 Introduction

Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a twice continuously differentiable function. In the following, we denote by $H_k = \nabla^2 f(x_k)$ the Hessian matrix of the function $f$ at the point $x_k$.

In this chapter, starting from the paper by Nash and Sofer in [90], we propose a simple adaptive truncation criterion for the inner iterations within a linesearch-based Newton-Krylov method (see the scheme in Section 2.1.3.1), and analyze its effectiveness in both the unpreconditioned and the preconditioned case (preconditioner in [48] will be considered). Our aim is to define an additional rule which enables to avoid "over-solving" of the Newton equation (2.74), that is

$$H_k p_k = -g_k,$$

in some circumstances. The latter phenomenon occurs whenever unnecessary inner

iterations are performed, so that indulging in solving the Newton equation does not produce a better search direction. This possibly yields a reduction of the overall inner iterations, for both convex and nonconvex problems. Our proposal is partially inspired by trust region approach (see e.g. [36]), and is based on a comparison between the reduction of the objective function predicted by the quadratic model, and the actual reduction obtained. In particular, we consider a linesearch-based Newton-Krylov method where the inner iterations are performed using the CG algorithm.

## 6.2 Motivation for the Truncation Rule

Both the stopping criteria (2.75) and (2.77), respectively,

$$\frac{\|r_k\|}{\|g_k\|} \leq \eta_k$$

and

$$\frac{\frac{q_k(p_j) - q_k(p_{j-1})}{q_k(p_j)}}{j} \leq \eta_k$$

(see Section 2.1.3.1.1), may not prevent over-solving of the Newton equation. For the residual-based criterion (2.75), different forcing sequences have been proposed to stem this phenomenon [46], still guaranteeing a good asymptotic rate of convergence.

As regards the criterion (2.77), it is well-grounded if the quadratic model is accurate, i.e. if the quadratic model is a good (local) approximation of the objective function. Conversely, if accuracy is poor, a successful strategy has been proposed in [91] in the context of Newton-Krylov methods. The strategy is simple and consists of allowing only one inner iteration if the stepsize determined by the linesearch procedure in the previous outer iteration is different from one. The rationale behind this strategy relies on the fact that a stepsize different from one (i.e. the search direction likely does not resemble the Newton direction) means that the quadratic model is likely inaccurate. In the context of a block method where each inner iteration represents a significant computation, this simple rule was appropriate and effective. When the standard CG method is used as the inner iteration, a more nuanced strategy is desired.

Additional justification for avoiding over-solving is provided by the computational results in [89]. This paper compares the performance of a Newton-Krylov method and a limited-memory BFGS method, and concludes that the Newton-Krylov method displays superior performance when the quadratic model is a good approximation to the objective function.

Based on this evidence, we propose an adaptive rule for dynamically setting the maximum number of inner iterations at each outer iteration of a linesearch-based Newton-Krylov scheme, possibly allowing the Hessian matrix $H_k$ to be indefinite. In particular, inspired by trust region methods [36] (and borrowing their terminology/notation), at each outer iteration $k$, our idea is to compare the *actual reduction* of the objective function

$$Ared_k = f_k - f(x_k + s_k)$$

with the *predicted reduction*, i.e. the reduction predicted by the quadratic model

$$Pred_k = q_k(0) - q_k(s_k) = -\left[\frac{1}{2}s_k^T H_k s_k + g_k^T s_k\right],$$

where $s_k = \alpha_k p_k$ and $\alpha_k$ is the stepsize computed by the linesearch procedure. Our truncation criterion will be based on the difference between actual and predicted reduction, an estimate of the difference between the quadratic model and the objective function. It is this quantity that was determined in [89] to be significant to the performance of a Newton-Krylov method.

To provide further insight into this choice, we examine the difference between *Ared* and *Pred*. If they are similar then we will conclude that the quadratic model is a good approximation to the objective function, and that the inner iteration is computing an effective search direction. Our focus is on the difference between the quadratic model and the higher-order terms in the Taylor series approximation to $f$, as motivated by the comments above.

Let us look at these quantities in more detail. Since $f \in C^2(\mathbb{R}^n)$,

$$Ared_k = f_k - f(x_k + s_k) = -s_k^T g_k - \frac{1}{2}s_k^T H(x_k + \xi s_k)s_k,$$

where $0 \leq \xi \leq 1$. For $Pred_k$ we obtain

$$Pred_k = f_k - [f_k + s_k^T g_k + \frac{1}{2}s_k^T H_k s_k] = -s_k^T g_k - \frac{1}{2}s_k^T H_k s_k.$$

Combining these, we have

$$Ared_k - Pred_k = \frac{1}{2}s_k^T [H_k - H(x_k + \xi s_k)]s_k.$$

Now, observe that if $f(x)$ is a quadratic function then

$$H_k = H(x_k + \xi s_k)$$

yielding

$$Ared_k - Pred_k = 0.$$

On the contrary, if $f(x)$ is not quadratic and the quadratic model is not a good approximation to it, then the difference $Ared_k - Pred_k$ will be large. Similarly, if $\|s_k\|$ is small, this difference will be small, as we can expect when the Newton-Krylov algorithm converges. Thus, on balance, we can monitor values of $|Ared_k - Pred_k|$ to possibly introduce an adaptive truncation criterion.

Our focus on *Ared* and *Pred* is reminiscent of trust region methods, but our approach is distinct. In a trust region method, if there is disagreement between *Ared* and *Pred* then a bound on the norm of the search direction is reduced. In our case (see the next Section 6.3) we will reduce a bound on the number of inner iterations, i.e., a bound on the computational effort. As motivated by [89], if the quadratic model is not a good approximation to the objective function, it is not worthwhile to use a large number of inner iterations to minimize the quadratic model to compute a search direction.

There is a relationship between the trust region approach and our approach. When the CG method is used in the inner iteration, the estimate of the search direction $s_k$ increases monotonically in norm (provided that a suitable norm is adopted), at each iteration. Hence bounding the norm of $s_k$ will limit the number of inner iterations, and *vice versa*, but the relationship between the two approaches is not precise. Even if the bound on $s_k$ is small a significant number of CG iterations might still result.

This is analogous to the relationship between the residual-based stopping criterion (2.75) and the quadratic-based stopping criterion (2.77). There is a theoretical relationship between them [87], but the latter is based directly on limiting the computational effort if it is determined that the inner iteration is not contributing to the progress of the optimization. As shown in [90], the norm of the residual can be a poor predictor of the quality of the search direction. It is our hope that we can reduce the effect of over-solving by focusing directly on the computational effort in the algorithm.

Our adaptive truncation criterion (see the next Section 6.3) will be based on $|Ared - Pred|$. We will be assessing how well the quadratic model approximates the objective function. This is different than in a traditional trust region method where it is more common to assess only whether the predicted reduction underestimates the actual reduction.

## 6.3 A novel Adaptive Truncation Criterion

The analysis of the previous section can be used for defining an Adaptive Truncation Criterion (ATC) to be used within a Newton-Krylov scheme. The quantity

$$\rho_k = \Big| Ared_k - Pred_k \Big|. \tag{6.1}$$

is at the basis of our adaptive rule. In particular, we adaptively set the maximum number $max\_it_{k+1}$ of inner iterations allowed at the outer iteration $k+1$, on the basis of the value of $\rho_k$.

The Adaptive Truncation Criterion we propose is detailed in the following scheme:

**ATC scheme**

***Data***: $0 < \gamma_1 < \gamma_2$, $0 < \sigma_3 < 1 < \sigma_2 < \sigma_1$, $0 < \theta_2 < \theta_1$ and $\ell \in \mathbb{N}$, $1 \leq \ell < n$.

If $\rho_k \leq C_k\gamma_1$ then (*very successful step*)
if $\alpha_k \geq \theta_1$ then set $max\_it_{k+1} = \min\{n\ ,\ \lfloor\sigma_1 max\_it_k\rfloor\}$

else if $\rho_k \leq C_k\gamma_2$ then (*successful step*)
if $\alpha_k \geq \theta_2$ then set $max\_it_{k+1} = \min\{n\ ,\ \lfloor\sigma_2 max\_it_k\rfloor\}$

else (*unsuccessful step*)
set $max\_it_{k+1} = \max\{\ell\ ,\ \lfloor\sigma_3 max\_it_k\rfloor\}$

The maximum number of inner iterations is increased in case of *successful steps*, otherwise it is decreased (*unsuccessful steps*). Note that in the *successful steps* an additional check on the stepsize $\alpha_k$ is introduced. This is motivated by the fact that when $p_k$ is poor, then $\|s_k\|$ can actually be very small after the linesearch procedure, possibly yielding an unexpected *successful/very successful* step. To prevent the latter drawback, following the rationale behind the proposal in [91], we verify that the quality of $s_k$ resembles $p_k$, by checking the steplength $\alpha_k$. Whenever the stepsize is too small, then $p_k$ is likely poor and we leave $max\_it_k$ unchanged.

The quantity $C_k$ is introduced in order to take into account the magnitude of $Ared_k$ and $Pred_k$, so that the adopted test is well scaled. A detailed discussion about possible choices for $C_k$ is reported in Section 6.4. The purpose of the threshold value $\ell$ is to guarantee that a certain number $\ell \ll n$ of inner iterations is anyhow performed. This value plays an important role whenever the information collected during the inner iterations is possibly used to construct preconditioners [48] and [49]. Indeed, in this case a threshold number of inner iterations might prevent the construction of an unreliable preconditioner. Of course, other possible preconditioning strategies might differently affect the choice of parameter $\ell$. Observe that, since in ATC we have $\ell \geq 1$, and since the first CG iteration produces a search direction proportional to the negative gradient, the ATC strategy always yields a gradient-related direction, which guarantees global convergence.

We can summarize the importance of the ATC criterion by observing that it complies with the following three issues:

1. it aims at extending the strategy in [91] already mentioned;
2. it attempts to partially exploit second order information on the objective function (including also the indefinite case), by considering in (6.1) a quadratic model update;
3. considering that we are dealing with large scale problems, it does not require significant additional computational burden and supplementary storage.

In order to clarify the latter key points, we first observe that the strategy in [91] substantially uses information from the linesearch procedure, to infer second order information on the function. Indeed, whenever the stepsize is equal to one, then the search direction is a Newton-like direction, implying that the local second order model is a "qualified" approximation of the objective function. In this regard, $\rho_k$ should be, to some extent, a measure of this "qualification". As regards item 2., note that $\rho_k$ includes information on second order derivatives of $f$, throughout the computation of the quadratic model. Thus, $\rho_k$ possibly summarizes some second order information on $f$, too. Finally, as concerns item 3., considering the large scale setting, the computational cost of ATC is definitely negligible.

Note that the early termination of the inner iterations is equivalent to restarting the iterative method used (say the CG method). In this regard, the use of a preconditioner (if any) could be helpful in avoiding a possible deterioration in performance due to this restart. This motivates the fact that in the numerical experiences we also include the use of a preconditioning strategy, combined with ATC. The considerations in the current section deserve a more accurate analysis based on numerical experiences, as reported in the next section.

We conclude this section by observing that, as already said, the rule we proposed is based on $\rho_k$. Since in (6.1) the predicted reduction is based on the quadratic Taylor series approximation, $\rho_k$ can be bounded in terms of the third derivatives of the objective function and the magnitude of the current search direction. As the algorithm converges, $\rho_k$ goes to zero, and our adaptive truncation criterion reverts to a traditional Newton-Krylov method.

## 6.4 Numerical experience

In order to assess the adaptive criterion proposed, we consider a standard implementation of a Newton-Krylov method. Namely, we used the linesearch-based truncated scheme described in Section 2.1.3.1. The CG method is employed in the inner iterations. The novelty consists in the adoption of the adaptive criterion described in the previous section, i.e. the maximum number of CG inner iterations allowed per outer iteration (initialized to $n$, i.e. $max\_it_1 = n$) is adaptively adjusted according to ATC. As regards the parameters in the ATC scheme, we set $\gamma_1 = 10^{-4}$, $\gamma_2 = 10^{-2}$, $\sigma_1 = 2$, $\sigma_2 = 1.1$, $\sigma_3 = 0.2$, $\theta_1 = 10^{-2}$, $\theta_2 = 10^{-4}$. This choice is suggested by a preliminary coarse tuning on the chosen test set. Moreover, since we tested ATC both within the unpreconditioned and the preconditioned framework proposed in [48], the value of the parameter $\ell$ is set to 7, in order to allow the construction of an effective preconditioner (see also the discussion about the choice of the parameter $h_{\max}$ in [48]).

As regards the set of test problems, we selected the same test set already used in this thesis, i.e. all the unconstrained convex and nonconvex large problems available in the CUTEst collection [56], and when a problem is of variable dimension, we considered two different dimensions (usually 1000 and 10000 variables). The resulting test set consists in 112 problems.

The algorithms were coded in FORTRAN 90 and the GFortran compiler under Linux Ubuntu 14.04 was used. The stopping criterion for the outer iterations is the

standard one given by

$$\|g_k\| \leq 10^{-5} \max\{1, \|x_k\|\} \tag{6.2}$$

(see, e.g. [83]). We state that a failure occurs whenever the latter test is not satisfied within 100000 outer iterations or if the CPU time exceeds 900 seconds. Moreover, in comparing different algorithms, we consider only the problems where the algorithms converge to the same stationary point. Following [24], this is checked by using the test

$$|f_1^* - f_2^*| \leq 10^{-3} \min\{|f_1^*|, |f_2^*|\} + 10^{-6},$$

being $f_1^*$, $f_2^*$ the optimal function values obtained by the two algorithms. Finally, we discarded all the test problems where the compared algorithms required a CPU time below 0.1 seconds.

As regards the stopping criterion for the CG inner iterations, we tested both the criteria recalled in Section 2.1.3.1.1: the residual-based criterion (2.75) and the quadratic model reduction-based criterion (2.77). Since the criterion (2.75) with $\eta_k = \min\{1/k, \|g_k\|\}$ proved to yield poorer results in practice, we preferred to report our results by considering the more reliable residual-based criterion adopted in [48]. This criterion sets

$$\eta_k \; = \; \max\left\{\|g_k\|, \|g_k\|^{\frac{1}{3}}\right\} \min\left\{\frac{n^{\frac{1}{2}}}{k}, \|g_k\|\right\},$$

which both takes into account the size ($n$) of the problem and allows a coarser solution when far from a stationary point. The criterion (2.77) adopts $\eta_k = 0.5$, as suggested in [90].

We remark that our main goal is to provide an adaptive rule to enhance the residual-based criterion (2.75). Nevertheless we also coupled ATC with criterion (2.77), though no significant improvement was expected, since (2.77) already contains second order information. On the basis of the latter observation, our numerical experience was designed in order to both assess the improvement of using (2.75)+ATC w.r.t.(2.75), and the substantial invariance of using (2.77)+ATC w.r.t. (2.77). However, we will also report (in Section 6.4.3) a comparison between our (linesearch-based) approach and the trust region approach (namely the use of TRON code [77]) to complete the numerical experiences.

We performed an extensive numerical experimentation and the results consist of several long tables which can not be reported here in full; thus, in the sequel, the comparisons are reported by means of performance profiles [42] in terms of number of function evaluations, number of CG inner iterations and CPU time. Occasionally, the detailed results on few test problems are reported too, in order to highlight specific behaviours. Note that in a Newton-Krylov method the number of inner iterations summarizes the effort to compute the search direction, while the CPU time is definitely a more complete measure of the overall computational burden.

### 6.4.1 Guidelines for the choice of $C_k$ in ATC scheme

The quantity $C_k$ in ATC plays the role of a scaling factor with respect to values of the objective function, which should duly take into account the size of $Ared_k$

and $Pred_k$. In particular, among several possibilities we tested the following two expressions of $C_k$

$$C_k = \min\{1, |f(x_k)|\} \tag{6.3}$$
$$C_k = \max\{1, |f(x_k)|\}, \tag{6.4}$$

whose rationale may be interpreted as follows. The expression (6.3) takes into account scaling of the function when $f(x_k)$ is relatively small (i.e. $|f(x_k)| \leq 1$). On the other hand, the expression (6.4) for $C_k$ takes into account scaling when $f(x_k)$ is relatively large. We experiment both the choices on the whole test set and, though apparently the choice (6.4) might be more intuitive, setting $C_k$ as in (6.3) yields better numerical results (at least on the test set considered). Note that with the choice (6.3), whenever $|f(x_k)|$ is small, we have $C_k$ close to zero. However, in this case we expect that also $Ared_k$ and $Pred_k$ are not relatively large. Therefore, if $C_k$ is close to zero and the test in ATC scheme fails, we expect that $Ared_k$ and $Pred_k$ differ appreciably, hence the step must be considered unsuccessful. In this regard, the choice for considering the step successful or unsuccessful is completely determined by the value of the parameter $\gamma_1$ and $\gamma_2$. For sake of brevity, we report here only the unpreconditioned case in Figures 6.1-6.3. The results concerning the preconditioned case are very similar.

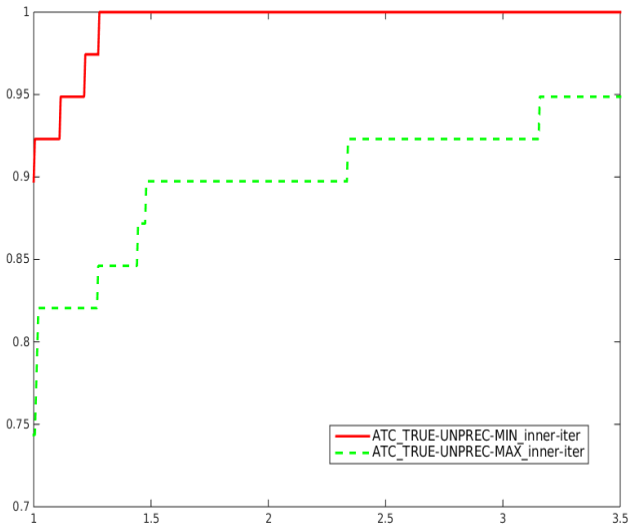


**Figure 6.1.** *Unpreconditioned* Newton-Krylov method using (2.75) with *ATC-true*: the choice of $C_k$ in (6.3) (solid line) vs. the choice of $C_k$ in (6.4) (dashed line), in terms of CG inner iterations.

On the basis of these results, on the test problems where we observe changes between using (6.3) and (6.4), we note that the choice of $C_k$ as in (6.3) is the best one. This motives the adoption of $C_k = \min\{1, |f(x_k)|\}$ for our numerical experience reported in the sequel.

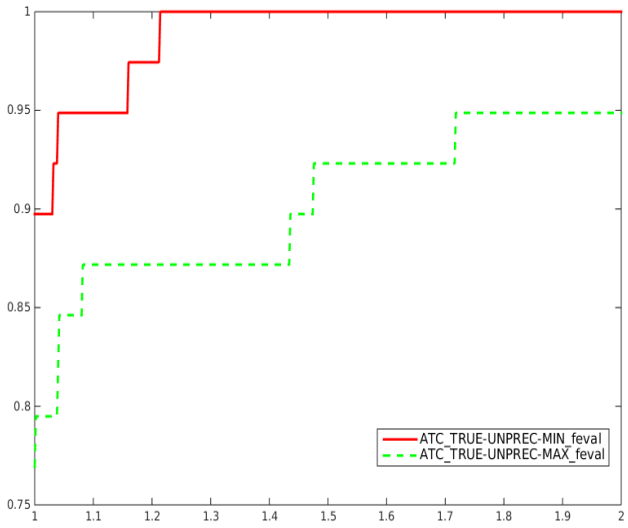


**Figure 6.2.** *Unpreconditioned* Newton-Krylov method using (2.75) with *ATC-true*: the choice of $C_k$ in (6.3) (solid line) vs. the choice of $C_k$ in (6.4) (dashed line), in terms of number of function evaluations.

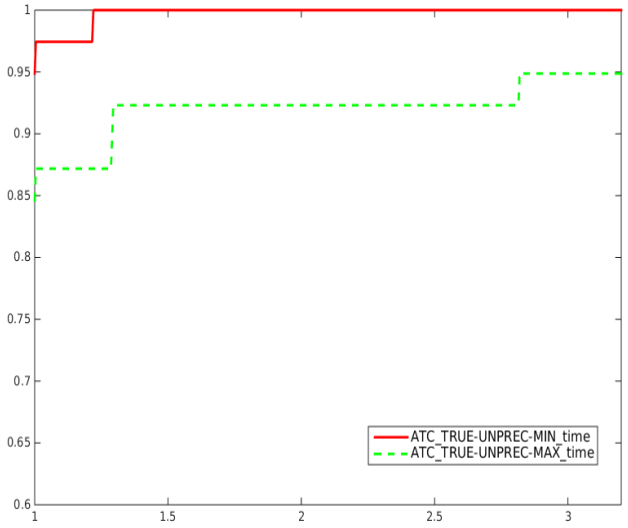


**Figure 6.3.** *Unpreconditioned* Newton-Krylov method using (2.75) with *ATC-true*: the choice of $C_k$ in (6.3) (solid line) vs. the choice of $C_k$ in (6.4) (dashed line), in terms of CPU time.

### 6.4.2 Numerical comparisons among different schemes

In this section we report the main results of the numerical experiences, namely the use of our adaptive truncation criterion ATC. In particular, our aim is to assess the improvement of using (2.75)+ATC w.r.t. (2.75) (in the figures, *ATC-true* vs. *ATC-false*). Moreover, we also report results obtained by using (2.77). We first observe that the adoption of ATC affects the results for several test problems, both in the unpreconditioned and preconditioned case in [48]. In Figures 6.4 and 6.5 we considered the ATC rule within the unpreconditioned Newton-Krylov method in [48], adopting the truncation criterion (2.75), w.r.t. CG inner iterations and CPU time, respectively. A similar numerical experience is reported in Figures 6.6 and 6.7, including the preconditioning strategy in [48]. By Figures 6.4-6.7, we deduce that the choice *ATC-true* outperforms *ATC-false* both in terms of number of CG inner iterations and CPU time. This confirms the expectation of our proposal, i.e., coupling the rule (2.75) with ATC, in practice enhances first order information with some second order information, thus improving the overall performance.

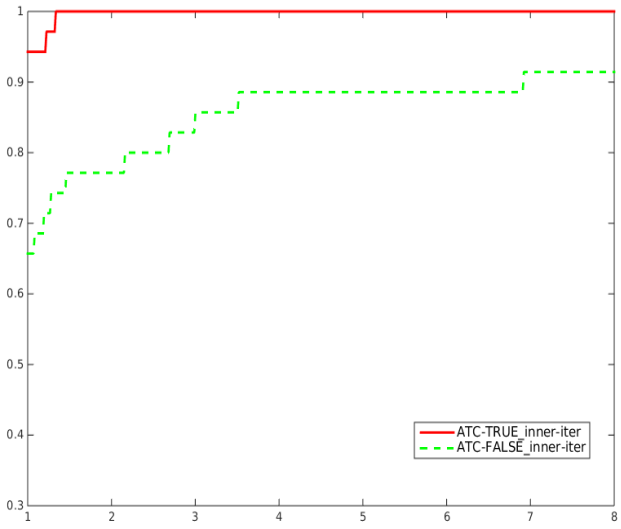


**Figure 6.4.** *Unpreconditioned* Newton-Krylov method using (2.75): comparison *ATC-true* vs. *ATC-false*, in terms of CG inner iterations.

To complete this numerical experience, in Figure 6.8 and 6.9 a comparison between the rule (2.75) with *ATC-true* and (2.77) is detailed, showing that, in practice, (2.75)+ATC both retains the appealing theoretical convergence properties of using (2.75) and, to some extent, performs similarly to (2.77). This reveals also that possibly the residual-based criterion does not include enough information on the reduction of the quadratic model, and that the joint use with ATC enables the partial recovery of this information. The latter phenomenon can be interpreted in the light of the following facts:

- a truncation criterion based on the reduction of the quadratic model (like (2.77)) already encompasses approximate second order information;

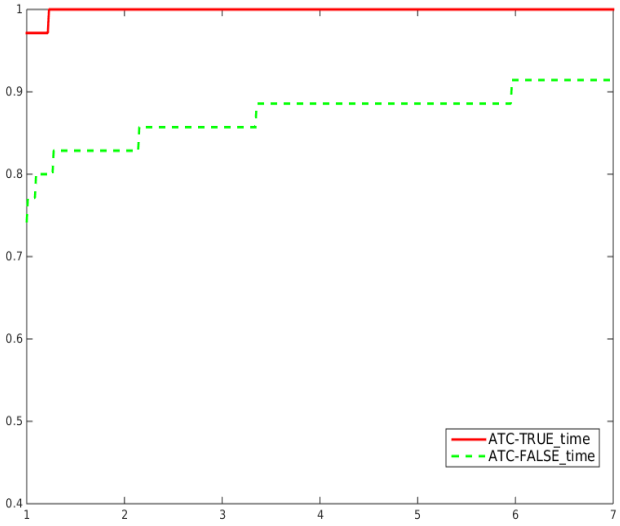


**Figure 6.5.** *Unpreconditioned* Newton-Krylov method using (2.75): comparison *ATC-true* vs. *ATC-false*, in terms of CPU time.

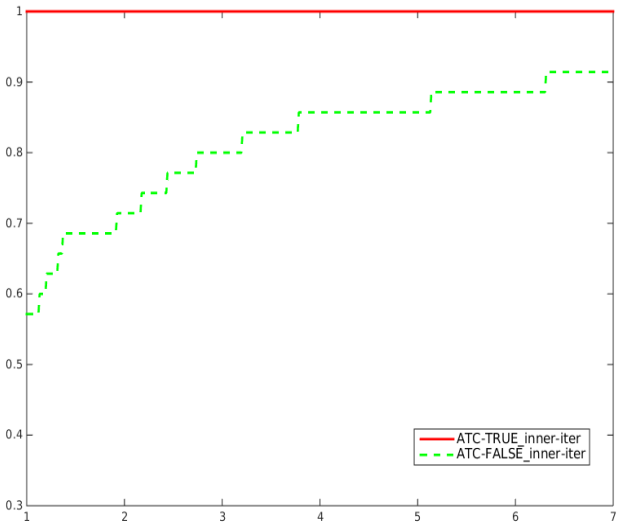


**Figure 6.6.** *Preconditioned* Newton-Krylov method using (2.75): comparison *ATC-true* vs. *ATC-false*, in terms of CG inner iterations.

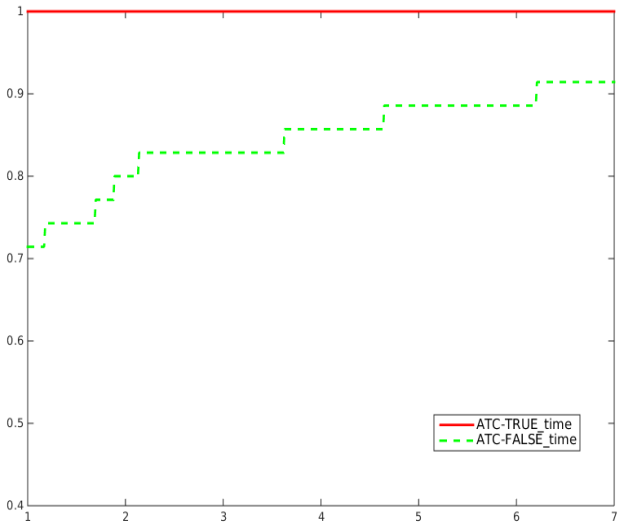


**Figure 6.7.** *Preconditioned* Newton-Krylov method using (2.75): comparison *ATC-true* vs. *ATC-false*, in terms of CPU time.

- the preconditioner we adopt is in the class of *approximate inverse preconditioners*, so that it includes approximate information on the Hessian matrix of the objective function, suitably collected during the early CG inner iterations;
- the test in the ATC scheme is based on a second order model of the objective function.

Thus, the approximate second order information in each of the latter three items has possibly a different source. As a matter of fact, pairing (2.75) with ATC proves to be successful, while coupling ATC with the first two items risks to spoil the information on second order derivatives, thus yielding inefficiency.

### 6.4.3 Comparison with trust region approach

Our ATC rule proposed in this work, to a large extent, draws inspiration from trust region methods, so that it could be significant to directly compare our (linesearch-based) approach versus a trust region approach. Humbly speaking, we do not aim at dealing with the old "quarrel" between trust region and linesearch-based methods. On the contrary, we only want to assess the behaviour of a Newton-Krylov method based on our ATC strategy, with respect to a standard trust region code. On this purpose we consider the TRON code [77] which represents one of the most commonly used implementation of a trust region Newton-Krylov method, for large bound-constrained problems (this code is available from Jorge Moré's web page). In this algorithm a descent direction for the trust region subproblem is generated, by means of a preconditioned conjugate gradient method, and the CG iterations are stopped whenever the trust region is violated, a negative curvature is encountered or a convergence condition is satisfied. For all the details we refer to [77].

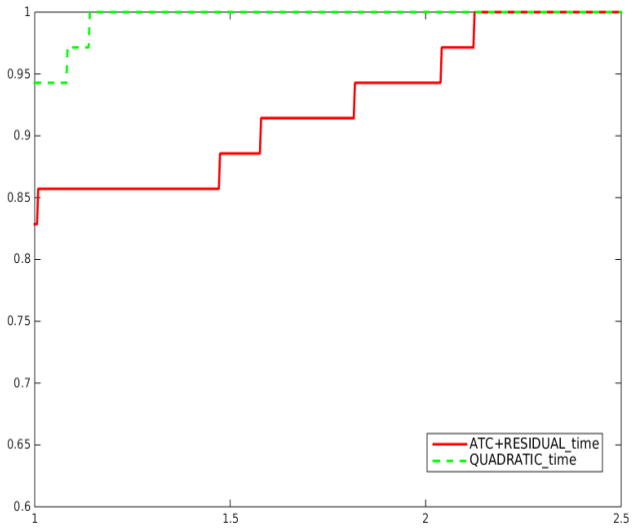


**Figure 6.8.** *Unpreconditioned* Newton-Krylov method: comparison between (2.75) with *ATC-true* and (2.77), in terms of CPU time.

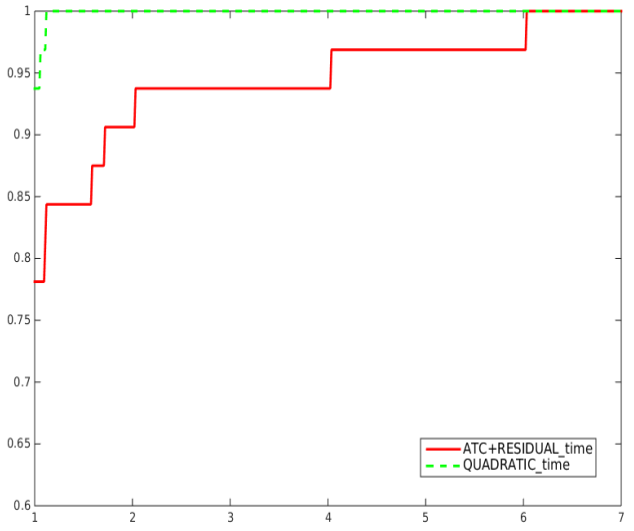


**Figure 6.9.** *Preconditioned* Newton-Krylov method: comparison between (2.75) with *ATC-true* and (2.77), in terms of CPU time.

We compare the results obtained by TRON on the whole test set versus those obtained by our approach. First, observe that, due to differences among the computational schemes used, the CPU time is likely the most significant indicator to assess the overall computational burden. Hence, it could be misleading to draw any conclusions by comparing, for instance, the number of outer iterations/function evaluations. Moreover, the default stopping criterion used in TRON code in the unconstrained case is

$$\|g_k\| \leq 10^{-5}. \tag{6.5}$$

As the authors themselves state, the test (6.5) is not scale invariant, and they prefer to replace (6.5) with $\|g_k\| \leq 10^{-5}\|g_0\|$ when solving some specific problems (see (7.2) in [77]). We run TRON by using all the default parameters of the code and by using both the original stopping criterion (6.5) and the one in (6.2). Of course, the test (6.5) is tighter than the test (6.2), so that the use of the stopping criterion (6.2) enables early stops for TRON code. As consequence, by using (6.2) in place of (6.5), TRON recovers some failures for CPU time limit or number of function evaluations.

On this guideline, in order to make a fair comparison with our proposal, we compare the results obtained by our preconditioned Newton-Krylov method and by TRON, using the same standard stopping criterion (6.2) for both the algorithms. This comparison is reported in Figures 6.10-6.12, in terms of number of function evaluations, number of CG inner iterations and CPU time, respectively. These plots show that our algorithm outperforms TRON in terms of CPU time, even if in terms of function evaluations and CG inner iterations TRON is more efficient than our proposal. The reason of this is clearly evidenced by observing some runs of TRON. Indeed (see also Table 6.1 below), on some large scale problems the Incomplete Cholesky Factorization (ICF) used by TRON, when computing the step in the trust region subproblem, is definitely time consuming. This implies that, in solving some difficult large scale problems, after 900 seconds only one or two outer iterations are possibly performed.

As regards the robustness, we remark that in order to evidence the comparison between the algorithms, in Figure 6.10 and Figure 6.11 the abscissa axis ranges in a large interval. This is due to the fact that TRON is much successful (in terms of number of function evaluations and CG inner iterations) for small values of the abscissa parameter, while it tends to lack robustness. In order to better clarify the latter issue, we report in Table 6.1 the detailed results for all the problems where at least one of the algorithms fails to converge.

For sake of completeness, in Table 6.1 we report the results obtained by our preconditioned Newton-Krylov method, including both the choices *ATC-true* and *ATC-false*, and by TRON when using both the original stopping criterion (6.5) and the standard one (6.2). We first remark that almost all the problems included in Table 6.1 have a large dimension. Moreover, on problems FLETCBV3 the algorithms converge towards different points so that the results obtained are not comparable. On the remaining eight test problems, even if in some cases by using the stopping criterion (6.2) TRON succeeds to converge (see problems NONCVXUN and SPARSINE), the CPU time needed is really lengthy. It it worthwhile to observe how on these problems the use of ATC rule proposed in this study is crucial for the success of the (linesearch-based) preconditioned Newton-Krylov method. In

| PROBLEM | $n$ | TRON (with original stopping criterion (6.5)) | | | TRON (with standard stopping criterion (6.2)) | | | *Preconditioned Newton-Krylov* with *ATC-true* (using (6.2)) | | | | *Preconditioned Newton-Krylov* with *ATC-false* (using (6.2)) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | it/nf | CG-it | time | it/nf | CG-it | time | it | nf | CG-it | time | it | nf | CG-it | time |
| FLETCBV3$^{(*)}$ | 1000 | $> 10^5$ | - | - | 9 | 8 | 0.00 | 9 | 9 | 14 | 0.00 | 9 | 9 | 14 | 0.00 |
| FLETCBV3$^{(*)}$ | 10000 | $> 10^5$ | - | - | 1870 | 1869 | 10.68 | 143 | 143 | 227 | 0.45 | 136 | 136 | 176 | 0.40 |
| FMINSURF | 5625 | - | - | >900 | - | - | >900 | 157 | 361 | 8414 | 12.51 | 23 | 133 | 16160 | 23.61 |
| NONCVXUN | 10000 | - | - | >900 | 10234 | 16976 | 461.02 | 3072 | 11940 | 25843 | 78.61 | - | - | - | >900 |
| PENALTY1 | 10000 | - | - | >900 | - | - | >900 | 64 | 123 | 80 | 0.13 | 64 | 123 | 80 | 0.11 |
| POWER | 10000 | - | - | >900 | - | - | >900 | 222 | 816 | 13343 | 6.03 | 118 | 704 | 84216 | 37.38 |
| SINQUAD | 10000 | 25 | 36 | 0.19 | 25 | 36 | 0.19 | - | - | - | >900 | - | - | - | >900 |
| SPARSINE | 10000 | - | - | >900 | 1999 | 3026104 | 864.60 | 901 | 2562 | 84553 | 144.26 | - | - | - | >900 |
| VARDIM | 10000 | - | - | >900 | - | - | >900 | 57 | 340 | 344 | 0.25 | 56 | 339 | 387 | 0.27 |
| VAREIGVL | 10000 | - | - | >900 | - | - | >900 | 21 | 179 | 20 | 0.08 | 21 | 179 | 20 | 0.08 |

**Table 6.1.** Detailed results obtained by TRON and by our preconditioned Newton-Krlov method. Note that $^{(*)}$ indicates test problems where the algorithms converge towards different stationary points.

this regard, by analyzing the complete results, it is possible to note that the latter situation very often occurs, as already highlighted in previous sections. On problem SINQUAD, our preconditioned Newton-Krylov methods actually detects the same stationary point generated by TRON in about half a second, but then the stopping criterion is not satisfied timely. On the remaining test problems in Table 6.1, TRON fails to converge and this occurs even if the maximum CPU time allowed would be greatly increased with respect to 900 seconds.

On summary, Figures 6.10-6.11, along with Table 6.1, reveal that to large extent our proposal compares favourably with respect to TRON.

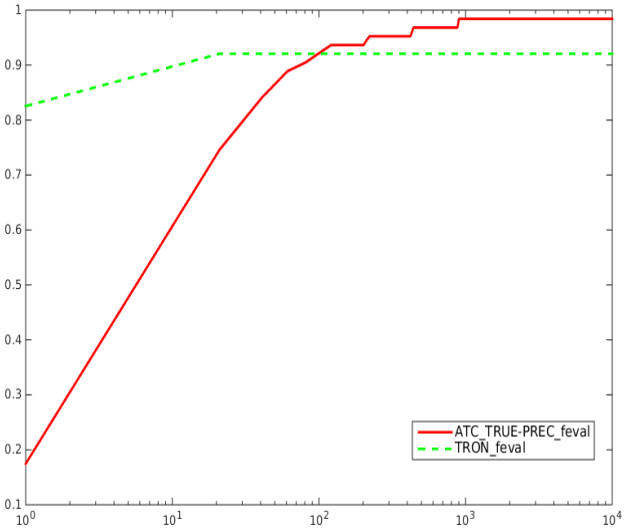


**Figure 6.10.** Comparison between our *Preconditioned* Newton-Krylov method with (2.75), *ATC-true* and TRON with standard stopping criterion (6.2), in terms of number of function evaluations. Abscissa axis is in logarithmic scale.

## 6.5 Conclusions

In this chapter we addressed the problem of "over-solving" the Newton equation within linesearch-based Newton-Krylov methods. In particular, we proposed an adaptive rule for dynamically setting the maximum number of inner iterations, at each outer iteration, for possibly nonconvex problems. It can be used jointly with any truncation criterion (based on the relative residual or on the reduction of the quadratic model). A significant numerical study has been performed, in order to assess the effectiveness and the robustness of the joint use of the adaptive rule and the two most popular truncation criteria. In particular, the adaptive rule has been experimented in both unpreconditioned and preconditioned frameworks. The results obtained confirm our guess that, the adaptive rule leads to computational savings in terms of overall number of inner iterations and CPU time, whenever the residual-based criterion (2.75) is used. Nevertheless, when (2.77) is coupled with

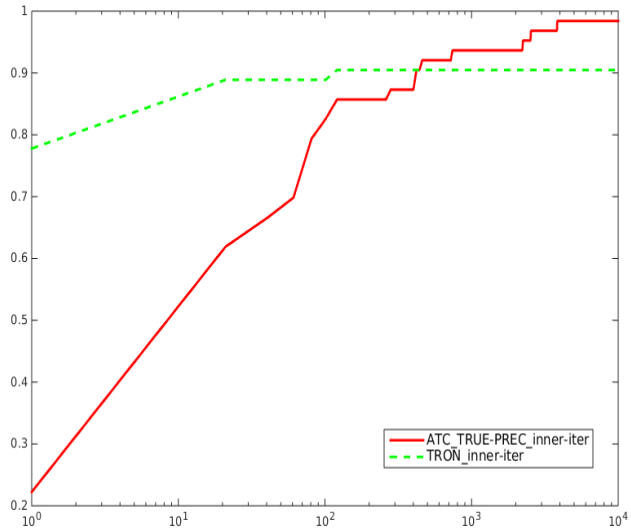


**Figure 6.11.** Comparison between our *Preconditioned* Newton-Krylov method with (2.75), *ATC-true* and TRON with standard stopping criterion (6.2), in terms of number of CG inner iterations. Abscissa axis is in logarithmic scale.

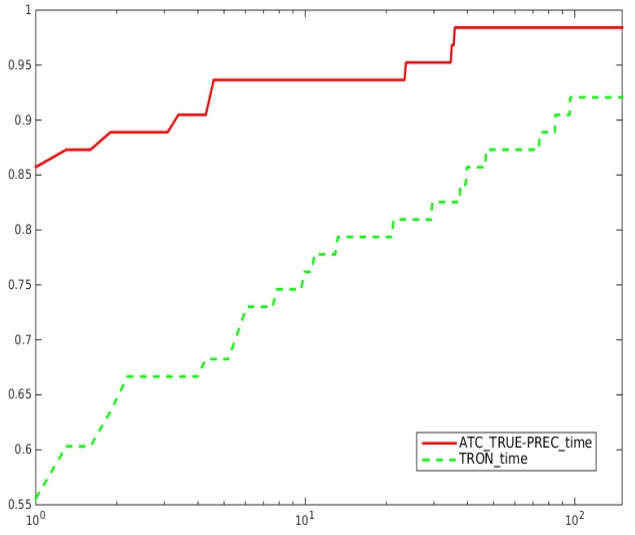


**Figure 6.12.** Comparison between our *Preconditioned* Newton-Krylov method with (2.75), *ATC-true* and TRON with standard stopping criterion (6.2), in terms of CPU time. Abscissa axis is in logarithmic scale.

ATC we did not experience evident improvement with respect to the sole use of (2.77).

The numerical experiences have been completed by a comparison with a preconditioned Newton-Krylov method based on trust region approach, namely with the TRON code. The results obtained showed that our linesearch-based preconditioned Newton-Krylov method, which uses the ATC rule proposed in this study, outperforms TRON in terms of CPU time. On the other hand, the trust region code is definitely more efficient in terms of number of function evaluations and CG inner iterations, though a lack of robustness is evidenced.

Overall, even if a careful tuning of the parameters used in the adaptive rule is certainly still needed, the results obtained definitely agree with those reported in [90] and [91], from which we have drawn inspiration for this work. Finally, we assert that the results obtained seem to indicate that the use of the adaptive rule is promising, particularly in tackling large scale difficult nonconvex problems.

# Chapter 7

# Approximate Inverse Preconditioners for Indefinite Linear Systems

In this chapter we introduce a work in progress: in particular we focus on a class of preconditioners for symmetric linear systems arising from numerical analysis and nonconvex optimization frameworks. Our aim is to embed the preconditioner within a linesearch-based Newton-Krylov method (see Section 2.1.3.1), using the conjugacy of the directions given by CG or SYMMBK algorithms. As well known, Newton-Krylov method needs to compute, at each outer iteration, a search direction by approximately solving the Newton equation (2.74). Starting from the class of preconditioners for symmetric linear systems described in [48] and used in Chapter 6, our aim is to provide an efficient class of preconditioners for the indefinite case. These preconditioners are specifically suited for *large* indefinite linear systems and may be obtained as *by-product* of Krylov subspace solvers, as well as by applying L-BFGS updates. Moreover, this proposal is also suited for the solution of a sequence of linear systems, say $Ax = b_i$ or $A_i x = b_i$, where respectively the right-hand side changes or the system matrix slightly changes, too. Each preconditioner in this class is identified by setting the values of a pair of parameters and a scaling matrix, which are user-dependent, and may be chosen according to the structure of the problem in hand.

## 7.1 Introduction

We study a class of preconditioners for the solution of the symmetric indefinite linear system

$$Ax = b, \qquad A \in \mathbb{R}^{n \times n}, \quad A = A^T,$$

where $n$ is *large* and we do not assume any sparsity pattern for the system matrix $A$.

Here we focus on the use of iterative methods to solve linear systems. Our iterative methods were introduced in Chapter 1, and are also used to collect sufficient information on the system matrix, in order to generate the preconditioners. We propose a general class of preconditioners, which uses information collected by any

Krylov subspace method or possibly using L-BFGS updates, in order to capture the structural properties of the system matrix.

The basic idea of our approach draws its inspiration from *Approximate Inverse Preconditioners*, which have proved in general to be remarkably robust and efficient in practice [19, 20]. These methods claim that in principle, an approximate inverse of $A$ should be computed and used as a preconditioner. Though in practice it might be difficult to ensure that the approximate inverse is sparse, suitable factorizations of matrix $A$ can be fruitfully exploited, in order to build the approximate inverse preconditioner. In particular, a generalization of the Gram-Schmidt process can be used to provide a tridiagonal factorization of $A^{-1}$, where the triangular matrices are in general dense. This is the basic idea of AINV preconditioner (see [19], Section 5.1.2).

Here we apply any Krylov subspace method to generate a tridiagonal factorization of $A^{-1}$. The latter is then used to build our preconditioners, namely the AINV$\mathcal{K}$ class, needing to store just a few vectors, without requiring any matrix storage and any product of matrices. As we collect information from Krylov subspace methods, we assume that the entries of the system matrix are not stored at once and the necessary information is gained by simply using a routine, which computes the product of the system matrix times a vector. Note that, typically, the product of a matrix times a vector allows fast parallel computing, which is another possible advantage of our approach, in large scale settings.

AINV$\mathcal{K}$ can be naturally extended to the solution of a sequence of large linear systems. When sequences of systems are tackled, we generate the preconditioner $\mathcal{P}$, for the solution of the first linear system in the sequence, i.e. $Ax = b_1$ or $A_1x = b_1$. Then, we apply $\mathcal{P}$ for solving either $Ax = b_i$ or $A_ix = b_i$, $i = 2, 3, \ldots$ Thus, the cost of computing $\mathcal{P}$, for $i = 1$, is repaid by accelerating the solution for $i = 2, 3, \ldots$; a similar strategy was proposed in [83]. The latter approach might be strongly advantageous in numerical analysis and optimization frameworks, where the cost for computing the preconditioner is relatively small, when compared to solving each linear system in the sequence. Furthermore, when a Krylov subspace method is adopted to compute the preconditioner, the full storage of system matrix is never required. On the other hand, the same Krylov subspace method might be used also to compute the solution of the linear system (see also [102, 105]).

## 7.2 Preliminaries

In this section we first introduce some preliminaries, then we propose our class of preconditioners. Consider the *indefinite* linear system

$$Ax = b, \tag{7.1}$$

where $A \in \mathbb{R}^{n\times n}$ is *symmetric* and $n$ is *large*. Assume that a Krylov subspace method is used for the solution of (7.1), e.g. the Lanczos process (see Section 1.1.2). As well known, the Lanczos process and the CG method (see Section 1.1.1) are equivalent as long as $A \succ 0$, whereas the CG, though cheaper, in principle may not cope with the indefinite case.

With reference to the definition in [53, 109], we say that a symmetric indefinite matrix $T$ is *factorizable* if the diagonal (or $2 \times 2$ block diagonal) matrix $D$ and the unit lower triangular matrix $L$ exist such that $T = LDL^T$. In the next assumption we consider that a finite number of steps, say $h \ll n$, of the Krylov subspace method adopted have been performed.

**Assumption 7.2.1. [Factorization]** *Let us consider any Krylov subspace method to solve the symmetric linear system* (7.1)*. Suppose at step $h$ of the Krylov subspace method, with $h \leq n-1$, the matrices $R_h \in \mathbb{R}^{n\times h}$, $T_h \in \mathbb{R}^{h\times h}$ and the vector $u_{h+1} \in \mathbb{R}^n$ are generated, such that*

$$AR_h = R_h T_h + \rho_{h+1} u_{h+1} e_h^T, \qquad \rho_{h+1} \in \mathbb{R}. \tag{7.2}$$

*Suppose that the matrix $T_h$ is factorizable, so that there exists the following decomposition:*

$$T_h = L_h B_h L_h^T, \tag{7.3}$$

*where*

$R_h = (u_1 \cdots u_h)$, $u_i^T u_j = 0$, $\|u_i\| = 1$, $1 \leq i \neq j \leq h+1$,

*$T_h$ is tridiagonal, irreducible, nonsingular, with eigenvalues not all coincident,*

*$B_h$ is $1 \times 1$ or $2 \times 2$ block diagonal, $L_h$ is unit lower bidiagonal.*

To have a better intuition on the reason for which $h$ steps of almost any Krylov subspace method satisfy Assumption 7.2.1, we remark that they are essentially all based on the generation of orthogonal vectors (the Lanczos vectors or the residuals for CG-based methods), used to transform the system (7.1) into a tridiagonal one. Then, they substantially differ only in the way the resulting tridiagonal system is solved by factorization.

In particular, also observe that from (7.2) we have $T_h = R_h^T A R_h$, so that whenever $A \succ 0$ then $T_h \succ 0$. The Krylov subspace method adopted may, in general, perform $m \geq h$ iterations, generating the orthonormal vectors $u_1, \ldots, u_m$. Then, we can set $R_h = (u_{\ell_1}, \ldots, u_{\ell_h})$, where $\{\ell_1, \ldots, \ell_h\} \subseteq \{1, \ldots, m\}$, and change relations (7.2)-(7.3) accordingly; i.e. Assumption 7.2.1 may hold selecting any $h$ out of the $m$ vectors (among $u_1, \ldots, u_m$) computed by the Krylov subspace method, up to step $m$.

Observe also that from Assumption 7.2.1 if $\rho_{h+1} \neq 0$ the subspace $span\{u_1, \ldots, u_h\}$ is not invariant under the transformation by matrix $A$. This implies that here we consider a more general case with respect to [16].

## 7.3 Our class of preconditioners AINV$\mathcal{K}$

On the basis of Assumption 7.2.1, we can now define our preconditioners. To this aim, suppose $T_h = L_h B_h L_h^T$ in (7.3), where $B_h = \bigoplus_{1\leq j\leq m}\{E_j^h\}$, and where either $E_j^h \in \mathbb{R}$ or $E_j^h \in \mathbb{R}^{2\times 2}$, for any $j$. Moreover, if $E_j^h \in \mathbb{R}^{2\times 2}$ for an index $j$, assume that we compute the eigen-decomposition

$$E_j^h = U_j^h D_j^h (U_j^h)^T, \tag{7.4}$$

with $D_j^h = diag\{d_{j_1,h}; d_{j_2,h}\}$ and $(U_j^h)^T U_j^h = U_j^h (U_j^h)^T = I_2$. On the other hand, if $E_j^h \in \mathbb{R}$ for an index $j$, for the sake of notation we again assume that (7.4) holds, setting

$$D_j^h \equiv d_{j_1,h} \equiv E_j^h \qquad \text{and} \qquad U_j^h = 1.$$

Then, we have

$$B_h = \bigoplus_{1\le j\le m} \left\{E_j^h\right\} = \bigoplus_{1\le j\le m} \left\{ U_j^h \begin{pmatrix} d_{j_1,h} & 0 \\ 0 & d_{j_2,h} \end{pmatrix} (U_j^h)^T \right\}$$

and we define (see also [52]) the matrix

$$|B_h| = \bigoplus_{1\le j\le m} \left\{ U_j^h \begin{pmatrix} |d_{j_1,h}| & 0 \\ 0 & |d_{j_2,h}| \end{pmatrix} (U_j^h)^T \right\}.$$

Moreover,

$$\begin{aligned} |B_h| &= \left[\bigoplus_{1\le j\le m} \left\{U_j^h\right\}\right] \cdot \left[\bigoplus_{1\le j\le m} \left\{ \begin{pmatrix} |d_{j_1,h}| & 0 \\ 0 & |d_{j_2,h}| \end{pmatrix} \right\}\right] \cdot \left[\bigoplus_{1\le j\le m} \left\{(U_j^h)^T\right\}\right] \\ &= U_h \cdot \mathcal{D}_h \cdot (U_h)^T, \end{aligned}$$

where

$$U_h = \bigoplus_{1\le j\le m} \left\{U_j^h\right\},$$

$$\mathcal{D}_h = \bigoplus_{1\le j\le m} \left\{ \begin{pmatrix} |d_{j_1,h}| & 0 \\ 0 & |d_{j_2,h}| \end{pmatrix} \right\},$$

and we also define

$$|T_h| \stackrel{\text{def}}{=} L_h |B_h| L_h^T.$$

Observe that of course, by the definition of $|B_h|$, we have $|T_h| = T_h$ in case $T_h$ is positive definite. Furthermore, it is easily seen that $|T_h|$ is positive definite, for any $h$, and $|T_h|^{-1} T_h^2 |T_h|^{-1} = I_h$ whenever $T_h \succ 0$. As a consequence, we have $T_h |T_h|^{-1} = (|T_h|^{-1} T_h)^T = L_h \hat{I}_h L_h^{-1}$, where

$$\hat{I}_h = B_h |B_h|^{-1} \tag{7.5}$$

is at most $2 \times 2$ block-diagonal with all the eigenvalues in $\{-1, +1\}$.

We are now ready to introduce the following class of preconditioners, which depends on the parameter $a$ and the matrices $W_h$, $D$

$$\begin{aligned} M_h^\sharp(a, W_h, D) &\stackrel{\text{def}}{=} D\left[I_n - (R_h \mid u_{h+1})(R_h \mid u_{h+1})^T\right] D^T \\ &\quad + (R_h \mid Du_{h+1}) \left(\begin{array}{c|c} |T_h(W_h)| & ae_h \\ \hline ae_h^T & 1 \end{array}\right)^{-1} (R_h \mid Du_{h+1})^T, \quad (7.6) \\ M_n^\sharp(a, W_h, D) &\stackrel{\text{def}}{=} R_n |T_n(W_h)|^{-1} R_n^T, \quad (7.7) \end{aligned}$$

where $a \in \mathbb{R}$, $W_h \in \mathbb{R}^{n\times n}$ is diagonal positive definite and $D \in \mathbb{R}^{n\times n}$ is nonsingular. Finally, we also define

$$|T_h(W_h)| = L_h U_h \left(W_h \mathcal{D}_h\right) U_h^T L_h^T \tag{7.8}$$

so that $W_h\mathcal{D}_h$ is diagonal.

In particular our idea is the following:

- as long as matrix $A$ in (7.1) is positive definite, in order to build $T_h$ we can use the CG method (see the relationship between Lanczos process and CG method in Section 1.1.3);
- if matrix $A$ in (7.1) is indefinite, in order to build $T_h$ we can use the SYMMBK algorithm (see Section 1.1.4.2).

Observe that a theoretical analysis and a full numerical experience are in progress, in order to show the effectiveness of the class of preconditioners AINV$\mathcal{K}$.

# Conclusions

In this thesis, in order to solve large scale unconstrained optimization problems, new preconditioning techniques for NCG methods and new developments in Newton-Krylov methods have been proposed, using the conjugacy of the directions given by CG or SYMMBK algorithms. Dealing with large scale problems, we propose new preconditioners matrix-free iteratively constructed to be used within Conjugate Gradient-type algorithms.

In particular, in Chapter 1 and Chapter 2 some preliminaries, respectively about iterative methods for solving linear systems and about some common methods for large scale unconstrained optimization, have been recalled.

Drawing inspiration from quasi-Newton updates, in order to obtain a good approximation of the inverse of the Hessian matrix, in Chapter 3 we have introduced new preconditioners to be used within the NCG method. In Chapter 4 we have extended the damped techniques to the NCG methods by using a class of preconditioners described in Chapter 3. In Chapter 5 some global convergence properties have been provided both an effective PNCG algorithm and the D-PR-PNCG method. In Chapter 6 an adaptive truncation rule for Newton-Krylov methods, both within the unpreconditioned and the preconditioned framework proposed in [48], has been provided. Finally, in Chapter 7 we have introduced some preliminaries and the structure of another class of preconditioners, within Newton-Krylov framework, specifically suited for large indefinite linear systems.

# Bibliography

[1] Ahn, C., Chew, W., Zhao, J., and Michielssen, E. Numerical study of approximate inverse preconditioner for two-dimensional engine inlet problems. *Electromagnetics*, **19** (1999), 131.

[2] Al-Baali, M. Descent property and global convergence of the Fletcher-Reeves method with inexact line search. *IMA Journal on Numerical Analysis*, **5** (1985), 121.

[3] Al-Baali, M. Damped techniques for enforcing convergence of quasi-Newton methods. *Optimization Methods and Software*, **29** (2014), 919.

[4] Al-Baali, M., Caliciotti, A., Fasano, G., and Roma, M. Exploiting damped techniques for nonlinear conjugate gradient methods. *Mathematical Methods of Operations Research*, **86** (2017), 501.

[5] Al-Baali, M., Caliciotti, A., Fasano, G., and Roma, M. Quasi-Newton based preconditioning and damped quasi-Newton schemes, for nonlinear conjugate gradient methods. In *Accepted for Publication on Springer Proceedings (PROMS)* (2018).

[6] Al-Baali, M. and Fletcher, R. On the order of convergence of preconditioned nonlinear conjugate gradient methods. *SIAM Journal on Scientific Computing*, **17** (1996), 658.

[7] Al-Baali, M. and Grandinetti, L. On practical modifications of the quasi-Newton BFGS methods. *AMO - Advanced Modeling and Optimization*, **11** (2009), 63.

[8] Al-Baali, M. and Grandinetti, L. Improved damped quasi-Newton methods for unconstrained optimization. *Pacific Journal of Optimization*, (2015). (To appear).

[9] Al-Baali, M., Grandinetti, L., and Pisacane, O. Damped techniques for the limited memory BFGS method for large-scale optimization. *Journal of Optimization Theory and Applications*, **161** (2014), 688.

[10] Al-Baali, M. and Purnama, A. Numerical experience with damped quasi-Newton optimization methods when the objective function is quadratic. *SQU Journal for Science*, **17** (2012), 1.

[11] Al-Baali, M., Spedicato, E., and Maggioni, F. Broyden's quasi-Newton methods for a nonlinear system of equations and unconstrained optimization: a review and open problems. *Optimization Methods and Software*, **29** (2014), 937.

[12] Alléon, G., Benzi, M., and Giraud, L. Sparse approximate inverse preconditioning for dense linear systems arising in computational electromagnetics. *Numerical Algorithms*, **16** (1997), 1.

[13] Andrei, N. Scaled memoryless BFGS preconditioned conjugate gradient algorithm for unconstrained optimization. *Optimization Methods and Software*, **22** (2007), 561.

[14] Babaie-Kafaki, S. Two modified scaled nonlinear conjugate gradient methods. *Journal of Computational and Applied Mathematics*, **261** (2014), 172.

[15] Babaie-Kafaki, S. and Ghanbari, R. The Dai-Liao nonlinear conjugate gradient method with optimal parameter choices. *European Journal of Operational Research*, **234** (2014), 625.

[16] Baglama, J., Calvetti, D., Golub, G., and Reichel, L. Adaptively preconditioned GMRES algorithms. *SIAM Journal on Scientific Computing*, **20** (1998), 243.

[17] Bellavia, S., Gondzio, J., and Morini, B. A matrix-free preconditioner for sparse symmetric positive definite systems and least-squares problems. *SIAM Journal on Scientific Computing*, **35** (2013), A192.

[18] Bellavia, S., Simone, V. D., Serafino, D. D., and Morini, B. A preconditioning framework for sequences of diagonally modified linear systems arising in optimization. *SIAM Journal on Numerical Analysis*, **50** (2013), 3280.

[19] Benzi, M. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, **182** (2002), 418.

[20] Benzi, M., Cullum, J., and Tuma, M. Robust approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing*, **22** (2000), 1318.

[21] Benzi, M., Kouhia, R., and Tuma, M. Stabilized and block approximate inverse preconditioners for problems in solid and structural mechanics. *Computer methods in applied mechanics and engineering*, **190** (2001), 6533.

[22] Bernstein, D. S. *Matrix mathematics: Theory, facts, and formulas with application to linear systems theory.* Princeton University Press (2005).

[23] Bertsekas, D. P. *Nonlinear programming.* Athena scientific Belmont (1999).

[24] Birgin, E. G., Castillo, R., and Martínez, J. M. Numerical comparison of augmented lagrangian algorithms for nonconvex problems. *Computational Optimization and Applications*, **31** (2005), 31.

[25] Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, New York (2004).

[26] Buckley, B. and Lenir, A. QN like variable storage conjugate gradients. *Mathematical Programming*, **27** (1983), 155.

[27] Calandra, H., Gratton, S., Lago, R., Pinel, X., and Vasseur, X. Two-level preconditioned Krylov subspace methods for the solution of three-dimensional heterogeneous Helmholtz problems in seismics. *Numerical Analysis and Applications*, **5** (2012), 175.

[28] Calandra, H., Gratton, S., Pinel, X., and Vasseur, X. An improved two-grid preconditioner for the solution of three-dimensional Helmholtz problems in heterogeneous media. *Numerical Linear Algebra with Applications*, **20** (2013), 663.

[29] Caliciotti, A., Fasano, G., Nash, S. G., and Roma, M. An adaptive truncation criterion, for linesearch-based truncated Newton methods in large scale nonconvex optimization. *Operations Research Letters*, **46** (2018), 7.

[30] Caliciotti, A., Fasano, G., Nash, S. G., and Roma, M. Data and performance profiles applying an adaptive truncation criterion, within linesearch-based truncated Newton methods, in large scale nonconvex optimization. *Data in Brief*, **17** (2018), 246.

[31] Caliciotti, A., Fasano, G., and Roma, M. Preconditioning strategies for nonlinear conjugate gradient methods, based on quasi-Newton updates. In *The American Institute of Physics (AIP) Conference Proceedings* (edited by Y. Sergeyev, D. Kvasov, F. Dell'Accio, and M. Mukhametzhanov), vol. 1776, pp. 0900071–0900074 (2016).

[32] Caliciotti, A., Fasano, G., and Roma, M. Novel preconditioners based on quasi-Newton updates for nonlinear conjugate gradient methods. *Optimization Letters*, **11** (2017), 835.

[33] Caliciotti, A., Fasano, G., and Roma, M. Preconditioned nonlinear conjugate gradient methods based on a modified secant equation. *Applied Mathematics and Computation*, **318** (2018), 196.

[34] Chandra, R., Eisenstat, S., and Schultz, M. *Conjugate gradient methods for partial differential equations*. Ph.D. thesis, Yale University New Haven, CT (1978).

[35] Conn, A. R., Gould, N. I. M., and Toint, P. L. LANCELOT: *A Fortran package for large-scale nonlinear optimization (Release A)*. Springer Verlag, Heidelberg, Berlin (1992).

[36] Conn, A. R., Gould, N. I. M., and Toint, P. L. *Trust-region methods*. MPS-SIAM Series on Optimization, Philadelphia, PA (2000).

[37] Dai, Y.-H. and Yuan, Y. A nonlinear conjugate gradient method with a strong global convergence property. *SIAM Journal on Optimization*, **10** (1999), 177.

[38] Dassios, I., Fountoulakis, K., and Gondzio, J. A preconditioner for a primal-dual Newton conjugate gradient method for compressed sensing problems. *SIAM Journal on Scientific Computing*, **37** (2015), A2783.

[39] Dembo, R., Eisenstat, S., and Steihaug, T. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, **19** (1982), 400.

[40] Dembo, R. and Steihaug, T. Truncated-Newton algorithms for large-scale unconstrained optimization. *Mathematical Programming*, **26** (1983), 190.

[41] Dennis, J. E., Jr and Wolkowicz, H. Sizing and least-change secant methods. *SIAM Journal on Numerical Analysis*, **30** (1993), 1291.

[42] Dolan, E. D. and Moré, J. Benchmarking optimization software with performance profiles. *Mathematical Programming*, **91** (2002), 201.

[43] Dong, X. L., Liu, H., Xu, Y. L., and Yang, X. M. Some nonlinear conjugate gradient methods with sufficient descent condition and global convergence. *Optimization Letters*, **9** (2015), 1421.

[44] Duff, I., Gratton, S., Pinel, X., and Vasseur, X. Multigrid based preconditioners for the numerical solution of two-dimensional heterogeneous problems in geophysics. *International Journal of Computer Mathematics*, **84** (2007), 1167.

[45] Eisenstat, S. and Walker, H. Globally convergent inexact Newton methods. *SIAM Journal on Optimization*, **4** (1994), 393.

[46] Eisenstat, S. and Walker, H. Choosing the forcing term in an inexact Newton method. *SIAM Journal on Scientific and Statistical Computing*, **17** (1996), 16.

[47] Esmaily-Moghadam, M., Bazilevs, Y., and Marsden, A. L. A new preconditioning technique for implicitly coupled multidomain simulations with applications to hemodynamics. *Computational Mechanics*, **52** (2013), 1141.

[48] Fasano, G. and Roma, M. Preconditioning Newton-Krylov methods in nonconvex large scale optimization. *Computational Optimization and Applications*, **56** (2013), 253.

[49] Fasano, G. and Roma, M. A novel class of approximate inverse preconditioners for large scale positive definite linear systems in optimization. *Computational Optimization and Applications*, **65** (2016), 399.

[50] Fletcher, R. and Reeves, C. M. Function minimization by conjugate gradients. *The computer journal*, **7** (1964), 149.

[51] Gilbert, J. and Nocedal, J. Global convergence properties of conjugate gradient methods for optimization. *SIAM Journal on Optimization*, **2** (1992), 21.

[52] Gill, P. E., Murray, W., Ponceleon, D. B., and Saunders, M. A. Preconditioners for indefinite systems arising in optimization. *SIAM Journal on Matrix Analysis and Applications*, **13** (1992), 292.

[53] Gill, P. E., Saunders, M. A., and Shinnerl, J. R. On the stability of Cholesky factorization for symmetric quasidefinite systems. *SIAM Journal on Matrix Analysis and Applications*, **17** (1996), 35.

[54] Golub, G. and Van Loan, C. *Matrix Computations*. The John Hopkins Press, Baltimore (2012). Fourth edition.

[55] Gould, N. I., Lucidi, S., Roma, M., and Toint, P. L. Solving the trust-region subproblem using the Lanczos method. *SIAM Journal on Optimization*, **9** (1999), 504.

[56] Gould, N. I. M., Orban, D., and Toint, P. L. CUTEst: a constrained and unconstrained testing environment with safe threads. *Computational Optimization and Applications*, **60** (2015), 545.

[57] Gratton, S., Laloyaux, P., Sartenaer, A., and Tshimanga, J. A reduced and limited-memory preconditioned approach for the 4D-Var data-assimilation problem. *Quarterly Journal of the Royal Meteorological Society*, **137** (2011), 452.

[58] Gratton, S., Mercier, S., Tardieu, N., and Vasseur, X. Limited memory preconditioners for symmetric indefinite problems with application to structural mechanics. *Numerical Linear Algebra with Applications*, **23** (2016), 865.

[59] Gratton, S. and Tshimanga, J. An observation-space formulation of variational assimilation using a restricted preconditioned conjugate gradient algorithm. *Quarterly Journal of the Royal Meteorological Society*, **135** (2009), 1573.

[60] Grattona, S., Sartenaer, A., and Tshimanga, J. On a class of limited memory preconditioners for large scale linear systems with multiple right-hand sides. *SIAM Journal on Optimization*, **21** (2011), 912.

[61] Greenbaum, A. *Iterative methods for solving linear systems*. SIAM (1997).

[62] Grippo, L. and Lucidi, S. A globally convergent version of Polak-Ribìere conjugate gradient method. *Mathematical Programming*, **78** (1997), 375.

[63] Grippo, L. and Sciandrone, M. *Metodi di ottimizzazione non vincolata*. Springer-Verlag Italia, Milan (2011).

[64] Gürol, S., Weaver, A., Moore, A., Piacentini, A., Arango, H., and Gratton, S. B-preconditioned minimization algorithms for variational data assimilation with the dual formulation. *Quarterly Journal of the Royal Meteorological Society*, **140** (2014), 539.

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

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

[67] Hager, W. and Zhang, H. A survey of nonlinear conjugate gradient methods. *Pacific Journal of Optimization*, **2** (2006), 35.

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

[69] Hestenes, M. R. and Stiefel, E. *Methods of conjugate gradients for solving linear systems*, vol. 49. NBS (1952).

[70] Hladik, I., Reed, M., and Swoboda, G. Robust preconditioners for linear elasticity FEM analyses. *International Journal for Numerical Methods in Engineering*, **40** (1997), 2109.

[71] Huang, H. and Lin, S. A modified Wei-Yao-Liu conjugate gradient method for unconstrained optimization. *Applied Mathematics and Computation*, **231** (2014), 179.

[72] Izmailov, A. and Solodov, M. A truncated SQP method based on inexact interior-point solutions of subproblems. *SIAM Journal on Optimization*, **20** (2010), 2584.

[73] Kaporin, I. E. and Konshin, I. N. A parallel block overlap preconditioning with inexact submatrix inversion for linear elasticity problems. *Numerical linear algebra with applications*, **9** (2002), 141.

[74] Kelley, C. T. *Iterative methods for Optimization*. SIAM Frontiers in Applied Mathematics, Philadelphia, PA (1999).

[75] Kim, S. and Im, Y. Parallel processing of 3D rigid-viscoplastic finite element analysis using domain decomposition and modified block Jacobi preconditioning technique. *Journal of Materials Processing Technology*, **134** (2003), 254.

[76] Konshin, I., Olshanskii, M., and Vassilevski, Y. LU factorizations and ILU preconditioning for stabilized discretizations of incompressible Navier-Stokes equations. *Numerical Linear Algebra with Applications*, **24** (2017).

[77] Lin, C.-J. and Moré, J. Newton's method for large bound-constrained optimization problems. *SIAM Journal on Optimization*, **9** (1999), 1100.

[78] Liu, D. and Nocedal, J. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, **45** (1989), 503.

[79] Luenberger, D. G. *Introduction to linear and nonlinear programming*, vol. 28. Addison-Wesley Reading, MA (1973).

[80] Mack, C. J. and Schmid, P. J. A preconditioned Krylov technique for global hydrodynamic stability analysis of large-scale compressible flows. *Journal of Computational Physics*, **229** (2010), 541.

[81] Menk, A. and Bordas, S. A robust preconditioning technique for the extended finite element method. *International Journal for Numerical Methods in Engineering*, **85** (2011), 1609.

[82] Mercier, S., Gratton, S., Tardieu, N., and Vasseur, X. A new preconditioner update strategy for the solution of sequences of linear systems in structural mechanics: application to saddle point problems in elasticity. *Computational Mechanics*, **60** (2017), 969.

[83] Morales, J. and Nocedal, J. Automatic preconditioning by limited memory quasi-Newton updating. *SIAM Journal on Optimization*, **10** (2000), 1079.

[84] Moré, J. and Thuente, D. Line search algorithms with guaranteed sufficient decrease. *ACM Transactions on Mathematical Software (TOMS)*, **20** (1994), 286.

[85] Moré, J. and Wright, S. *Optimization Software Guide*. SIAM - Frontiers in Applied Mathematics, Philadelphia (1993).

[86] Nash, S. G. Newton-type minimization via the lanczos method. *SIAM Journal on Numerical Analysis*, **21** (1984), 770.

[87] Nash, S. G. Truncated-Newton methods for large-scale function minimization. In *Applications of Nonlinear Programming to Optimization and Control* (edited by H. Rauch), pp. 91–100. Pergamon Press, Oxford (1984).

[88] Nash, S. G. A survey of truncated-Newton methods. *Journal of Computational and Applied Mathematics*, **124** (2000), 45.

[89] Nash, S. G. and Nocedal, J. A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization. *SIAM Journal on Optimization*, **1** (1991), 358.

[90] Nash, S. G. and Sofer, A. Assessing a search direction within a truncated-Newton method. *Operations Research Letters*, **9** (1990), 219.

[91] Nash, S. G. and Sofer, A. A general-purpose parallel algorithm for unconstrained optimization. *SIAM Journal on Optimization*, **1** (1991), 530.

[92] Nazareth, L. A relationship between the BFGS and conjugate gradient algorithms and its implications for new algorithms. *SIAM Journal on Numerical Analysis*, **16** (1979), 794.

[93] Nocedal, J. Updating Quasi-Newton matrices with limited storage. *Mathematics of Computation*, **35** (1980), 773.

[94] Nocedal, J. Large scale unconstrained optimization. In *The state of the art in Numerical Analysis* (edited by A. Watson and I. Duff), pp. 311–338. Oxford University Press, Oxford (1997).

[95] Nocedal, J. and Wright, S. *Numerical Optimization*. Springer-Verlag, New York (2006). Second edition.

[96] O'Leary, D. P. and Yeremin, A. The linear algebra of block quasi-Newton algorithms. *Linear Algebra and its Applications*, **212** (1994), 153.

[97] Polak, E. and Ribiere, G. Note sur la convergence de méthodes de directions conjuguées. *Revue française d'informatique et de recherche opérationnelle, série rouge*, **3** (1969), 35.

[98] Polyak, B. T. Introduction to optimization. translations series in mathematics and engineering. *Optimization Software*, (1987).

[99] Powell, M. J. D. Algorithms for nonlinear constraints that use Lagrangian functions. *Mathematical Programming*, **14** (1978), 224.

[100] Powell, M. J. D. How bad are the BFGS and DFP methods when the objective function is quadratic? *Mathematical Programming*, **34** (1986), 34.

[101] Pytlak, R. *Conjugate Gradient Algorithms in Nonconvex Optimization*. Springer, Berlin (2009).

[102] Saad, Y. *Iterative Methods for Sparse Linear Systems - Second Edition*. SIAM, Philadelphia, PA (2003).

[103] Schlick, T. and Fogelson, A. TNPACK - A truncated Newton package for large-scale problems: I. Algorithm and usage. *ACM Transaction on Mathematical Software*, **18** (1992), 46.

[104] Schlick, T. and Fogelson, A. TNPACK - A truncated Newton package for large-scale problems: II. Implementation examples. *ACM Transaction on Mathematical Software*, **18** (1992), 71.

[105] Simoncini, V. and Szyld, D. Recent computational developments in Krylov subspace methods for linear systems. *Numerical Linear Algebra with Applications*, **14** (2007), 1.

[106] Trefethen, L. N. and Bau III, D. *Numerical linear algebra*, vol. 50. SIAM (1997).

[107] Tshimanga, J., Gratton, S., Weaver, A., and Sartenaer, A. Limited-memory preconditioners, with application to incremental four-dimensional variational data assimilation. *Quarterly Journal of the Royal Meteorological Society*, **134** (2008), 751.

[108] Van der Vorst, H. A. *Iterative Krylov methods for large linear systems*, vol. 13. Cambridge University Press (2003).

[109] Vanderbei, R. J. Symmetric quasi-definite matrices. *SIAM Journal of Optimization*, **5** (1995), 100.

[110] Veersé, F., Auroux, D., and Fisher, M. Limited-memory BFGS diagonal preconditioners for a data assimilation problem in meteorology. *Optimization and Engineering*, **1** (2000), 323.

[111] Vlček, J. and Lukšan, L. A conjugate directions approach to improve the limited-memory bfgs method. *Applied Mathematics and Computation*, **219** (2012), 800.

[112] Xie, D. and Schlick, T. Efficient implementation of the truncated-Newton algorithm for large-scale chemistry applications. *SIAM Journal on Optimization*, **10** (1999), 132.

[113] Yuan, G., Wei, Z., and Li, G. A modified Polak-Ribière-Polyak conjugate gradient algorithm for nonsmooth convex programs. *Journal of Computational and Applied Mathematics*, **255** (2014), 86.