**Springer Proceedings in Mathematics & Statistics**

Volume 224

## Springer Proceedings in Mathematics & Statistics

This book series features volumes composed of selected contributions from workshops and conferences in all areas of current research in mathematics and statistics, including operation research and optimization. In addition to an overall evaluation of the interest, scientific quality, and timeliness of each proposal at the hands of the publisher, individual contributions are all refereed to the high quality standards of leading journals in the field. Thus, this series provides the research community with well-edited, authoritative reports on developments in the most exciting areas of mathematical and statistical research today.

More information about this series at http://www.springer.com/series/10533

Alberto A. Pinto · David Zilberman
Editors

# Modeling, Dynamics, Optimization and Bioeconomics III

## DGS IV, Madrid, Spain, June 2016, and Bioeconomy VIII, Berkeley, USA, April 2015—Selected Contributions

Springer

*Editors*
Alberto A. Pinto
LIAAD—INESC TEC, Department of Mathematics, Faculty of Science
University of Porto
Porto
Portugal

David Zilberman
Department of Agricultural and Resource Economics
University of California
Berkeley, CA
USA

ISSN 2194-1009 ISSN 2194-1017 (electronic)
Springer Proceedings in Mathematics & Statistics
ISBN 978-3-319-74085-0 ISBN 978-3-319-74086-7 (eBook)
https://doi.org/10.1007/978-3-319-74086-7

Library of Congress Control Number: 2018930243

Mathematics Subject Classification (2010): 37-XX, 49-XX, 91-XX, 58-XX, 60-XX, 62-XX, 97M10, 97M40



Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# Foreword

*L'esprit n'use de sa faculté créatrice que quand l'expérience lui en impose la nécessité.*

Henri Poincaré

The tremendous challenges that are faced by humanity as the 21st century unfolds itself require a multidisciplinary approach. Now, more than ever before, the need for exploring our creative capacities to solve relevant problems for society is crucial for our very survival. However, this can only be done by a cross-cultural and multidisciplinary networking effort.

Having a group of scientists from different areas networking and exchanging experiences in a vibrant environment requires the correct environment. One can safely say that such environment came about in the two opportunities connected to the present volume of papers, namely in Berkeley, at the University of California, during the month of March 2014, and in Madrid, at the Universidad Nacional de Educatión a Distancia (UNED), during the month of June 2016.

In the first occasion, during the Seventh Berkeley Bioeconomy Conference, the ideal environment of the San Francisco Bay Area, with its sunny days and foggy afternoons, conjoined with the highly inquisitive and revolutionary tradition of the Cal Berkeley Campus had as its central theme "Biofuels as part of a sustainable strategy". In this occasion, an array of leading experts under the coordination of David Zilberman tackled topics ranging from global biofuel investments to the future of Brazilian biofuels, passing through extreme weather, biotechnology and agricultural productivity. In the present volume, the paper "Simulation and Advanced Control of the Continuous Biodiesel Production Process" by Brásio et al. and "Myopia of Governments and Optimality of Irreversible Pollution Accumulation" by Policardo are good examples of a quantitative follow-up of the conference themes. The paper by Mendes et al. deals with modelling by differential equations the kinetic separation of hexane isomers when they flow through a packed bed containing the micro-porous Metal-Organic Framework (MOF) ZIF-8 adsorbent. It is shown that a proper combination of two characteristic times can lead

to very different dynamics of fixed bed adsorbers wherein a limiting case can give rise to a spontaneous breakthrough curve of solutes.

In the second occasion, in the quiet Madrileño Summer and under the auspices of the UNED, we had the "4th International Conference on Dynamics, Games and Science" with the key topic being decision models in a complex economy. Here, under the warm hospitality of our Spanish colleagues, I was pleased to witness a broad plethora of distinguished speakers discussing topics ranging from human decisions, from a game theoretical viewpoint, to the simulation of energy demand and efficiency and passing through swarms of interacting agents in random environments. The remaining papers that can be found in the present volume are in a certain sense a written testimony of such diversity and effusiveness of interactions. For instance, the article by dos Santos et al. studies the influence of human mobility of dengue's transmission in the state of Rio de Janeiro from a statistical viewpoint. Still within the area of statistics, but from a broad theoretical perspective, the chapter by Casaca discusses prior information in Bayesian linear multivariate regression. The work of Nassif et al. presents a mathematical model for the tick life cycle based on the McKendrick partial differential equation. The article of Balsa et al. proposes a two-phase acceleration technique for the solution of symmetric and positive-definite linear systems with multiple right-hand sides. The paper by Rüppel et al. presents a constructive proof of the complete nonholonomy of the rolling ellipsoid. The two articles by Lopes and collaborators deal with important theoretical aspects of dynamical systems (such as the fat attractor) and quantum mechanics.

Summing up, the present volume displays a variety of works by leading researchers in a broad range of subjects where mathematical models have a substantial role and impact in society.

Rio de Janeiro, Brasil Jorge P. Zubelli
March 2017

# Acknowledgements

We thank the authors of the chapters for having shared their vision with us in this book, and we thank the anonymous referees.

We are grateful to Jorge P. Zubelli for contributing the foreword of this book.

We thank the Executive Editor for Mathematics, Computational Science and Engineering at Springer-Verlag, Martin Peters, for his invaluable suggestions and advice throughout this project. We thank Ruth Allewelt at Springer-Verlag for her assistance throughout this project.

We thank João Paulo Almeida, Susan Jenkins, José Martins, Abdelrahim Mousa, Bruno Oliveira, Diogo Pinheiro, Filipe Martins, Renato Soeiro, Ricard Trinchet Arnejo and Yusuf Aliyu Ahmad for their invaluable help in assembling this volume and for editorial assistance. Alberto Adrego Pinto would like to thank LIAAD–INESC TEC and gratefully acknowledge the financial support received by the FCT —Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology)—within project UID/EEA/50014/2013 and European Regional Development Fund (ERDF) through the COMPETE Program (operational programme for competitiveness) and by National Funds, through the FCT within Project "Dynamics, optimization and modelling" with reference PTDC/MAT-NAN/6890/2014, and Project "NanoSTIMA: Macro-to-Nano Human Sensing: Towards Integrated Multimodal Health Monitoring and Analytics/N0RTE-01-0145-FEDER-000016" financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

# Contents

**Optimal Regional Regulation of Animal Waste** . . . . . 1
Antti Iho, Doug Parker and David Zilberman

**An Overview of Synchrony in Coupled Cell Networks** . . . . . 25
Manuela A. D. Aguiar and Ana P. S. Dias

**Inexact Subspace Iteration for the Consecutive Solution of Linear Systems with Changing Right-Hand Sides** . . . . . 49
Carlos Balsa, Michel Daydé, José M. L. M. Palma and Daniel Ruiz

**Location Around Big Cities as Central Places** . . . . . 79
Fernando Barreiro-Pereira

**Predicting Energy Demand in Spain and Compliance with the Greenhouse Gas Emissions Agreements** . . . . . 107
Diego J. Bodas-Sagi and José M. Labeaga

**Simulation and Advanced Control of the Continuous Biodiesel Production Process** . . . . . 127
Ana S. R. Brásio, Andrey Romanenko and Natércia C. P. Fernandes

**Prior Information in Bayesian Linear Multivariate Regression** . . . . . 147
J. Casaca

**Perceptions of True and Fair View: Effects of Professional Status and Maturity** . . . . . 159
J. A. Gonzalo-Angulo, A. M. Garvey and L. Parte

**Topics of Disclosure on the Websites: An Empirical Analysis for FinTech Companies** . . . . . 187
T.-C. Herrador-Alcaide and M. Hernández-Solís

**On the Thin Boundary of the Fat Attractor** . . . . . 205
Artur O. Lopes and Elismar R. Oliveira

**Transport and Large Deviations for Schrodinger Operators and Mather Measures** . . . . . 247
A. O. Lopes and Ph. Thieullen

**Dynamics of a Fixed Bed Adsorption Column in the Kinetic Separation of Hexane Isomers in MOF ZIF-8** . . . . . 257
Patrícia A. P. Mendes, Alírio E. Rodrigues, João P. Almeida and José A. C. Silva

**A Simulation Model for the Physiological Tick Life Cycle** . . . . . 273
Nabil Nassif, Dania Sheaib and Ghina El Jannoun

**Long-Term Value Creation in Mergers and Acquisitions: Contribution to the Debate** . . . . . 285
Julio Navío-Marco and Marta Solórzano-García

**Cournot Duopolies with Investment in R&D: Regions of Nash Investment Equilibria** . . . . . 303
B. M. P. M. Oliveira, J. Becker Paulo and Alberto A. Pinto

**A Stochastic Logistic Growth Model with Predation: An Overview of the Dynamics and Optimal Harvesting** . . . . . 313
S. Pinheiro

**Myopia of Governments and Optimality of Irreversible Pollution Accumulation** . . . . . 331
Laura Policardo

**Stochastic Modelling of Biochemical Networks and Inference of Model Parameters** . . . . . 369
Vilda Purutçuoğlu

**Complete Nonholonomy of the Rolling Ellipsoid - A Constructive Proof** . . . . . 387
F. Rüppel, F. Silva Leite and R. C. Rodrigues

**Methodological Approaches to Analyse Financial Exclusion from an Urban Perspective** . . . . . 403
Cristina Ruza-Paz-Curbera, Beatriz Fernández–Olit and Marta de la Cuesta-González

**Prospective Study About the Influence of Human Mobility in Dengue Transmission in the State of Rio de Janeiro** . . . . . 419
Bruna C. dos Santos, Larissa M. Sartori, Claudia Peixoto, Joyce S. Bevilacqua and Sergio M. Oliva

**The Impact of the Public-Private Investments in Infrastructure on Agricultural Exports in Latin American Countries** . . . . . 429
Bárbara Soriano and Amelia Pérez Zabaleta

**Major Simulation Tools for Biochemical Networks** . . . . . 443
Gökçe Tuncer and Vilda Purutçuoğlu

# Inexact Subspace Iteration for the Consecutive Solution of Linear Systems with Changing Right-Hand Sides

**Carlos Balsa, Michel Daydé, José M. L. M. Palma and Daniel Ruiz**

**Abstract** We propose a two-phase acceleration technique for the solution of Symmetric and Positive Definite linear systems with multiple right-hand sides. In the first phase we compute some partial spectral information related to the ill conditioned part of the given coefficient matrix and, in the second phase, we use this information to improve the convergence of the Conjugate Gradient algorithm. This approach is adequate for large scale problems, like the simulation of time dependent differential equations, where it is necessary to solve consecutively several linear systems with the same coefficient matrix (or with matrices that present very close spectral properties) but with changing right-hand sides. To compute the spectral information, in the first phase, we combine the block Conjugate Gradient algorithm with the Inexact Subspace Iteration to build a purely iterative algorithm, that we call BlockCGSI. We proceed to an inner-outer convergence analysis and we show that it is possible to determine when to stop the inner iteration in order to achieve the targeted invariance in the outer iteration. The spectral information is used in a second phase to remove the effect of the smallest eigenvalues in two different ways: either by building a Spectral Low Rank Update preconditioner, or by performing a deflation of the initial residual in order to remove part of the solution corresponding to the smallest eigenvalues.

**Keywords** Inexact inverse iteration · Subspace iteration · Block conjugate gradient · Chebyshev filtering polynomials · Spectral projector

C. Balsa (✉)
Instituto Politécnico de Bragança (IPB), Bragança, Portugal
e-mail: balsa@ipb.pt

M. Daydé · D. Ruiz
IRIT, Universitée de Toulouse, CNRS, INPT, Toulouse, France
e-mail: dayde@enseeiht.fr

D. Ruiz
e-mail: ruiz@enseeiht.fr

J. M. L. M. Palma
Faculdade de Engenharia da Universidade do Porto (FEUP), Porto, Portugal
e-mail: jpalma@fe.up.pt



A. A. Pinto and D. Zilberman (eds.), *Modeling, Dynamics, Optimization and Bioeconomics III*, Springer Proceedings in Mathematics & Statistics 224,
https://doi.org/10.1007/978-3-319-74086-7_3

## 1 Introduction

We are interested in the computation of a *near*-invariant subspace associated with the smallest eigenvalues of a given symmetric and positive definite matrix, with a type of inexact subspace iteration method that exploits only matrix vector products. To this end, we exploit an algorithm, called BlockCGSI, which combines the inverse subspace iteration (SI), see [1] for instance, with a stabilized version of the block Conjugate Gradient algorithm (blockCG) [2, 3] to solve iteratively the set of multiple linear systems in each inverse iteration. The implicit use of the inverse of the coefficient matrix by means of an iterative solution (inner iteration), is suitable for large scale problems, where traditionally the factorization of the matrix is difficult to achieve, or when the matrix itself is not explicitly available. However, it also introduces an error - when computing the approximate solutions - that may affect the linear convergence of the inverse iteration (the outer iteration). The difficulty is to find an appropriate stopping threshold for the iterative method (in the inner iteration) that enables a suitable convergence of the inverse iteration and, if possible, that minimizes the computational work.

The central part in this work is to propose a way to monitor effectively the convergence of this inner-outer type of iterative scheme. We therefore analyze, from a geometrical point of view, the convergence of the inverse subspace iteration combined with the blockCG inner solver, and we derive an expression that relates the two residual norms used in the two iteration levels. From this, we can extract a residual measure for the blockCG that is directly linked with the convergence of the outer process toward the desired eigenvectors, and propose a stopping threshold parameter that minimizes the total amount of computational work to achieve some targeted accuracy.

In Sect. 2, we recall some important properties of the subspace iteration and of the inexact inverse iteration that are the basic components of the BlockCGSI algorithm. We also introduce some algorithmic techniques to improve the method. In particular, we exploit Chebyshev polynomials as a spectral filtering tool when building the starting vectors, and we introduce the concept of *"sliding window"* as an algorithmic feature for the computation of a *near*-invariant subspace of any dimension. Section 3 is dedicated to the analysis of the convergence properties. In particular, we explain how one should monitor the convergence of the blockCG in conjunction with the global convergence of the inverse iteration. The study presented in Sect. 3 follows on from our initial work on the monitoring of BlockCGSI presented in [4]. We finish the analysis of the BlockCGSI algorithm in Sect. 4, with a review of its main algebraic operations and computational costs.

The combination of the inverse subspace iteration with the block Conjugate Gradient (BlockCGSI algorithm) was initially exploited in the experimental study by [5], and used to deflate the initial residual in consecutive runs of the CG algorithm. This is of particular interest in the simulation of time dependent partial differential

equations, where at each global iteration (or time step) there are several systems with the same spectral properties to be solved.

Following this work, and to illustrate the effectiveness of the monitoring that we propose, we devote the last two sections to some numerical simulations where we first perform some partial spectral decomposition, and exploit this information to improve the convergence in the iterative solution of the following linear systems. In the case of Symmetric Positive Definite (SPD) matrices, several solutions have already been proposed to improve the convergence of Conjugate Gradient (CG) algorithm (see [6], for instance). In Sect. 5, we highlight two of these techniques, namely the deflation of the initial residual and the Spectral Low Rank Update (SLRU) preconditioner [7]. Section 6 is then concerned with the numerical results illustrating the proposed monitoring strategy for the BlockCGSI Algorithm as well as the potential of the pre-computed spectral information to accelerate the convergence of the Conjugate Gradient. We finish in Sect. 7 with some concluding remarks.

## 2 The Subspace (Inverse) Iteration

Subspace inverse iteration, or simply subspace iteration, is a generalization of the inverse iteration, where a set of vectors is multiplied consecutively by the inverse o a matrix instead of just one vector.

Consider the symmetric and positive definite matrices $A$ and the matrix of the wanted eigenvectors $U = [u_1, u_2, \ldots, u_s]$, where $Au_j = \lambda_j u_j$, $j = 1, \ldots, s$. Let $Z$ be a matrix whose columns generate a subspace of dimension $s$. If we multiply $Z$ successively by $A^{-1}$ we will generate a sequence $\{A^{-k}Z : k = 0, 1, 2, \ldots\}$ that converges to $U$. Defining the error angle between the approximated subspace generated by the columns of the matrix $A^{-k}Z$ and some specific eigenvector $u_j$ as

$$\varphi_j^{(k)} \equiv \angle(u_j, A^{-k}Z) \equiv \min \angle(u_j, q) \qquad \text{over} \qquad q \in A^{-k}Z, \tag{1}$$

it is proved in [1, p. 333] that, under certain assumptions, each eigenvector $u_j$, $j \leq s$, satisfies

$$\tan\left(\varphi_j^{(k)}\right) \leq \left(\frac{\lambda_j}{\lambda_{s+1}}\right)^k \tan\left(\varphi_j^{(0)}\right). \tag{2}$$

Normally, after one (or several) multiplication by the matrix $A^{-1}$, the column vectors from the resulting matrix $Q^{(k)} = A^{-k}Z$ are orthonormalized. This simplest version of the subspace iteration is also called orthogonal iteration.

In order to get an optimal approximation to each individual eigenvector $u_j$, using all the information in the basis $Q$, the orthogonal iteration is normally followed by the Ritz (or Rayleigh-Ritz) acceleration. The resulting Ritz values $diag(\Delta) = \delta_1, \ldots, \delta_s$ and Ritz vectors $V = [v_1, v_2, \ldots, v_s]$ are approximations to the eigenpairs in $A$ corresponding to the eigenvalues in the range $]0, \lambda_s]$. In [1, p. 334], it is indirectly

proved that $v_j^{(k)}$ converges linearly to $u_j$ by proving that $v_j^{(k)} \longrightarrow q_j^{(k)}$ at the same asymptotic rate $\lambda_j/\lambda_{s+1}$ that $q_j^{(k)}$ converges to $v_j^{(k)}$.

By (2), we can see that the reduction of the error angle in the subspace iteration is proportional to $\lambda_j/\lambda_{s+1}$ instead of $\lambda_s/\lambda_{s+1}$ as it is the case in the inverse iteration. This convergence property shows that if $\lambda_j$ is well separated from $\lambda_{s+1}$, we can have a good estimation of the eigenvalue $u_j$ in a few number of inverse iterations. In some cases, it can be useful to increase the block size $s$ just to benefit from a better gap between $\lambda_j$ and $\lambda_{s+1}$. This technique is also denoted as the use of *"Guard Vectors"*, e.g. extra vectors that are incorporated just to increase the rate of convergence in (2).

Compared with other reliable Lanczos algorithms, the subspace iteration just needs to store the current set of $s$ approximated eigenvectors (Ritz Vectors). The previous vectors of the Krylov sequence are discarded. This can be an important advantage if we have to work with a low memory storage and with slow convergence. The main difficulty in the subspace iteration is that we must set a priori the working block size $s$. The block size defines the dimension of the targeted invariant subspace and also, as we can verify by (2), the convergence rate. As we don't know the eigenvalue distribution, the chosen block size can lead to a very slow convergence or, if the gap between the $s$ wanted eigenvalues and the others is large, to a fast convergence where only a few subspace iterations will be needed for convergence. In this work we will also suggest a dynamical way to set up the block size $s$ without any information a priori about the spectrum of the matrix $A$.

In the subspace iteration the Ritz values $\delta_j$ converge faster to their limit (the eigenvalue $\lambda_j$) than the corresponding Ritz vectors $v_j$ to the eigenvector $u_j$. This means that one can have a converged Ritz value even if the corresponding Ritz vector is far from the wanted eigenvector. We can determine how close a Ritz vector $v_j$ is close from the corresponding eigenvector $u_j$ through the following error angle measure given by [1]

$$|\sin \angle(v_j, u_j)| \leq \frac{||Av_j - \delta_j v_j||_2}{gap}, \tag{3}$$

where $gap = \min\{|\lambda_{j-1} - \lambda_j|, |\lambda_j - \lambda_{j+1}|\}$. In practice we can not use (3) to monitor the convergence because the $gap$ is unknown, but when the Ritz values have converged, we can use the $\delta_j$'s to approximate the $gap$ and thus obtain a computable estimate of the error angle. However, for clustered eigenvalues, the spectral residual can be a bad measure because $v_j$ can approximate another eigenvector different from $u_j$, and the formula (3) will not be reliable (see [1]). The error measure (3) also indicate that the angle between a Ritz vector $v_j$ and the corresponding eigenvalue $u_j$ is directly proportional to the invariance measure $||Av_j - \delta_j v_j||_2$.

In each subspace iteration a linear system with $s$ right-hand sides is solved either by factorizing the coefficient matriz $A$ or with an iterative solver. If it is solved iteratively, the error introduced by the inexact solution of the linear system may affect the convergence rate of the subspace iteration given by (2). The difficulty is to find an appropriate stopping threshold for the iterative method that enables a suitable convergence of the inverse iteration and, if possible, that minimizes the

global computational work. In the next section we proceed to an overview of the main ideas about this problematic in the case of working with a single vector (block size reduced to one), in which case the process is called inexact inverse iteration.

## 2.1 The Inexact Inverse Iteration

In the inverse iteration (subspace iteration with block size $s = 1$), the system of linear equation is traditionally solved through the factorization of the matrix $A$. This can be expensive or impractical if $A$ has large dimension. Alternatively, we can solve these systems by an iterative method. Iterative methods are attractive in large scale problems because they require modest memory storage and the coefficient matrix does not need to be known explicitly, but just the result of it multiplication with any vector.

The inexact inverse iteration (see for instance [8]) includes two levels of iterations. One is the outer iteration, and corresponds to the main loop of the inverse iteration. The other is the inner iteration and corresponds to the iterative solution of the linear system in each outer iteration. The convergence of inexact inverse iteration is not yet perfectly well understood in all details, but has nevertheless been analyzed in several recent contributions. In this Section, we present a short survey of the main ideas exposed in some of these papers.

In [8, 9], for instance, it is proved that inexact inverse iteration can converge linearly at the same rate as the exact case even if the system is not solved accurately, and it is given some practical ways to choose the inner stopping threshold.

More recently in [10], a general convergence analysis of the correlation between the convergence rate and the threshold parameter is shown. It is proved that with some specific error measure, that if the threshold parameter is larger or equal to the ratio between the two smallest eigenvalues, the inexact inverse iteration converges linearly with a convergence rate directly given by the threshold parameter.

In [11], it is proved that it is worth continuing the inner loop until the norm of the solution vector stagnates. The growth of this norm is indeed directly linked to the reduction the eigenvalue residual norm in the inverse iteration. Consequently, the authors suggest a stopping criterion for the inner iteration based on the observation of the stagnation of the norm of approximate solution. Following this recommendation it is highlighted in [12] that this strategy is quite sensitive to the choice of the tolerance that measures this stagnation, as opposed to a strategy based on the measure of the standard relative residual of the system.

Similarly, the combination of the Jacobi–Davidson method with the Conjugate Gradient method as the inner solver has also been studied in [13]. It is established, from an analytical point of view, a relation between the reduction of the inner residual norm and the convergence of the outer process that allows an optimal stopping criterion for the inner iteration.

In most of these inexact inverse iteration analysis, the monitoring of the spectral error, in the outer iteration, is performed through some type of invariance measure of the approximated eigenvector $v_1$, like for instance

$$||Av_1 - \delta_1 v_1||_2 \leq \varepsilon, \tag{4}$$

where $\delta_1$ is an approximation to the eigenvalue corresponding to $v_1$, like for instance the corresponding Ritz value. By Eq. (3) we know that this measure enables to control indirectly the error angle between $v_1$ from the corresponding eigenvector $u_1$. Note also that if $v_1$ is not orthonormalized the error measure (4) must be divided by $||v_1||_2$.

## 2.2 The BlockCGSI Algorithm

In this section, we present and detail partly the BlockCGSI algorithm (Algorithm 1) used to compute an M-orthonormal basis, represented by matrix $W$, of a *near*-invariant subspace associated with the smallest eigenvalues in the preconditioned matrix $M^{-1}A$, where $M$ and $A$ are both symmetric and positive definite. If this eigenspace incorporates, for instance, all the eigenvalues of $M^{-1}A$ in the range $]0, \mu[$, we can expect, when using it later as a second level of preconditioning, that the condition number of the coefficient matrix will be reduced to about $\kappa = \lambda_{\max}/\mu$ (where $\lambda_{\max}$ is the largest eigenvalue in $M^{-1}A$). In Algorithm 1, $\lambda_{\max}$ and $\mu$ are considered as input parameters. However there is no specific need to know exactly the largest eigenvalue, and some upper bound on $\lambda_{\max}$ is sufficient, provided it gives some rough estimation of the actual 2-norm of $M^{-1}A$. In our experiments, we simply set $\lambda_{\max} = 1$.

Another input concerns the choice of the block size $s$ that defines the dimension of the working subspace at each inverse iteration. In the basic version of the inverse subspace algorithm, this also sets the number of approximated eigenvalues and eigenvectors at the end. Finally, it also gives the number of right-hand sides and solution vectors of the multiple linear systems solved by the blockCG algorithm at each inverse iteration, and therefore the amount of memory required as working space.

As a starting point, the algorithm requires the generation of an M-orthonormal matrix $W$ of dimension $s$; the closer are these column vectors to the targeted *near*-invariant subspace, the faster the convergence of the inverse iteration will be. The scope of steps 1 to 4, in Algorithm 1, is to generate an initial M-orthonormal set $V^{(0)}$ of $s$ vectors with eigencomponents corresponding to eigenvalues in the range $[\mu_f, \lambda_{\max}]$ below some predetermined value $\xi \ll 1$ (denoted as the *filtering level*). This filtering technique is based on Chebyshev polynomials (step 3) and is detailed in Sect. 2.4.

As we have seen, the essence of the inverse subspace iteration is the orthogonal iteration. It consists in multiplying a set of vectors by $A^{-1}M$ and M-orthonormalizing it in turn. If $W^{(k-1)}$ (initially empty) contains the set of vectors that have already converged at inverse iteration $(k-1)$, the current subspace $Q^{(k)}$, in step iii, should converge gradually to a *near*-invariant subspace that is M-orthogonal to $W^{(k-1)}$. In step i, the multiplication by $A^{-1}M$ is performed implicitly through the iterative solution of the system $M^{-1}AZ^{(k)} = V^{(k-1)}$ via the blockCG solver. In order to reduce the computational costs, this system is solved with an accuracy determined by the residual threshold $\varepsilon$. The appropriate choice of $\varepsilon$ is detailed in Sect. 3.3.

ALGORITHM 1: BLOCKCGSI WITH SLIDING WINDOW

**Inputs:** $A$, $M = R^T R \in \mathbb{R}^{n\times n}$, $\mu$, $\lambda_{max} \in \mathbb{R}$, $s \in \mathbb{N}$
**Output: a *near*-invariant subspace $W$ associated with all eigenvalues of $M^{-1}A$ in the range** $]0, \mu]$

**Begin**

**Generate the initial subspace (with filtering)**

1. $Z^{(0)}$ =RANDOM$(n, s)$
2. $V^{(0)} = Z^{(0)}\Gamma$ such that $V^{(0)^T} V^{(0)} = I_{s\times s}$
3. $Q^{(0)}$=Chebyshev-Filter$(V^{(0)}, \xi, [\mu_f, \lambda_{\max}], A, R)$
4. $V^{(0)} = R^{-1} Q^{(0)}\Gamma$ such that $V^{(0)^T} M V^{(0)} = I_{s\times s}$
5. $W^{(0)}$ = empty
6. **For** $k = 1, ...,$ **until converge Do:**

**Orthogonal iteration**

i. Solve $M^{-1}AZ^{(k)} = V^{(k-1)}$ with BlockCG
ii. $P^{(k)} = Z^{(k)} - W^{(k-1)} W^{(k-1)^T} M Z^{(k)}$
iii. $Q^{(k)}\Gamma_k = P^{(k)}$ such that $Q^{(k)^T} M Q^{(k)} = I_{s\times s}$
iv. $Q^{(k)} = [W^{(k-1)}\ Q^{(k)}]$

**Ritz acceleration**

v. $\beta_k = Q^{(k)^T} A Q^{(k)}$
vi. Diagonalize $\beta_k = U_k \Delta_k U_k^T$
where $U_k^T = U_k^{-1}$
and $\Delta_k$ =Diag$(\delta_1, ..., \delta_{p+s})$ (Ritz Values)
vii. $V^{(k)} = Q^{(k)} U_k$ (Ritz Vectors)

***"Sliding window"***

viii. $W^{(k)}$ =converged columns of $V^{(k)}$
ix. $V^{(k)}$ =non-converged columns of $V^{(k)}$
x. $(n, p)$ =size$(W^{(k)})$
xi. Update the computational window $(V^{(k)})$
xii. $(n, s)$ =size$(V^{(k)})$

7. **EndDo**

**End**

In step ii, the approximate solution vectors $Z^{(k)}$ are then projected onto the orthogonal complement of the converged vectors $W^{(k-1)}$, in order to remove the influence of eigencomponents associated with the already converged eigenvalues. The set of projected vectors $P^{(k)}$ is then M-orthonormalized (step iii), and gathered together with $W^{(k-1)}$ in the matrix $Q^{(k)}$.

The orthogonal iteration is followed by the Ritz acceleration (steps v to vii). The spectral information contained in $Q^{(k)}$ is thus redistributed in the column vectors of $V^{(k)}$ that will contain separately better approximations of each individual eigenvector in the targeted invariant subspace. Steps v and vi give the Ritz values $diag(\Delta) = \delta_1, \ldots, \delta_{p+s}$ ranged in increasing order, where $p$ is the dimension of $W^{(k-1)}$, i.e. the number of converged vectors in the inverse iteration $(k-1)$, and $s$ is the current block size. The Ritz values and Ritz vectors $V = [v_1, v_2, \ldots, v_p, \ldots, v_{p+s}]$ are approximations to the eigenpairs in $M^{-1}A$ corresponding to the eigenvalues in the range $]0, \lambda_{p+s}]$. The convergence rate of each individual non-converged Ritz vector is then of order $\lambda_i/\lambda_{p+s+1}$, with $p+1 \leq i \leq p+s$.

The end of the BlockCGSI algorithm consists in testing the convergence and updating the computational window. In step viii, all the Ritz vectors that are considered as *near*-invariant, with respect to the given accuracy, are assigned to $W^{(k)}$. More details about the monitoring of the convergence are given in Sect. 3.1. Step xi consists in the update of the current set of vectors $V^{(k)}$. This algorithmic issue in the BlockCGSI algorithm is denoted as *"sliding window"* and detailed in Sect. 2.5.

## 2.3 Improvements of the BlockCGSI Algorithm

In this section, we describe briefly the two techniques incorporated in the BlockCGSI algorithm to improve the convergence: the Chebyshev based filtering technique at step 3 and the *"sliding window"* at step xi in Algorithm 1.

## 2.4 Chebyshev Based Filtering Technique

The purpose of the Chebyshev based filtering technique is to bring the randomly generated set of $s$ starting vectors closer to the eigenvectors corresponding to $s$ smallest eigenvalues. Chebyshev polynomials in $M^{-1}A$ are used to *damp* the eigenfrequencies associated with all the eigenvalues in the range $[\mu_f, \lambda_{max}]$, in the sense that those eigencomponents associated to all eigenvalues in this range are reduced to about 0, and the other ones are left close to their original value. We summarize here the outline of this technique, and for details, we refer to [6].

In the BlockCGSI algorithm, the application of the Chebyshev polynomial in $M^{-1}A$ to the set of starting vectors $V$ is denoted as

$$Q = \text{Chebyshev-Filter}(V, \xi, [\mu_f, \lambda_{\max}], A, R),$$

with $M = R^T R$. This step can also be expressed formally by

$$Q = \mathscr{F}_m(M^{-1}A)V,$$

where $\mathscr{F}_m$ is a polynomial function of degree $m$ given by

$$\mathscr{F}_m(\lambda) = \frac{T_m(w(\lambda))}{T_m(w(0))},$$

with $T_m$ the usual Chebyshev polynomial of degree $m$ and $w(\lambda)$ the mapping function that brings $\mu_f$ to 1 and $\lambda_{\max}$ to $-1$.

After the filtering process, the vectors $q_j = \mathscr{F}_m(M^{-1}A)v_j$ will have eigencomponents equal to $\mathscr{F}_m(\lambda_l)\zeta_j$, with $\zeta_j = \langle v_j, u_l \rangle, l = 1, \ldots, n$, and

$$\mathscr{F}_m(\lambda_l) \in \begin{cases} [-\xi, \xi] & \text{if } \lambda_l \in [\mu_f, \lambda_{\max}] \\ [\xi, 1] & \text{if } \lambda_l \in ]0, \mu_f[, \end{cases}$$

where $\xi$, the filtering level, is chosen a priori much lower than 1 in order to make the eigencomponents corresponding to the eigenvalues in the range $[\mu_f, \lambda_{\max}]$ close to 0.

The Chebyshev polynomial $T_m$ is computed implicitly by a recurrence formula from the two previous values $T_{m-1}$ and $T_{m-2}$ ($m \geq 2$). At each update, it requires that a set of $s$ vectors be multiplied by $M^{-1}A$. For given values of $\mu_f$, $\lambda_{\max}$ and $\xi$, the degree $m$ depends on the ratio $\lambda_{\max}/\mu_f$ and is inversely proportional to $\xi$.

The reason behind the use of these Chebyshev filters at the starting point is to put the inverse subspace iteration in the situation of working directly in the orthogonal complement of a large number of eigenvectors, e.g. all those associated with the eigenvalues in the range $[\mu_f, \lambda_{\max}]$. Obviously, there is some compromise to achieve, in the sense that a very small value of $\mu_f$ will minimize the number of inverse iterations but will increase strongly the computational efforts in the Chebyshev filtering step.

## 2.5 Sliding Window

The original version of the BlockCGSI algorithm computes a fixed number $s$ of approximated eigenvectors associated to the $s$ smallest eigenvalues in the iteration matrix. The difficulty when choosing the parameter $s$ is that we do not know a priori the distribution of the eigenvalues, and consequently how many eigenvalues we need approximate to reduce substantially the condition number. A too small block size $s$ can lead to a non effective improvement in the convergence rate of the iterative solver in the following runs, whereas a too large block size $s$ may induce unnecessary extra computational work. To circumvent this problem, we have included the possibility of enlarging the size of the *near*-invariant subspace along with the inverse iterations, as well as changing the block size $s$ whenever appropriate. The idea is to start the algorithm with a block size $s$ determined only on the basis of computer aspects like, for instance, the efficiency of Level-3 BLAS [14] internal kernels (used in the BlockCG algorithm), or the memory requirements. Then, when computing the Ritz values and checking the invariance of the Ritz vectors, at the end of each inverse iteration, we can decide how to adapt effectively the actual number of approximated eigenvectors.

In practice, when one or more of the $s$ Ritz vectors in the current set $V^{(k)}$ are detected as *near*-invariant, these vectors are moved to the set of converged vectors $W^{(k)}$ (step viii of Algorithm 1), and there remains open the choice of incorporating new vectors to replace these ones to form the current block of $s$ working vectors $V^{(k)}$, or to reduce the block size $s$ (step xi of Algorithm 1). Incorporating new vectors is appropriate until the approximated eigenvalues cover a sufficiently large interval $[0, \mu]$ for an effective reduction of the condition number. When this target is met, it is then possible to reduce the block size $s$ until all the targeted Ritz vectors have converged. However if we detect a gap in the actual range of the approximated eigenvalues, it can also be useful to keep the block size unchanged and to make use of the extra vectors to accelerate the convergence of the targeted ones in the inverse iteration. Effectively, with the *sliding window* the convergence rate given by (2) is now proportional to $\lambda_j/\lambda_{p+s+1}$, where $p$ is the number of converged Ritz vectors.

Another issue in this algorithm comes from the fact that the solution of the linear systems in each inverse iteration are not obtained with high accuracy. Indeed, our purpose is to stop the blockCG as soon as possible. Consequently, it can happen that some of the Ritz values converge first to internal eigenvalues, before the smaller ones are actually discovered. In this case, some of the already converged Ritz vectors may appear as not enough invariant after the discovery of the extreme eigenvalues, and it may therefore be necessary to enlarge the block size $s$ and refine furthermore these vectors. This risk of seeing internal eigenvalues coming first is the reason why it is important to systematically incorporate the assumed converged vectors $W^{(k-1)}$ when recomputing the Ritz pairs (step iv of Algorithm 1). This is the only way to ensure the appropriate redistribution of the eigencomponents within each approximate eigenvector in the long run.

The update of the computational window is mentioned at step xi of algorithm 1. The operation that consists in introducing new vectors, after a set of $\ell$ Ritz vectors has converged, is detailed in Algorithm 2.

ALGORITHM 2: INCORPORATE NEW VECTORS

**Inputs:** $A$, $M = R^T R \in \mathbb{R}^{n\times n}$, $V^{(k)} \in \mathbb{R}^{n\times(s-\ell)}\mu_f$, $\lambda_{max} \in \mathbb{R}$, $\ell \in \mathbb{N}$

**Begin**

a) $P$ =RANDOM($n$, $\ell$)

b) $P = Q\Gamma$ such that $Q^T Q = I_{\ell\times\ell}$

c) $P$ = Chebyshev-Filter($Q$, $\xi$, $[\mu_f, \lambda_{\max}]$, $A$, $R$)

d) $Q = R^{-1} P\Gamma$ such that $Q^T M Q = I_{\ell\times\ell}$

e) $P = Q - W^{(k)} W^{(k)^T} M Q$

f) $V^{(k)} = [V^{(k)} P]$

**End**

It begins by generating randomly the new vectors and filtering them, as in the starting steps of the BlockCGSI algorithm (steps a, b and c). After that the vectors are projected in the M-orthogonal complement of the converged ones (step d and e), in order to remove the correspondent eigencomponents. The remaining operations (step f) adjust the block size $s$ with respect to the current set of working vectors $V^{(k)}$.

## 3 Convergence Analysis

The BlockCGSI algorithm involves two iterative loops: the first, that we also denote as the outer iteration, at step 6 corresponds to the inverse iteration, and the second loop, or inner iteration, is in the call to the blockCG algorithm (at step i in Algorithm 1) for the iterative solution of the linear system with multiple right-hand sides, $M^{-1}AZ^{(k)} = V^{(k-1)}$. These two iterations levels require each some specific stopping criterion in order to monitor the convergence of the algorithm. Sections 3.1 and 3.2 are devoted to and analyze some properties associated with these aspects. In Sect. 3.3, we propose a way to link the monitoring of the convergence in the inner loop with the measure of the convergence in the outer loop.

### *3.1 Subspace Inverse Iteration (Outer Loop)*

At each inverse iteration $(k)$ in Algorithm 1, the blockCG algorithm solves the $s$ linear systems $M^{-1}Az_j^{(k)} = v_j^{(k-1)}$, $j = 1, \ldots, s$, where the matrix $A$ is preconditioned with a symmetric and positive definite preconditioner, $M = R^T R$. The symmetrized system can be written as usual as

$$R^{-T}AR^{-1}Rz_j^{(k)} = Rv_j^{(k-1)} \Longleftrightarrow \tilde{A}\tilde{z}_j^{(k)} = \tilde{v}_j^{(k-1)}, \qquad j = 1, \ldots, s. \tag{5}$$

where $\tilde{A} = R^{-T}AR^{-1}R, \tilde{z}_j = Rz_j$ and $\tilde{v}_j = Rv_j$. For simplicity we will omit to repeat that $j$ varies from 1 to $s$. We will consider that the subscript $j$ refers to the position of the correspondent eigenvalue in the current working set. The superscript $(k)$ denotes the inverse iteration number, and the tilde refers to the symmetrized system (5).

The outer iteration produces a sequence of Ritz vectors $\tilde{v}_j^{(1)}, \tilde{v}_j^{(2)}, \ldots, \tilde{v}_j^{(k)}$, that converge to the eigenvector $\tilde{u}_j = Ru_j$ corresponding to the eigenvalue $\lambda_j$ of both matrices $\tilde{A}$ and $A$. At the outer iteration $(k)$, the vectors $\tilde{v}_j^{(k)}$ are orthonormal, while the vectors $v_j^{(k)}$ (columns of matrix $V^{(k)}$, in step vii of Algorithm 1) are M-orthonormal. The corresponding Ritz values (diagonal elements of the matrix $\Delta^{(k)}$) are given by $\delta_j{}^{(k)} = v_j{}^{(k)^T} Av_j^{(k)} = \tilde{v}_j^{(k)T}\tilde{A}\tilde{v}_j^{(k)}$.

As we have seen in Sect. 2, we can evaluate indirectly the error angle (between $v_j$ and the corresponding eigenvector $u_j$) through the measure

$$\frac{||\tilde{A}\tilde{v}_j^{(k)} - \delta_j^{(k)}\tilde{v}_j^{(k)}||_2}{||\tilde{v}_j^{(k)}||_2} = ||M^{-1}Av_j^{(k)} - \delta_j^{(k)}v_j^{(k)}||_M. \tag{6}$$

Dividing (6) by $\delta_j^{(k)}$ (as an approximation of $\lambda_j$) we obtain a relative invariance measure that is used in step viii of Algorithm 1 to decide if a Ritz vector $v_j^{(k)}$ has converged or not, as for instance when

$$\frac{||M^{-1}Av_j^{(k)} - \delta_j^{(k)} v_j^{(k)}||_M}{\delta_j^{(k)}} \leq \varepsilon_{\text{outer}}, \tag{7}$$

if a certain tolerance $\varepsilon_{\text{outer}}$ is enough. Since $\varepsilon_{\text{outer}}$ is fixed, the error angle will be smaller for the Ritz vectors corresponding to the smallest eigenvalues because, in these cases, the invariance measure (6) will be divided by a smaller values $\delta_j$. We use the stopping criterion (7) because, as we will see in Sect. 5, the Ritz vectors are used to improve the convergence of the CG algorithm, and this measure of *near*-invariance in the Ritz vectors is indeed very appropriate in that respect. Note also that (7) is also used in the same context by [11].

### 3.2 The BlockCG Iteration (Inner Loop)

The block Conjugate Gradient (blockCG) algorithm under concern is a numerically stable variant [3] that avoids the numerical problems that can occur when some of the $s$ systems are about to converge. It solves simultaneously the $s$ linear systems from Eq. (5). For each system, $j = 1, .., s$, it produces a sequence of vectors $\tilde{z}_j^{[i]}$, giving, after convergence, the approximate solution $\tilde{z}_j^{(k)}$ used in the $k$th inverse iteration,

$$\tilde{z}_j^{[1]}, \tilde{z}_j^{[2]}, \ldots, \tilde{z}_j^{[i]} \rightarrow \tilde{z}_j^{(k)}, \tag{8}$$

where the superscript $[i]$ stands for the blockCG (inner) iteration number.

The residual vector associated with each iterate $\tilde{z}_j^{[i]}$ is

$$\tilde{r}_j^{[i]} = \tilde{v}_j^{(k-1)} - \tilde{A}\tilde{z}_j^{[i]}. \tag{9}$$

We also introduce another vector which we will use to measure the proximity of the current iterate $\tilde{z}_j^{[i]}$ from the corresponding eigenvector $\tilde{u}_j$,

$$\tilde{S}_j^{[i]} = \tilde{A}\tilde{z}_j^{[i]} - \tilde{\delta}_j^{[i]}\tilde{z}_j^{[i]} = \tilde{v}_j^{(k-1)} - \tilde{\delta}_j^{[i]}\tilde{z}_j^{[i]} - \tilde{r}_j^{[i]}, \tag{10}$$

where $\tilde{\delta}_j^{[i]} = \tilde{z}_j^{[i]T}\tilde{A}\tilde{z}_j^{[i]} / \tilde{z}_j^{[i]T}\tilde{z}_j^{[i]} = \delta_j^{[i]}$ is the Rayleigh quotient corresponding to the current iterate $\tilde{z}_j^{[i]}$. The error measure (6) applied on the symmetrized system, at each inner iteration $[i]$, yields

$$\frac{||\tilde{S}_j^{[i]}||_2}{||\tilde{z}_j^{[i]}||_2} = \frac{||\tilde{v_j}^{(k-1)} - \delta_j^{[i]}\tilde{z}_j^{[i]} - \tilde{r}_j^{[i]}||_2}{||\tilde{z}_j^{[i]}||_2}. \tag{11}$$

Additionally, if we start the blockCG iteration with $\tilde{z}_j^{[0]} = 0$, at each iteration the current residual $\tilde{r}_j^{[i]}$ remains orthogonal to both $\tilde{v_j}^{(k-1)}$ (the right-hand side) and $\tilde{z}_j^{[i]}$ (linear

combination of the current Krylov vectors). Thus, $\tilde{r}_j^{[i]T}(\tilde{v_j}^{(k-1)} - \delta_j^{[i]}\tilde{z}_j^{[i]}) = 0$, and we can write

$$||\tilde{v_j}^{(k-1)} - \delta_j^{[i]}\tilde{z}_j^{[i]} - \tilde{r}_j^{[i]}||_2^2 = ||\tilde{v_j}^{(k-1)} - \delta_j^{[i]}\tilde{z}_j^{[i]}||_2^2 + ||\tilde{r}_j^{[i]}||_2^2. \tag{12}$$

Finally, if we translate the previous properties to the non-symmetrized system,

$$\frac{||M^{-1}Az_j^{[i]} - \delta_j^{[i]}z_j^{[i]}||_M}{||z_j^{[i]}||_M} = \sqrt{\frac{||v_j^{(k-1)} - \delta_j^{[i]}z_j^{[i]}||_M^2}{||z_j^{[i]}||_M^2} + \frac{||r_j^{[i]}||_M^2}{||z_j^{[i]}||_M^2}} \stackrel{def}{=} \sqrt{\phi_j^{[i]2} + \omega_j^{[i]2}}, \tag{13}$$

where we can see that the reduction of the invariance of each iterate $z_j^{[i]}$ during the inner loop depends on the relative residual measure $\omega_j^{[i]} = ||r_j^{[i]}||_M/||z_j^{[i]}||_M$ and on the value $\phi_j^{[i]} = ||v_j^{(k-1)} - \delta_j^{[i]}z_j^{[i]}||_M/||z_j^{[i]}||_M$.

Even if we expect that the backward error measure $\omega_j^{[i]}$ will decrease down to a level of small magnitude, the value of $\phi_j^{[i]}$ is more likely to stagnate on a higher level, depending on the proximity of the right-hand side $v_j^{(k-1)}$ from the correspondent eigenvector $u_j$. Therefore, the bound in (13) can be dominated by the value of $\phi_j^{[i]}$, and little improvement on the global convergence of the algorithm can be expected by further iterations in the blockCG.

We now investigate the asymptotic behavior of $\phi_j^{[i]}$, assuming that $z_j^{[i]}$ actually converges to $z_j^* = A^{-1}Mv_j^{(k-1)}$. Let us first introduce the asymptotic limit of the Rayleigh quotient $\delta_j^{[i]}$, $\delta_j^* = \langle z_j^*, v_j^{(k-1)}\rangle_M/||z_j^*||_M^2$, and the angle $\theta_j$ in the $M$-norm between $z_j^*$ and $v_j^{(k-1)}$, whose cosine is given by

$$\cos(\theta_j) = \frac{\langle z_j^*, v_j^{(k-1)}\rangle_M}{||z_j^*||_M||v_j^{(k-1)}||_M} = \delta_j^*||z_j^*||_M. \tag{14}$$

As a consequence of the M-orthonormalization of $v_j^{(k-1)}$, we can write

$$\sin(\theta_j) = ||v_j^{(k-1)} - \delta_j^* z_j^*||_M, \tag{15}$$

which is also the asymptotic limit of $||v_j^{(k-1)} - \delta_j^{[i]}z_j^{[i]}||_M$. Consequently, the asymptotic limit of the component $\phi_j^{[i]}$ in (13) is

$$\phi_j^{[i]} \xrightarrow[i\to\infty]{} \frac{\sin(\theta_j)}{||z_j^*||_M} = \delta_j^* \tan(\theta_j). \tag{16}$$

We can see that the asymptotic limit of $\phi_j^{[i]}$ depends only on the two vectors $v_j^{(k-1)}$ and $z_j^* = A^{-1}Mv_j^{(k-1)}$. It is also clear that, if $v_j^{(k-1)}$ is close to an eigenvector, the angle $\theta_j$ should be very small, as well as the corresponding asymptotic limit of $\phi_j^{[i]}$. With respect to the bound in (13), this allows more room for decreasing the backward error $\omega_j^{[i]}$ in the blockCG iteration. The strategy suggested by this analysis is to decrease the value of the stopping criterion in the blockCG (inner loop) along with the convergence of the inverse iteration (outer loop). This basic idea is further developed in the next section.

### 3.3 Stopping Criterion for the BlockCG

The stopping criterion for the blockCG defines the approximation degree of $\tilde{z}^{(k)} \approx \tilde{A}^{-1}\tilde{v}_j^{(k-1)}$ or equivalently of $z^{(k)} \approx A^{-1}Mv_j^{(k-1)}$. Its choice is crucial because demanding a high accuracy can lead to a great amount of unnecessary extra work, whereas an insufficient level of accuracy in the solution may deteriorate the convergence rate of the inverse iteration (given by (2)).

We propose to monitor only the convergence of the iterates $z_1^{[i]}$, corresponding to the smallest Ritz value $\delta_1^{(k-1)}$ in the previous inverse iteration. In general, this system needs more computational efforts to be solved accurately. As we have seen in the previous section, the relative residual in the inner loop is given by

$$\omega_1^{[i]} = \frac{||v_1^{(k-1)} - M^{-1}Az_1^{[i]}||_M}{||z_1^{[i]}||_M}, \tag{17}$$

and is readily available in the blockCG iteration (see [3]). Notice also that $\omega_1^{[i]}$ is very close to the usual Rigal–Gaches [15] backward error measure

$$\frac{||v_1^{(k-1)} - M^{-1}Az_1^{[i]}||_M}{||z_1^{[i]}||_M + 1},$$

using the fact that the M-norm of the current right-hand side $v_1^{(k-1)}$ equals 1, and assuming that the 2-norm of the preconditioned matrix $M^{-1}A$ is close to one.

In the outer loop, we monitor the accuracy of the approximated eigenvectors through the relative invariance, as indicated in (6) and (7). At inverse iteration $(k-1)$, we consider that a Ritz vector $v_j^{(k-1)}$ has converged when

$$\frac{||M^{-1}Av_j^{(k-1)} - \delta_j^{(k-1)}v_j^{(k-1)}||_M}{\delta_j^{(k-1)}} \leq \varepsilon_{\text{outer}}. \tag{18}$$

In order to satisfy (18) in the current inverse iteration ($k$), the stopping criterion in the blockCG is set as

$$\omega_1^{[i]} \leq \varepsilon_{\text{inner}}, \quad \text{with} \quad \varepsilon_{\text{inner}} = \varepsilon_{\text{outer}} \delta_1^{(k-1)}, \tag{19}$$

where $\delta_1^{(k-1)}$ is the smallest of the Ritz values corresponding to current set of non converged Ritz vectors. This stopping criterion is based on the decomposition (13), assuming that $\phi_1^{[i]}$ is not dominant and that the value of $w_1^{[i]}$ governs the absolute invariance measure in the inner iteration. This is surely the case when the Ritz vector $v_1^{(k-1)}$ is close to an eigenvector because, in this case, the value of $\tan(\theta_1)$ should be small. This can even occur at the first inverse iteration because the starting vectors are previously filtered with Chebyshev based polynomials. A second assumption, implicit in (19), is that the inner invariance measure, given by Eq. (13), will be close to the outer invariance measure, given by Eq. (6). This can be justified from the theoretical analysis given in [1], which shows that the Ritz vectors $V^{(k)}$ actually converge to the solution vectors $Z^{(k)}$ at the same rate of convergence of these $Z^{(k)}$ vectors to the set of targeted eigenvectors (see Sect. 2).

Under these assumptions, the idea in (19) is to achieve a given tolerance $\varepsilon_{\text{outer}}$ in (18) while minimizing the number of blockCG iterations. Note also that the strategy (19) for the stopping criterion is in agreement with other analysis of the inexact inverse iteration, like for instance in [8, 10, 12], where it is suggested to use a decreasing value for the inner tolerance $\varepsilon_{\text{inner}}$. This is the case, indeed, in (19) since the value of $\delta_1^{(k)}$ decreases gradually toward $\lambda_1$ along with the convergence of the outer iteration.

As we have seen in Sect. 3.2, when $\phi_1^{[i]}$ has reached its stagnation level defined in (16), the bound in (13) is then dominated by this asymptotic value, and there is no need to decrease any further the value of $\omega_1^{[i]}$. No more improvements on $v_1^{(k)}$ with respect to $u_1$ might be expected, and it is better to stop the blockCG iteration and to launch the next inverse iteration. This strategy is very close to the one proposed in [13] and, in some way, to the one also proposed in [11], because the stagnation of $\phi_1^{[i]}$ implies the stagnation of $||z_1^{[i]}||_M$ which is the basic argument used in this article to monitor the convergence.

The risk of having $\omega_1^{[i]}$ much smaller than $\phi_1^{[i]}$ during the blockCG iterations is also limited with a closer tolerance $\varepsilon_{\text{outer}}$ not too small, like for instance $10^{-1}$ or $10^{-2}$. These values are in general enough for the purpose of building a *near*-invariant subspace for preconditioning the solution of consecutive linear systems with the same coefficient matrix. However, if one is interested in computing an accurate invariant subspace, the inner threshold parameter $\varepsilon_{\text{inner}}$ in (19) should be set to the maximum between $\varepsilon_{\text{outer}} \delta_1^{(k-1)}$ and the asymptotic value of $\phi_1^{[i]}$ in (16). From the analysis in 3.2, this is indeed the maximum level of accuracy that it is reasonable to achieve in each blockCG run. Do not forget also that along with the convergence of the Ritz vectors towards the corresponding eigenvectors, the asymptotic value of $\phi_1^{[i]}$ in (16) tends to zero proportionally to the tangent of the angle $\theta_j$. Because of that, it is ensured that after some appropriate number of inverse (outer) iteration, the maximum between $\varepsilon_{\text{outer}} \delta_1^{(k-1)}$ and this asymptotic value of $\phi_1^{[i]}$ will always remain the first of these two, and the targeted accuracy in the inverse iteration can then be expected to be achieved afterward.

**Table 1** Basic operations counts in BlockCGSI algorithm

| Operation | Size | Flops | Symbol | BLAS level |
|---|---|---|---|---|
| $x = \text{RANDOM}$ | $n$ | $3n$ | $\mathscr{C}_{RAND}$ | – |
| $y \leftarrow y^T x$ | $n$ | $2n$ | $\mathscr{C}_{DOT}$ | 1 |
| $y \leftarrow y + \sigma x$ | $n$ | $2n$ | $\mathscr{C}_{AXPY}$ | 1 |
| $y = Ax$ | $n$ | $2nnz(A) - n$ | $\mathscr{C}_{A}$ | 2 |
| $x = M^{-1} y$ | $n$ | $4nnz(R) - 2n$ | $\mathscr{C}_{M}$ | 2 |
| $C \leftarrow C + \sigma VB$ | $n \times s$ | $2s^2 n$ | $\mathscr{C}_{GEMM}$ | 3 |
| $P = QR$ | $n \times s$ | $2s^2 n$ | $\mathscr{C}_{ORTHO(s)}$ | 1 |

## 4 Operations Counts

The precomputation of the basis $W$ of a *near*-invariant subspace with the BlockCGSI algorithm, has a cost which we denote by $\mathscr{C}_{BCGSI}$. For a fixed accuracy, this cost depends essentially on the dimension $q$ of the basis $W$ and on some working parameters like the block size $s$, the filtering level $\xi$ and the cut-off filtering value $\mu_f$. To be effective, the gains obtained in the acceleration of the convergence of the classical Conjugate Gradient algorithm must cover the extra cost for the computation of this spectral information. In Table 1 we present the costs in floating point operations (flops) associated with some basic operations performed in the BlockCGSI algorithm, as well as the corresponding BLAS level. The computational cost of each part in Algorithm 1 will be expressed as a function of these basic operations. We denote the computational cost of one operation $OP$ by the symbol $\mathscr{C}_{OP}$, like for instance $\mathscr{C}_A$ that is the cost of on sparse matrix-vector product, where the number of non-zeros elements of $A$ is given by $nnz(A)$. As mentioned before, a first level of left preconditioner $M$ is also used, which purpose is to cluster the spectrum of our iteration matrix. The cost of the multiplication of a vector by $M^{-1}$ is represented by $\mathscr{C}_M$. Since, $M$ is constructed in our experiments by means of the Incomplete Cholesky factorization or Jacobi scaling, its cost can be estimated as mentioned in Table 1, where $nnz(R)$ is the number of nonzero elements in the factor $R$ from the Incomplete Cholesky or Jacobi factorization.

In the beginning of the BlockCGSI algorithm, we apply the Chebyshev filtering polynomial in $A$ to bring the set of $s$ random generated vectors $V^{(0)}$ near the eigenvectors corresponding to the smallest eigenvalues. The cost of one Chebyshev iteration is $\mathscr{C}_{CHEBY} \approx s\mathscr{C}_A + s\mathscr{C}_M + 3s\mathscr{C}_{AXPY}$. Additionally, the starting vectors are orthonormalized before the filtering step and M-orthonormalized after. The computation of the starting vectors has a total cost given by

$$\mathscr{C}_{START} \approx s\mathscr{C}_{RAND} + 2\mathscr{C}_{ORTHO(s)} + \frac{s}{2}\mathscr{C}_M + \texttt{ChebIt} \times \mathscr{C}_{CHEBY}, \qquad (20)$$

where `ChebIt` is the total number of Chebyshev filtering iterations.

The QR iteration represents the most expensive step in Algorithm 1 in terms of computational cost. It consists of the iterative solution of the system with $s$ right-hand sides using the stabilized blockCG solver. The cost of each inner iteration in the blockCG is:

$$\mathscr{C}_{bCG} \approx s\mathscr{C}_A + s\mathscr{C}_M + 3\mathscr{C}_{GEMM} + 2\mathscr{C}_{ORTHO(s)}. \tag{21}$$

After the blockCG run, the solution vectors $Z^{(k)}$ are projected onto the M-orthogonal complement of the converged Ritz vectors $W^{(k-1)}$ (of dimension $p$, that varies from 0 to $q-1$), and are M-orthonormalized. The estimation of the operations count corresponding to the steps included in the QR iteration at each inverse iteration resumes in:

$$\mathscr{C}_{QR} \approx \texttt{bCGIt(k)} \times \mathscr{C}_{bCG} + s\mathscr{C}_{PROJ(p)} + \mathscr{C}_{ORTHO(s)} + \mathscr{C}_M, \tag{22}$$

where `bCGIt(k)` is the number of blockCG inner iterations performed at inverse iteration $(k)$, and where $\mathscr{C}_{PROJ(p)} = \mathscr{C}_M + p\mathscr{C}_{DOT} + \mathscr{C}_{AXPY}$. The cost of the QR iteration cannot be determined a priori, because the parameters `bCGIt(k)` and $p$ change from one inverse iteration to the other in a non-deterministic way. Therefore, we trace their actual values as the algorithm progresses, and we compute the total number of flops at the end. We proceed in the same way in the Ritz acceleration with the size of $Q^{(k)}$, given by $s+p$. The cost of one Ritz acceleration is given by

$$\mathscr{C}_{RITZ} \approx (s+p)\mathscr{C}_A + 2(s+p)^2\mathscr{C}_{DOT} + 5(s+p)^3, \tag{23}$$

where $5(s+p)^3$ is the cost corresponding to the spectral decomposition of the matrix $\beta_k$ at step vi in Algorithm 1.

The amount of work to update the computational window is induced by the convergence test, Eqs. (6) and (7), and by the incorporation of the new vectors (see Algorithm 2). If we maintain the same block size $s$, the $\ell$ converged vectors are replaced in the computational window by $\ell$ new vectors. After these new vectors are filtered and M-orthonormalized, as done with the initial set of starting vectors, they are finally projected onto the M-orthogonal complement of the converged ones. The cost of these steps is then given by

$$\mathscr{C}_{UPDATE} \approx \mathscr{C}_{INV} + \ell\mathscr{C}_{RAND} + \texttt{ChebIt} \times \mathscr{C}_{CHEBY} + 2\mathscr{C}_{ORTHO(\ell)} + \frac{\ell}{2}\mathscr{C}_M + \ell\mathscr{C}_{PROJ(p)}, \tag{24}$$

where

$$\mathscr{C}_{INV} \approx 2(s+p)\mathscr{C}_M + (s+p)\mathscr{C}_{AXPY} + 2(s+p)\mathscr{C}_{DOT} \tag{25}$$

is the cost spent when testing the invariance of all the Ritz vectors.

Finally, the estimate of the total number of floating point operations performed in the BlockCGSI algorithm, over all the inverse iterations `InvIt`, is

$$\mathscr{C}_{BCGSI} \approx \mathscr{C}_{START} + \texttt{InvIt} \times (\mathscr{C}_{QR} + \mathscr{C}_{RITZ} + \mathscr{C}_{UPDATE}). \tag{26}$$

## 5 Exploiting the Spectral Information

Once the *near*-invariant subspace linked to the smallest eigenvalues of the linear system is obtained, we can use it for solving any system with the same coefficient matrix, taking advantage of this spectral information to remove the effect of the poor conditioning. An overview of techniques that exploit this idea to improve the convergence of the Conjugate Gradient can be found, for instance, in [6]. Here, we summarize two of these techniques based on the same approach, i.e. building a spectral projector that enables to work in the orthogonal complement of the invariant subspace corresponding to the smallest eigenvalues.

### *5.1 Deflated Starting Guess*

One of the possible methods is to compute a starting guess, by means of an oblique projection of the initial residual ($r^{(0)} = M^{-1}b - M^{-1}Ax^{(0)}$) onto the *near*-invariant subspace associated with the eigenvalues in the range $]0, \mu[$, to get the corresponding eigencomponents in the system solution:

$$\begin{aligned} r^{(1)} &= r^{(0)} - M^{-1}AW\Delta^{-1}W^T Mr^{(0)}, \\ \text{and } x^{(1)} &= x^{(0)} + W\Delta^{-1}W^T Mr^{(0)}, \end{aligned}$$

where $W$ and $\Delta$ are the matrices of the $q$ converged Ritz vectors and values, obtained by the BlockCGSI algorithm on the preconditioned matrix $M^{-1}A$. To compute the remaining part $x^{(2)}$ of the exact solution vector $x^* = x^{(1)} + x^{(2)}$, we can solve $M^{-1}Ax^{(2)} = r^{(1)}$ with the Conjugate Gradient algorithm.

In practice, as $x^{(0)} = 0$ we run the Conjugate Gradient to solve $M^{-1}Ax = M^{-1}b$ starting from the deflated component of the solution

$$x^{(1)} = W\Delta^{-1}W^T b. \tag{27}$$

With this initial starting guess, we expect that the CG will converge to the remaining part of the solution very quickly, since the difficulties caused by the smallest eigenvalues have been swallowed, and the eigenvalues bounds are given by $\mu$ and $\lambda_{\max}$, as explained in [16]. In this way, the Conjugate Gradient should reach linear convergence immediately [17]. For clarity, we will call INIT- CG the algorithm corresponding to CG with a starting guess obtained with deflation.

### *5.2 Spectral Low Rank Update (SLRU) Preconditioner*

Another way of exploiting spectral information from the coefficient matrix is to perform a deflation at each CG iteration, instead of just at the beginning. This approach, proposed in [7], is called Spectral Low Rank Update (SLRU) preconditioning.

The computation of the solution of the preconditioned system $M^{-1}Ax = M^{-1}b$ is obtained by means of the CG algorithm applied to an equivalent system $\widehat{M}Ax = \widehat{M}b$, where the preconditioner $\widehat{M}$ is given by

$$\widehat{M} = M^{-1} + W\Delta^{-1}W^T. \tag{28}$$

In this case $M$ and $\widehat{M}$ are also called the first and second level of preconditioning. The preconditioner $\widehat{M}$ will shift the smallest eigenvalues in the coefficient matrix $M^{-1}A$ close to one (see [7]). In some cases, it can be useful to shift the smallest eigenvalues close to some predetermined value $\lambda$ (with $\lambda = \lambda_{max}$, for instance), in which case the spectral preconditioner should be set to

$$\widehat{M} = M^{-1} + \lambda W\Delta^{-1}W^T.$$

This is not useful in our test problem, because the spectrum is previously clustered near one with the first level of preconditioning $M$. In the following, we will call SLRU- CG to the algorithm corresponding to the preconditioned Conjugate Gradient with SLRU as preconditioner.

## 5.3 Pratical Considerations

We first consider operations count. As in Sect. 4, we assume that $q \ll n$ so that we neglect terms not containing $n$, ($q$ is the dimension of the *near*-invariant basis $W$).

In addition to the sparse matrix-vector product with $A$ at each iteration, the CG iteration add merely two dot-products, DOT, and three vector updates, AXPY (see Sect. 4).

The algorithms INIT- CG and SLRU- CG perform both an oblique projection of the initial residual onto the *near*-invariant basis $W$ of size $q$, involving the pre-computation of $W\Delta^{-1}W^Tb$, where $\Delta = W^TAW$ is the Ritz matrix computed on step vi of Algorithm 1. The cost of its inversion is not significant because we consider it as a diagonal matrix. We note that $\Delta^{-1}$ is stored jointly with the basis $W$. In the scheme SLRU-CG, the multiplication with the preconditioner $\widehat{M}$ also implies an oblique projection $W\Delta^{-1}W^Tr^{(k)}$ at each iteration (see Eq. (28)). This projection can be done using common level 2 BLAS operations [18] with a total cost roughly equal to

$$\mathscr{C}_{\text{Proj}} \approx 4\,(p+1)\,n. \tag{29}$$

In Table 2, we indicate the total cost in floating-point operations for each algorithm, with an initial cost, and a cost per iteration. One of the major differences between INIT-CG and SLRU- CG is that SLRU- CG uses the projection operator $W\Delta^{-1}W^T$ at each iteration, whereas INIT- CG does exploit this only at the beginning (for computing a starting vector $x^{(1)}$, see formula (27)). Anyway, this also contributes to better numerical stability in the convergence process of SLRU- CG.

**Table 2** Cost in floating-point operations for different methods

| | The cost in floating-point operations | |
|---|---|---|
| | At the beginning | At each iteration |
| CG | $\mathscr{C}_A + \mathscr{C}_M + - + -$ | $\mathscr{C}_A + \mathscr{C}_M + 3\mathscr{C}_{AXPY} + 2\mathscr{C}_{DOT} + -$ |
| INIT- CG | $\mathscr{C}_A + \mathscr{C}_M + \mathscr{C}_{\text{Proj}} + -$ | $\mathscr{C}_A + \mathscr{C}_M + 3\mathscr{C}_{AXPY} + 2\mathscr{C}_{DOT} + -$ |
| SLRU- CG | $\mathscr{C}_A + \mathscr{C}_M + \mathscr{C}_{\text{Proj}} + -$ | $\mathscr{C}_A + \mathscr{C}_M + 3\mathscr{C}_{AXPY} + 2\mathscr{C}_{DOT} + \mathscr{C}_{\text{Proj}}$ |

Let us briefly examine the memory requirements of these two acceleration techniques. In comparison with the CG algorithm, the INIT- CG and SLRU- CG schemes require about the same amount of extra storage, of order $n(q+1)$, to store $W$, the basis of the *near*-invariant subspace (Ritz vectors), and $diag(\Delta)$ the corresponding Ritz Vectors.

## 6 Numerical Experiments

In this section, we report on some numerical experiments concerning the computation of the spectral information associated with the smallest eigenvalues of a preconditioned matrix $M^{-1}A$. We also include some experiments concerning the use of the pre-computed spectral information to improve the consecutive solution of linear systems with the same coefficient matrix as mentioned in Sect. 5. Theses aspects are illustrated on a test matrix coming from the 2D heterogeneous diffusion equation in a L shape region, discretized by finite elements, with size $n = 7969$. Another application of the BlockCGSI algorithm to a larger problem can be found in [19].

We also precondition the resulting linear system with Jacobi scaling or classical Incomplete Cholesky (see Table 3).

We divide the experiments in two parts, the first one concerns the monitoring of the BlockCGSI algorithm discussed in Sect. 3 and the second concerns the improvement of the convergence of the CG algorithm with the pre-computed spectral information.

**Table 3** Properties of the test matrix $M^{-1}A$ with two different preconditioners

| Preconditioner | $\lambda_{\min}$ | $\lambda_{\max}$ | # eigs. below $\mu =$ | | | $nnz$ | $nnz$ | $n$ |
|---|---|---|---|---|---|---|---|---|
| | | | $1e-3$ | $5e-3$ | $1e-2$ | $R$ | $A$ | |
| Jacobi | $3.07e-09$ | 2.08 | 2 | 9 | 18 | 7969 | 55131 | 7969 |
| IC(0) | $1.66e-08$ | 1.55 | 2 | 2 | 3 | 31550 | | |

### 6.1 Monitoring the Subspace Iteration

In Fig. 1, we show the convergence behavior of the inner invariance measure

$$\frac{||M^{-1}Az_1^{[i]} - \delta_1^{[i]} z_1^{[i]}||_M}{||z_1^{[i]}||_M}, \tag{30}$$

which is directly related with the values of $\omega_1$ and $\phi_1$ as evidenced in (13). The two plots in Fig. 1 illustrate the behavior of these three values in the blockCG run at the first inverse iteration ($k = 1$). The first plot corresponds to the case of non filtered starting vectors, and the second one to the case of starting vectors filtered with a level $\xi = 1e-2$ and a cut-off value $\mu_f = 5e-3$.

We can observe the effect of the Chebyshev filtering of the starting vectors, which helps to make the value of $\phi_1$ much smaller than what it can be with a randomly generated initial set of vectors. The direct consequence is that $\omega_1$ then becomes a good measure of the inner invariance measure (30), even at the very beginning of the algorithm. Additionally, the filtering of the starting vectors changes the convergence behavior of the blockCG, because the filtered right-hand sides have more favorable spectral properties. It also enables to decrease substantially the asymptotic value of $\phi_1$ in the first inverse iteration, allowing a larger range of values for the choice of the threshold $\varepsilon_{\text{inner}}$ in the blockCG, which is a desirable feature for the algorithm as discussed in Sect. 3.3.

In Fig. 2 we plot the evolution of the invariance measure (6) of the Ritz vector $v_1$ as a function of the number of inverse iterations. We compare two different inner stopping criteria with the exact inverse iteration. The lozenge curve corresponds to the stopping criteria (19), the circles curve corresponds to stop the inner iteration when $\omega_1 < \phi_1$ and the circles curve corresponds to the exact case (performed by the Cholesky factorization of the coefficient matrix). We can observe in Fig. 2 that the proposed inner stopping

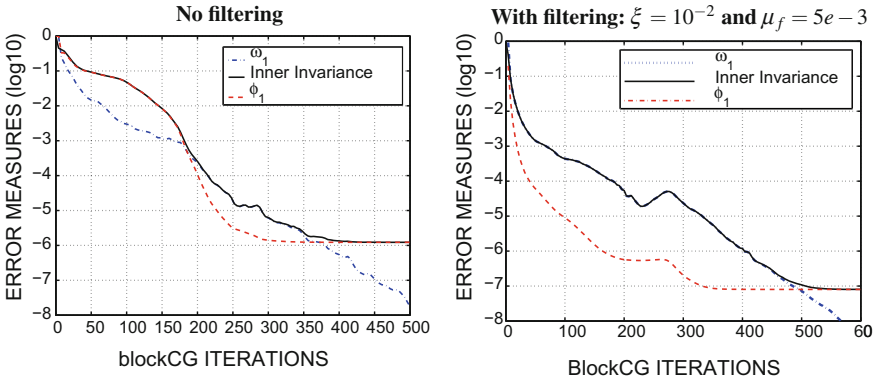


**Fig. 1** Correlation between the inner invariance measure (30), $\omega_1$ and $\phi_1$, in the blockCG with block size 4, and at the first subspace iteration. The test matrix is preconditioned with Jacobi scaling

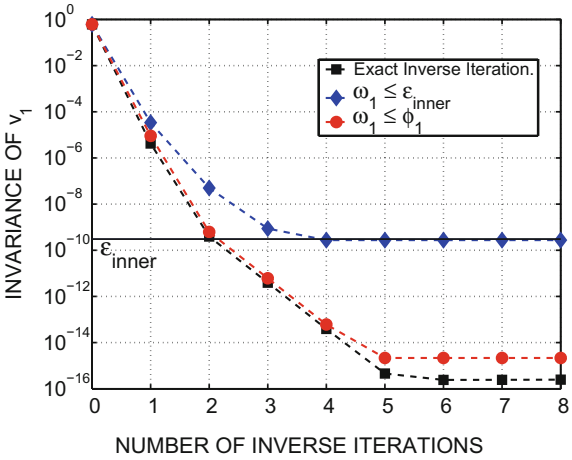


**Fig. 2** Invariance of the Ritz vector $v_1$ over the number of outer iterations

criteria force the invariance to decrease until the targeted tolerance is reached, which occur at the 4th inverse iteration. Stopping the blockCG when $\omega_1 < \phi_1$ enables to reach the same behavior as in the exact case, which is in agreement with the analysis developed in Sect. 3.2.

In Table 4, we present both the total number of inner and outer iterations in the BlockCGSI algorithm (see Algorithm 1) to compute a *near*-invariant subspace associated with all eigenvalues in the range $]0, \mu[$. The requested relative invariance in these approximated eigenvectors was set to $\varepsilon_{\text{outer}} = 10^{-1}$, with respect to the convergence criterion for the outer loop given by Eq. (18), and the stopping criterion for the blockCG set accordingly as in (19). The total number of inverse iterations is indicated by `InvIt`, and the value of `bCGIt` denotes the sum of all the iterations performed by the blockCG solver in the given BlockCGSI run. The Chebyshev iterations count, `ChebIt`, incorporates all the Chebyshev iterations spent when filtering the starting vectors, as well as when incorporating new vectors during update of the computational window (see Algorithm 2). Finally, we also include the total number of floating point operations performed by the BlockCGSI algorithm ($\mathscr{C}_{BCGSI}$), in millions (`Mflops`), computed as in (26). We varied the filtering level from $\xi = 1e-6$ to $\xi = 1e-16$, including the case of no filtering. The block size was chosen to illustrate these cases, e.g. when it is below, equal or greater than the targeted number of eigenvectors ($q$). The two cut-off values of the filtering step $\mu_f$ correspond to the cases when it is greater or equal to the principal cut-off value $\mu$ in Algorithm 1.

The results in Table 4 show that the algorithm manages to compute the targeted spectral information independently of the choice of the block size $s$. Of course, it is optimal when $s$ is correlated to the actual number of eigenvalues ($q$) in the range $]0, \mu[$. In this case, all the iterations counts are minimized as well as the total number of operations. With larger block sizes $s$, the algorithm benefits from the *"guard vectors"* effect (see Sect. 2), and the number of inverse iterations are reduced. A greater block size also improves the convergence of the block Conjugate Gradient. For these reasons, the

**Table 4** Iteration and operation counts as a function of the filtering level $\xi$

| Filter level $\xi$ | $\mu = 1.0e-3$ (2 eigenvalues) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $s = 2, \mu_f = 1.0e-2$ | | | | $s = 5, \mu_f = 1.0e-2$ | | | |
| | InvIt | bCGIt | ChebIt | Mflops | InvIt | bCGIt | ChebIt | Mflops |
| – | 6 | 240 | – | 138 | 2 | 177 | – | 469 |
| $1e-6$ | 2 | 170 | 105 | 131 | 2 | 98 | 105 | 347 |
| $1e-8$ | 2 | 139 | 138 | 125 | 2 | 67 | 138 | 294 |
| $1e-10$ | 2 | 109 | 171 | 119 | 2 | 50 | 171 | 278 |
| $1e-12$ | 2 | 80 | 205 | 114 | 2 | 26 | 205 | 245 |
| $1e-14$ | 2 | 53 | 238 | 110 | 2 | 10 | 238 | 230 |
| $1e-16$ | 2 | 35 | 275 | 120 | 2 | 9 | 275 | 256 |
| Filter level $\xi$ | $\mu = 5.0e-2$ (9 eigenvalues) | | | | | | | |
| | $s = 5, \mu_f = 1.0e-2$ | | | | $s = 9, \mu_f = 1.0e-2$ | | | |
| | InvIt | bCGIt | ChebIt | Mflops | InvIt | bCGIt | ChebIt | Mflops |
| – | 29 | 432 | – | 1347 | 14 | 247 | – | 2033 |
| $1e-6$ | 10 | 119 | 630 | 1333 | 2 | 65 | 105 | 661 |
| $1e-8$ | 4 | 69 | 414 | 777 | 2 | 41 | 138 | 530 |
| $1e-10$ | 3 | 51 | 342 | 572 | 2 | 23 | 171 | 444 |
| $1e-12$ | 3 | 27 | 410 | 595 | 2 | 6 | 205 | 367 |
| $1e-14$ | 3 | 11 | 476 | 636 | 2 | 3 | 238 | 387 |
| $1e-16$ | 3 | 10 | 542 | 715 | 1 | 3 | 271 | 436 |
| Matrix preconditioned with Jacobi scaling | | | | | | | | |
| Filter level $\xi$ | $\mu = 1.0e-2$ (3 eigenvalues) | | | | | | | |
| | $s = 3, \mu_f = 1.0e-1$ | | | | $s = 5, \mu_f = 1.0e-1$ | | | |
| | InvIt | bCGIt | ChebIt | Mflops | InvIt | bCGIt | ChebIt | Mflops |
| – | 6 | 96 | – | 148 | 4 | 73 | – | 243 |
| $1e-6$ | 6 | 75 | 28 | 143 | 2 | 44 | 28 | 180 |
| $1e-8$ | 6 | 63 | 37 | 134 | 2 | 35 | 37 | 165 |
| $1e-10$ | 6 | 53 | 46 | 127 | 2 | 27 | 46 | 153 |
| $1e-12$ | 3 | 35 | 55 | 99 | 2 | 19 | 55 | 140 |
| $1e-14$ | 3 | 27 | 64 | 96 | 2 | 12 | 64 | 128 |
| $1e-16$ | 2 | 19 | 75 | 88 | 1 | 10 | 73 | 130 |
| Filter level $\xi$ | $\mu = 3.0e-2$ (9 eigenvalues) | | | | | | | |
| | $s = 5, \mu_f = \mu$ | | | | $s = 9, \mu_f = \mu$ | | | |
| | InvIt | bCGIt | ChebIt | Mflops | InvIt | bCGIt | ChebIt | Mflops |
| – | 30 | 191 | – | 907 | 17 | 118 | – | 1310 |
| $1e-6$ | 4 | 41 | 104 | 358 | 2 | 25 | 52 | 351 |
| $1e-8$ | 3 | 20 | 138 | 352 | 2 | 11 | 69 | 274 |
| $1e-10$ | 3 | 12 | 170 | 390 | 1 | 3 | 85 | 235 |
| $1e-12$ | 3 | 6 | 204 | 434 | 1 | 3 | 102 | 275 |
| $1e-14$ | 2 | 4 | 236 | 450 | 1 | 1 | 118 | 295 |
| $1e-16$ | 2 | 4 | 270 | 556 | 1 | 1 | 135 | 335 |
| Matrix preconditioned with $IC(0)$ | | | | | | | | |

increase of $s$ does not necessarily imply an increase of the total amount of work. When the block size is smaller than $q$, the *"sliding window"* feature enables to obtain at any rate all the targeted vectors. Our experiments also show that the final number of converged vectors can exceed the actual number of eigenvalues in the range $]0, \mu[$ when the block size is not equal to this number.

As we can observe in Table 4, the filtering of the new vectors with Chebyshev polynomials can improve quite a lot the efficiency of the BlockCGSI algorithm. As the filtering level $\xi$ decreases, the number of inverse iterations is reduced because the resulting filtered vectors get closer to a *near*-invariant subspace, and the stagnation level of $\phi_j$ becomes much lower (see also Fig. 1). This also gives room for larger decrease of the inner invariance (30) at each inverse iteration. When the block size $s$ is equal to the number $q$ of eigenvalues in $]0, \mu[$, the convergence can be reached with a minimum number of inverse iterations (`InvIt` $= 1$ for instance). The number of blockCG iterations is also reduced because of the better spectral properties of the right-hand sides. Obviously, decreasing the filtering level $\xi$ also increases the number of Chebyshev iterations in the filtering process. In that respect, there is a compromise to reach in terms of total computational cost. The optimal value of $\xi$, that minimizes this computational work (`Mflops`), also depends on the other filtering parameter $\mu_f$, and on the number of targeted eigenvalues $q$. We have observed that it is better in general to take the value of the initial filtering parameter $\mu_f$ not too large with respect to the actual bound $\mu$ on the range of targeted eigenvalues, and also that when the number of targeted eigenvalues is small, a good filtering level $\xi$ is indicated.

## 6.2 Improving the CG Convergence

Based on the pre-computed spectral information, we can improve the convergence of the CG algorithm. To illustrate this, we have solved the two tests systems with both INIT- CG or SLRU- CG. In Fig. 3, we plot the backward error

$$\rho^{(i)} = \frac{||M^{-1}r^{(i)}||_M}{||M^{-1}r^{(0)}||_M} = \frac{||R^{-T}r^{(i)}||_2}{||R^{-T}r^{(0)}||_2}, \tag{31}$$

normally used in the preconditioned Conjugate Gradient, where $r^{(i)} = b - Ax^{(i)}$.

The results show the effectiveness of the use of the spectral information to reduce the total number of iterations. When the system is preconditioned with Jacobi scaling (see plots on the top of Fig. 3), the initial condition number, of order $10^8$, can even be reduced to the order of $10^3 \approx \lambda_{max}/\mu$ with the use of the first two vectors associated with the two smallest eigenvalues only. With the use of a larger number of Ritz vectors, the reduced condition number is maintained to about the same level, and just little improvements can be expected in the convergence rate of the CG algorithm. With the Incomplete Cholesky preconditioner, the number of critical eigenvalues seems also to be 2 (see plots on the bottom of Fig. 3).

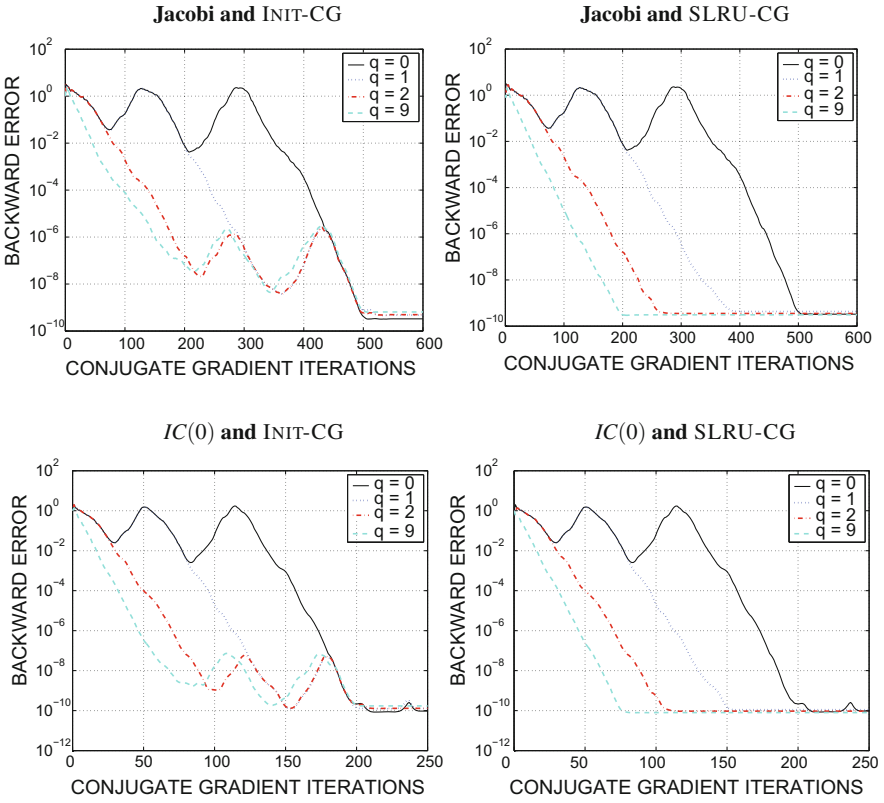


**Fig. 3** Convergence behavior of INIT- CG and SLRU- CG with different sizes $q$ of the pre-computed *near*-invariant subspace. The system is initially preconditioned either with Jacobi scaling (top) or with the standard Incomplete Cholesky (bottom)

Regarding the results in Fig. 3, the SLRU preconditioner is numerically more stable than the deflation technique. The SLRU- CG converges linearly while the INIT- CG looses the linear rate of convergence when reaching small residual values (say $10^{-6}$ with Jacobi preconditioner and $10^{-8}$ with $IC(0)$). To maintain this linear rate all through the iterations, the spectral information needs to be more accurate, as we can observe in Fig. 4. Indeed, with larger values for the number of correct digits $t$ (see formula (18)) the irregularities in the rate of convergence of INIT- CG are smoothed gradually. However the cost for computing a much more accurate *near*-invariant basis can be rather large, and it can be preferable to simply use the SLRU- CG algorithm if solutions with high precision are needed.

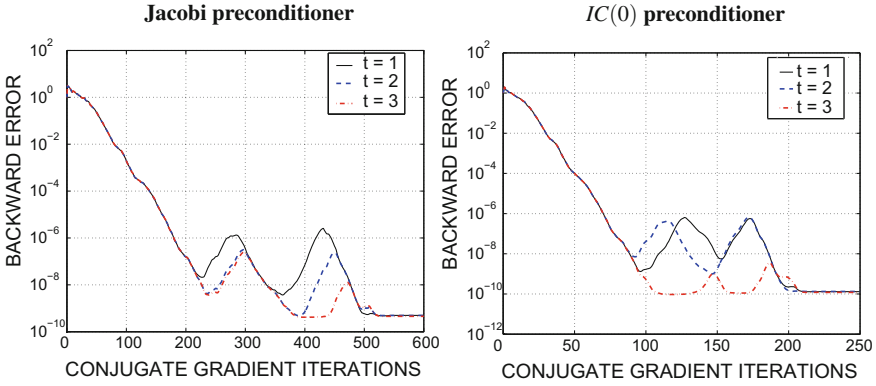


**Fig. 4** Effect of the accuracy of the spectral information on the convergence behavior of INIT- CG

## *6.3 Cost-Benefit*

We have proposed a technique to improve the consecutive solutions of several systems with the same coefficient matrix but with different right-hand sides. This technique is based on a two phase approach: we first perform a partial spectral decomposition of the coefficient matrix $M^{-1}A$ with the help of BlockCGSI algorithm, and we use this information afterward to accelerate the CG through the deflation of the starting guess or with a second level of preconditioning. We illustrate how the gains obtained at each solve can reduce substantially the total computational cost in the long run.

We begin by presenting the costs in floating-point operations involved in each CG run (see Sect. 5.3). In Fig. 5, we plot the history of the backward error versus the number of floating-point operations. In the case of SLRU- CG algorithm, we observe that higher dimension $q$ of the *near*-invariant subspace does not always bring an improvement in the convergence. The oblique projection (28) performed at each iteration is responsible for the growth of the computational work when the dimension $q$ gets larger. As we can see on the right of Fig. 5, when $q$ varies from 3 to 20 the rate of convergence decreases, and no gains are obtained despite the effective reduction of the number of CG iterations (see Fig. 3). As opposed to that, we can observe in the case of INIT- CG algorithm (left of Fig. 5) that we always get improvements with larger values of $q$.

As we have seen in Sect. 4, the pre-computation of the *near*-invariant subspace $W$ has a cost, that we denote by $\mathscr{C}_{BCGSI}$, depends on the dimension of this subspace and on some working parameters in the BlockCGSI algorithm. To be effective, the gains obtained in the acceleration of the convergence of the given iterative solvers must compensate, in some way, the extra cost for this spectral pre-computation. In Table 5, we present the computational costs $\mathscr{C}_{BCGSI}$ (in millions of operations, `Mflops`) for three different cases that correspond to different choices for the cut-off value $\mu$. For each one, we have computed all the $q$ Ritz vectors corresponding to the $q$ eigenvalues in the range $]0, \mu[$. The spectral information is computed with two correct digits ($t = 2$) in the case of Jacobi preconditioner, and with one digit ($t = 1$) in the case of $IC(0)$.

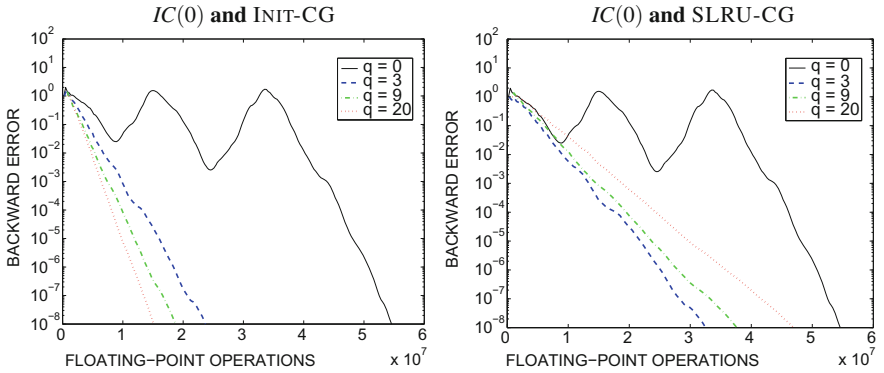


**Fig. 5** History of the backward error as a function of the computational cost, for different sizes $q$ of the *near*-invariant subspace. The case $q = 0$ corresponds to the CG algorithm, for comparison

To illustrate the cost-benefits, we stop the CG iterations when the relative residual norm $\rho^{(i)}$ (see Eq. (31)) is below $10^{-8}$. When preconditioned with Jacobi scaling, the Conjugate Gradient performs 478 iterations with a cost of 95 `Mflops`, and when preconditioned with $IC(0)$, 187 iterations with a cost of 55 `Mflops`. In this table, we indicate the number of CG iterations (`Nit`), the number of floating-point operations `Mflops` and the number of amortization right-hand sides (`Amor. rhs.`), i.e., the number of right-hand sides that have to be considered in consecutive solves before the extra cost $\mathscr{C}_{BCGSI}$ is compensated. The number of amortization right-hand sides is given by

$$\texttt{Amor. rhs.} = \left\lfloor \frac{\mathscr{C}_{BCGSI}}{\mathscr{C}_{CG} - \mathscr{C}_{aCG}} \right\rfloor + 1,$$

where $\mathscr{C}_{CG}$ is the cost of CG algorithm without acceleration, and $\mathscr{C}_{aCG}$ is cost of the accelerated CG, either INIT- CG or SLRU- CG algorithm. These informations are given for each cut-off value $\mu$. For instance, in Table 5, with Jacobi scaling and $\mu = 4.0e-3$, 211 `Mflops` are needed for the spectral pre-computation, out of which the INIT- CG algorithm achieves convergence in 190 iterations and 38 `Mflops`, i.e. a reduction of 60% compared to the run which does not use this spectral information. Consequently, the 211 extra `Mflops` are paid back after four consecutive accelerated solves, compared to four runs of the non-accelerated CG.

Table 5 shows that Init-CG and SLRU-CG converge, in general, in the same number of iterations if the spectral information enough accurate. The main difference is that SLRU demands more `Mflops` as the size of the *near*-invariant subspace ($q$) increases. For this reason, the number of amortization vectors (`Amor. rhs`) is greater with the SLRU preconditioner. In the case of the chosen stopping threshold ($10^{-8}$), the INIT- CG approach seems to be preferable.

In summary, the numerical results demonstrate that, when an accuracy of order $10^{-8}$ is required to solve linear systems in sequence with the same matrix but with changing right-hand sides, the cost of pre-computation of a *near*-invariant subspace with Block-

**Table 5** Cost-benefits of CG accelerated with the spectral information

| | Spectral fact. | | | Deflated $x^{(0)}$ | | | SLRU prec. | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\mu$ | $q$ | $\mathscr{C}_{BCGSI}$ | CG | | Amor. rhs. | CG | | Amor. rhs. |
| | | | Mflops | Nit | Mflops | | Nit | Mflops | |
| Jacobi precond. | – | 0 | – | 478 | 95 | – | 478 | 95 | – |
| | $1.0e-3$ | 2 | 184 | 227 | 45 | 4 | 227 | 63 | 6 |
| | $4.0e-3$ | 5 | 211 | 190 | 38 | 4 | 187 | 71 | 9 |
| | $5.5e-3$ | 9 | 427 | 176 | 38 | 8 | 166 | 84 | 39 |
| $IC(0)$ precond. | – | 0 | – | 187 | 55 | – | 187 | 55 | – |
| | $1.0e-2$ | 3 | 88 | 89 | 26 | 4 | 80 | 33 | 4 |
| | $2.0e-2$ | 5 | 145 | 79 | 23 | 5 | 74 | 35 | 8 |
| | $3.0e-2$ | 9 | 235 | 75 | 22 | 8 | 62 | 37 | 14 |

CGSI algorithm is largely compensated by the gains obtained in the long run. This is still more effective if a first level of preconditioning is applied to cluster better the spectrum of the iteration matrix.

## 7 Concluding Remarks

The BlockCGSI algorithm computes a *near*-invariant subspace, associated with the smallest eigenvalues in $M^{-1}A$, which combines the subspace inverse iteration and a stabilized version of the block Conjugate Gradient algorithm. The main focus in this work was the control of the accuracy when solving the system with multiple right-hand sides at each inverse iteration, and the good agreement of the stopping criterion used in the blockCG iteration with the measure of convergence of the inverse iteration itself. Similarly to other inexact inverse iteration analysis (for instance [9, 11, 20]), we analyze the inner-outer iteration in the blockCGSI algorithm, and we propose to measure the residuals of the system through a Rigal–Gaches type of backward error. This measure enables the control of the absolute eigenvalue error of the inverse iteration, at the same time that the system is solved. The control is even more effective at the first inverse iteration if the starting vectors are previously filtered with Chebyshev polynomials. We also derive an expression, linked to the proposed residual measure, that indicates when the inner iteration must be stopped if we want to recover the same type of convergence as in the exact inverse iteration. Based on the asymptotic behavior of this expression, we suggest how to avoid unnecessary extra computational work in the blockCG inner iteration.

We also investigated some particular techniques, like the Chebyshev filtering of the random generated vectors, and a form of dynamic adjustment of the dimension of the

current subspace at each inverse iteration. The experiments indicated that Chebyshev filtering is useful to reduce the total number of inverse and blockCG iterations, and consecutively to reduce the total amount of work. The *"sliding window"* technique is helpful to make the algorithm flexible and robust. Some of the good features of the BlockCGSI algorithm (see Algorithm 1) also yield in the easy control of the memory requirements as well as the a priori control of the accuracy.

Once we have computed the spectral information associated with the smallest eigenvalues, we experiment different strategies for improving the consecutive solution of linear systems with the Conjugate Gradient algorithm. In that respect, we have focused on two closely related approaches: (1) deflating the eigencomponents associated with the smallest eigenvalues with an appropriate starting guess, or (2) using the SLRU preconditioner that shifts these eigenvalues away from zero. The latter appeared to be numerically more stable, achieving linear convergence, even when the pre-computed spectral information was obtained with low accuracy. Nevertheless, the first approach is less expansive in terms of computational cost, and is a preferable option if the multiples systems are solved with a not very small stopping criterion.

The experiments show that, if the spectrum is previously clustered, with the help for instance of a first level of preconditioning, the strategy is very efficient in the reduction of the total cost of solving consecutive linear systems with changing right-hand sides. The extra work needed to compute the spectral information is payed back after a small number of consecutive solutions.

The two-phase strategy is also very effective in other applications. In previous work [19], were it was used to accelerate the simulation of the flow around an airplane wing, we have verified that the reduction of the total amount of computational costs can reach 70%.

## References

1. Parlett, B.N.: The Symmetric Eigenvalue Problem. SIAM, Philadelphia (1998)
2. O'Leary, D.P.: The block conjugate gradient algorithm and related methods. Linear Algebra Appl. **29**, 293–322 (1980)
3. Arioli, M., Duff, I., Ruiz, D., Sadkane, M.: Block Lanczos techniques for accelerating the block cimmino method. SIAM J. Sci. Stat. Comput. **16**(6), 1478–1511 (1995)
4. Balsa, C., Daydé, M., Palma, J.M.L.M., Ruiz, D.: Monitoring the block conjugate gradient convergence within the inexact inverse subspace iteration. In: Wyrzykowski, R., Dongarra, J., Meyer, N., Wasniewski, J. (eds.) Parallel Processing and Applied Mathematics. Lectures Notes in Computer Science - PPAM05, LNCS 3911, pp. 494–504. Springer, Berlin (2006)
5. Arioli, M., Ruiz, D.: Block conjugate gradient with subspace iteration for solving linear systems. In: Margenov, S., Vassilevski, P. (eds.) Iterative Methods in Linear Algebra, Second IMACS Symposium on Iterative Metohds in Linear Algebra, pp. 64–79. Blagoevgrad, Bulgaria (1995)
6. Giraud, L., Ruiz, D., Touhami, A.: A comparative study of iterative solvers exploiting spectral information for SPD systems. SIAM J. Sci. Comput. **27**(5), 1064–8275 (2006)
7. Carpentieri, B., Duff, I., Giraud, L.: A class of spectral two-level preconditioners. SIAM J. Sci. Comput. **25**(2), 749–765 (2003)
8. Lai, Y.L., Lin, K.Y., Lin, W.W.: An inexact inverse iteration for large sparse eigenvalue problems. Numer. Linear Algebra Appl. **4**(5), 425–437 (1997)

9. Smit, P., Paardekooper, M.H.C.: The effects of inexact solvers in algorithms for symmetric eigenvalue problems. Linear Algebra Appl. **287**, 337–357 (1999)
10. Golub, G.H., Ye, Q.: Inexact inverse iteration for generalized eigenvalue problems. BIT **40**, 671–684 (2000)
11. Simoncini, V., Eldén, L.: Inexact Rayleigh quotient-type methods for eigenvalue computations. BIT **42**, 159–182 (2002)
12. Berns-Mueller, J., Graham, I. G., Spence, A.: Inexact inverse iteration for symmetric matrices. Linear Algebra and its Appl. **416**(2–3), 389–413 (2006)
13. Notay, Y.: Combination of Jacobi-Davidson and conjugate gradients for the partial symmetric eigenproblem. Numer. Linear Algebra Appl. **9**, 21–44 (2002)
14. Dongarra, J.J., Ducroz, J., Hammarling, S., Duff, I.: A set of level 3 basic linear algebra subprograms. ACM Trans. Math. Softw. **16**(1), 1–28 (1990)
15. Rigal, J.L., Gaches, J.: On the compatibility of a given solution with the data of a linear system. J. ACM **14**(3), 543–548 (1967)
16. Hageman, L.A., Young, D.M.: Applied Iterative Methods. Academic Press, New York (1981)
17. Van der Sluis, A., van der Vorst, H.A.: The rate of convergence of Conjugate Gradients. Numer. Math. **48**(5), 543–560 (1986)
18. Dongarra, J.J., Ducroz, J., Hammarling, S., Hanson, R.: An extended set of Fortran basic linear algebra subprograms. ACM Trans. Math. Softw. **14**, 1–17 (1988)
19. Balsa, C., Braza, M., Daydé, M., Palma, J.M.L.M., Ruiz, D.: Improving the numerical simulation of an airflow problem with the BlockCGSI algorithm. In: Daydé, M., Palma, José M.L.M., Coutinho, Álvaro L.G.A., Pacitti, Esther, Correia Lopes, J. (eds.) High Performance Computing for Computational Science - VECPAR 2006. LNCS 4395, pp. 281–291. Springer, Berlin (2007)
20. Notay, Y.: Convergence analysis of inexact Rayleigh quotient iteration. SIAM J. Matrix Anal. Appl. **24**(3), 627–644 (2003)