



Universidade do Porto
Faculdade de Engenharia

FEUP



Inexact Subspace Iteration to Accelerate the Solution of Linear Systems with Multiple Right-Hand Sides

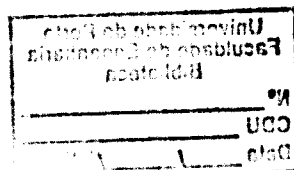
Carlos Jorge da Rocha Balsa

(MsC in Computational Methods in Sciences and Engineering)

Thesis submitted in partial fulfillment of the requirements of joint Doctoral Degrees
in Engineering Sciences by the University of Porto, and Doctorat de l'Institut National
Polytechnique de Toulouse, Mention Informatique et Télécommunications.

Inexact Subspace Iteration to Accelerate the Solution of Linear Systems with Multiple Right-Hand Sides

Carlos Jorge da Rocha Balsa



004(043) / BALC / 1114

Universidade do Porto	
Faculdade de Engenharia	
Biblioteca	
Nº	84425
CDU	
Data	12 / 6 / 2006

Abstract

We propose a two-phase acceleration technique for the solution of Symmetric and Positive Definite (SPD) 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 (CG) 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, we combine the block Conjugate Gradient algorithm with the Subspace Iteration to build a purely iterative algorithm, that we call Block-CGSI. We analyze the convergence of the blockCG algorithm and exploit the possibility of reducing the total amount of computational work by controlling in a same appropriate manner the accuracy during the solution of linear systems at each subspace iteration.

We also improve the global convergence of this algorithm by using Chebyshev polynomials as a spectral filtering tool when building the starting vectors. The concept of “*sliding window*” was also introduced as an algorithmic feature that enables the computation of a *near*-invariant subspace of any dimension.

The spectral information computed by the BlockCGSI algorithm is then used to remove the effect of the smallest eigenvalues in two different ways: either by building a Spectral Low Rank Update (SLRU) preconditioner that basically adds the value 1 to the approximated eigenvalues, or by performing a deflation of the initial residual in order to remove part of the solution corresponding to the smallest eigenvalues. Both techniques yield important reductions of the total number of iterations and computational work in each subsequent runs of the Conjugate Gradient algorithm.

We report on experiments on a 2D diffusion equation as well as on two applications coming from Computational Fluid Dynamics (CFD). The analysis of costs and benefits, in terms of floating point operations, helps to validate the strategy as a good way to speed up the solution of symmetric and positive definite linear systems.

Key words: Numerical simulations, sparse matrices, block iterative methods, preconditioning, Conjugate Gradient, block Conjugate Gradient, inexact subspace iteration, partial spectral factorization, eigenvalues and eigenvector approximation, Chebyshev polynomials, convergence analysis, stopping criterion.

Resumo

Propomos uma técnica em duas fases para acelerar a resolução de sistemas de equações lineares, simétricos e definidos positivos, com vários termos independentes. Na primeira fase uma parte da informação espectral, associada ao mau condicionamento da matriz do sistema, é extraída e, numa segunda fase, esta informação é utilizada para melhorar a convergência do método do Gradiente Conjugado (CG).

Esta abordagem tem interesse particular na resolução de problemas de grande dimensão, como por exemplo na simulação numérica de equações diferenciais dependentes do tempo, onde se resolvem sequencialmente vários sistemas de equações algébricas com matrizes idênticas ou com aproximadamente as mesmas propriedades espectrais, mas com diferentes termos independentes.

Para a extração da informação espectral, propomos uma combinação do algoritmo do Gradiente Conjugado por blocos (blockCG) com a iteração de sub-espço. O resultado consiste num algoritmo puramente iterativo, chamado BlockCGSI. Analisamos a convergência do blockCG a fim de reduzir o número total de cálculos através do controlo apropriado da resolução dos sistemas em cada iteração de sub-espço.

Melhoramos também a convergência global do algoritmo BlockCGSI através de filtros espectrais, baseados nos polinómios de Tchebycheff, aplicados sobre os vectores de partida. Introduzimos também no algoritmo o conceito de “*janela deslizante*” para permitir o cálculo de sub-espços invariantes cuja dimensão não é antecipadamente conhecida.

A informação espectral, calculada com o algoritmo BlockCGSI, é então utilizada para retirar o efeito provocado pelos valores próprios mais pequenos de duas maneiras diferentes: através da construção de um segundo nível de pré-condicionamento chamado *Spectral Low Rank Update* (SLRU) que adiciona o valor 1 aos valores próprios mais pequenos pré-calculados; ou através da projecção do resíduo inicial para retirar a parte da solução correspondente aos valores próprios mais pequenos. As duas técnicas permitem reduções importantes sobre o número total de operações efectuadas pelo CG.

O conjunto é validado através de um problema de difusão heterogénea bi-dimensional, e em duas aplicações provenientes da Mecânica dos Fluidos Computacional. A análise de custos/benefícios, em termos de número total de operações, valida a estratégia, mostrando

que a abordagem proposta permite acelerar eficientemente a resolução de sistemas lineares simétricos e definidos positivos.

Palavras chave: Simulação numérica, matrizes esparsas, métodos iterativos por blocos, précondicionamento, Gradiente Conjugado e Gradiente Conjugado par blocos, iteração de sub-espço inexacta, factorização espectral parcial, valores próprios e vectores próprios aproximados, polinómios de Tchebycheff, análise de convergência, critério de paragem.

Résumé

Nous proposons une technique en deux-phases pour accélérer la solution de systèmes d'équations linéaires, Symétriques et Positifs Définis (SPD), avec plusieurs seconds membres. Dans la première phase, une certaine partie de l'information spectrale, associée au mauvais conditionnement de la matrice du système, est extraite et, dans une seconde phase, cette information est utilisée pour améliorer la convergence de la méthode du Gradient Conjugué (CG).

Cette approche présente un intérêt pour la résolution de problèmes de grande taille, come par exemple dans la simulation numérique d'équations différentielles dépendantes du temps, où normalement on doit résoudre séquentiellement plusieurs systèmes linéaires dans lesquels la matrice reste la même (où bien avec des matrices qui préservent à peu près les mêmes propriétés spectrales) mais avec différents second membres.

Pour extraire l'information spectrale, nous proposons une combinaison de l'algorithme du Gradient Conjugué par blocs (blockCG) avec l'itération de sous-espace. Le résultat consiste en un algorithme purement itératif, appelé BlockCGSI. Nous analysons la convergence du blockCG afin de permettre la réduction du nombre total des calculs en contrôlant de manière appropriée la résolution des systèmes à chaque itération de sous-espace.

Nous améliorons aussi la convergence globale de l'algorithme par l'intermédiaire de filtres spectraux, basés sur les polynômes de Tchebycheff, qui sont appliqués sur les vecteurs de départ. Le concept de "*fenêtre glissante*" est aussi introduit dans l'algorithme, pour permettre de calculer des sous-espaces invariants avec une dimension non connue à l'avance.

L'information spectrale, calculée avec l'algorithme BlockCGSI, est alors utilisée de deux manières différentes pour retirer l'effet provoqué par les petites valeurs propres : soit en construisant un second niveau de préconditionnement appelé *Spectral Low Rank Update* (SLRU) qui ajoute la valeur 1 aux valeurs propres pré-calculées, soit en effectuant une projection du résidu de départ pour retirer la partie de la solution correspondante aux petites valeurs propres. Les deux techniques permettent des réductions importantes sur le nombre total d'opérations effectuées par le CG.

L'ensemble est validé sur des tests effectués en particulier sur un problème de diffusion hétérogène bi-dimensionnel, ainsi que dans le cadre de deux autres applications provenant

de la Mécanique des Fluides Numérique. L'analyse des coûts et des gains, en terme de nombre total d'opérations, valide la stratégie en montrant que l'approche proposée permet d'accélérer efficacement la résolution de systèmes d'équations linéaires symétriques et positifs définis.

Mots clés : Simulation numérique, matrices creuses, méthodes itératives par blocs, préconditionnement, Gradient Conjugué et Gradient Conjugué par blocs, itération de sous-espace inexacte, factorisation spectrale partielle, valeurs propres et vecteurs propres approchés, polynômes de Tchebycheff, analyse de la convergence, critères d'arrêt.

Acknowledgements

Agradeço a todas as pessoas que me ajudaram, directa ou indirectamente, ao longo destes três últimos anos, na realização do trabalho descrito nesta Tese.

Antes de tudo agradeço à minha mulher Aldina e aos nossos filhos André e Adelaide, pelo apoio essencial à concretização desta Tese. Foram tempos difíceis em que tiveram de suportar as minhas ausências prolongadas, andar com a casa à costas, se adaptar a um novo país e até aprender uma nova língua. Por tudo isto, esta tese é também a vossa.

Agradeço ao Professor José Laginha Palma, meu professor e orientador de Mestrado e agora de Doutoramento, por me ter possibilitado esta maravilhosa aventura e se ter empenhado em obter os meios necessários a sua concretização. Agradeço-lhe igualmente a sua grande disponibilidade, conselhos e opiniões que me foi transmitindo ao longo desta caminhada.

Estou igualmente grato às pessoas com quem convivi por largos períodos na Faculdade de Engenharia da Universidade do Porto: Aristides Castro, Alexandre Lopes, Pedro Areal, Amadeu Borges, Delmar Jorge, Carlos Santos assim como a todos os restantes membros do CEsa, pelo bom ambiente de trabalho que sempre aí encontrei.

Je suis très reconnaissant envers Daniel Ruiz, mon Codirecteur de Thèse à l'INPT, pour avoir partagé avec moi beaucoup de ses innombrables connaissances en Algèbre Linéaire, ainsi que pour toutes ses précieuses corrections et suggestions indispensables à la rédaction de cette Thèse. Je lui remercie également l'amitié dont lui et sa famille ont toujours fait preuve le long de ses trois années.

Je remercie Michel Daydé, directeur du site IRIT de l'ENSEEIH et mon directeur de Thèse à l'INPT, pour m'avoir réuni toutes les conditions nécessaires au développement de mon travail dans son laboratoire, ainsi que pour avoir toujours trouvé le temps de faire une lecture finale des articles soumis aux conférences.

Je remercie aussi la sympathie avec laquelle j'ai été reçue dans l'équipe d'Algorithmes Parallèles et Optimisation du site IRIT de l'ENSEEIH, en particulier par les membres du projet Grid-Tlse. Patrick Amestoy, grand exemple de professionnalisme, qui a toujours eu un mot amical et d'encouragement sur mon travail. Ronan Guivarc'h, grand amateur de Football et Rugby, avec qui j'ai eu le plaisir d'assister à des grands matchs. Chiara Puglisi

qui, outre de nombreuses compétences, assurait aussi la pause café, moment très agréable du matin ou ont eu lieu de très intéressantes discussions. Et toutes les autres personnes également responsables pour la bonne ambiance de travail existant au quatrième étage de la rue d'Aubuisson, notamment Ahmed Touami, Aurelie Hurault, Abdou Guermouch, Stéphane Pralet, Luc Giraud, Christophe Hamerling et Marc Pantel.

I am also very grateful to the many different financing sources and institutions that made this work possible: under the PRODEP III program (action 5.3), I could get free of teaching duties at *Instituto Politécnico de Bragança* for a three-year period, during which this work was developed. Periods of staying in Toulouse were covered by the French Ministry of Education and Research, the Socrates program, and the joint program between ACCTI-*Instituto de Cooperação Científica e Tecnológica Internacional*, CNRS-*Centre National de la Recherche Scientifique* and French embassy, under the project untitled: Parallel Iterative Solvers for Solving Linear System in CFD.

Preface

The present doctoral thesis was developed under the joint scientific and administrative supervision between the INPT- *Institut National Polytechnique de Toulouse* and FEUP- *Faculdade de Engenharia da Universidade do Porto*. As set in the conditions of the agreement signed by the two institutions, upon successful oral examination, the text of the diplomas delivered by each institution shall specifically mention the thesis joint supervision and will bear the seal of both institutions. In what concerns the *Institut National Polytechnique de Toulouse*, it will state: *Doctorat de l' Institut National Polytechnique de Toulouse, Mention Informatique et Télécommunications*. In what concerns the *Universidade do Porto* it will state: *Doutor em Ciências da Engenharia*.

*À memória do meu avô Manuel,
por ter despertado em mim o gosto pelo saber.*

Contents

Abstract	i
Resumo	iii
Résumé	v
Acknowledgements	vii
Preface	ix
Nomenclature	xxi
1 Introduction	1
1.1 Objectives of this thesis	1
1.2 Motivation and preliminary work	2
1.3 Short bibliographic review	3
1.4 Contributions of this thesis	6
1.5 Brief outline of the thesis	7
2 Use of partial spectral information on a CFD application	9
2.1 Introduction.	9
2.2 The test problem	10
2.2.1 Fluid flow equations	10
2.2.2 Properties of the coefficient matrices	11
2.3 Methods to obtain and use the spectral information	12
2.3.1 Computation of the invariant subspaces	13
2.3.2 Deflation	14
2.3.3 Spectral Low Rank Update (SLRU) preconditioner	14
2.4 Experiments	15
2.4.1 Monitoring the BlockCGSI algorithm	15
2.4.2 Improving the convergence of the CG algorithm	20
2.5 Cost-benefit and concluding remarks	22
3 Inverse iteration and related methods.	25

3.1	Introduction.	25
3.2	Direct and inverse iteration methods	26
3.2.1	Rayleigh quotient	26
3.2.2	Power iteration	27
3.2.3	Inverse iteration	28
3.2.4	Rayleigh quotient iteration	29
3.3	Inexact inverse iteration	30
3.4	Jacobi-Davidson method	37
3.5	Subspace iteration	39
3.6	Inexact subspace iteration	41
3.6.1	Stabilized block Conjugate Gradient algorithm	41
3.6.2	Block Conjugate Gradient coupled with Subspace Iteration	45
4	Monitoring and improving the BlockCGSI algorithm	47
4.1	Introduction	47
4.2	The BlockCGSI algorithm	48
4.3	Convergence analysis	51
4.3.1	Inverse subspace iteration (outer loop)	51
4.3.2	The blockCG iteration (inner loop)	52
4.3.3	Stopping criterion for the blockCG	54
4.4	Improvements of the BlockCGSI algorithm	56
4.4.1	Chebyshev based filtering technique	57
4.4.2	Sliding window	58
4.5	Operation counts	59
4.6	Exploiting the spectral information	61
4.6.1	Deflated starting guess	62
4.6.2	Spectral Low Rank Update (SLRU) preconditioner	62
4.6.3	Practical considerations	63
4.7	Numerical experiments	64
4.8	Cost-benefit analysis	68
5	Validation of the strategy on an airflow problem	73
5.1	Description of the problem	73
5.2	Numerical solution of the Navier-Stokes equations	74
5.3	Properties of the coefficient matrix	75
5.4	Improving the systems solution with an inaccurate basis	77
5.5	Optimal dimension of the basis	79
5.5.1	Pre-computational cost	80
5.5.2	Solution cost	81
5.5.3	Minimizing the total cost function	83
5.6	Costs-benefits of the two-phase approach	84
6	Closure	87

<i>Contents</i>	xv
6.1 Conclusions	87
6.2 Future work	90
Bibliography	93
Publications	97

List of Figures

2.1	Two different residual measure for the convergence of the blockCG with block size $s = 5$ and $s = 10$	17
2.2	Evolution of the error bound (2.10) associated to the five smallest eigenvalues, computed with the BlockCGSI algorithm.	18
2.3	Classical Conjugate Gradient with the SLRU preconditioner.	20
2.4	Classical Conjugate Gradient with the deflation of the starting guess. . . .	21
4.1	Correlation between the residual measures ω_1 , ϕ_1 and the upper bound S_1 on the spectral error $ \lambda_1 - \delta_1 $ in the blockCG with block size $s = 4$ at the first inverse iteration. Matrix A preconditioned with Jacobi scaling.	65
4.2	Convergence behavior of INIT-CG and SLRU-CG with different dimensions q of the pre-computed <i>near</i> -invariant subspace. The system is initially preconditioned either with Jacobi scaling (top) or with the standard Incomplete Cholesky (bottom).	68
4.3	Effect of the accuracy of the spectral information on the convergence behavior of INIT-CG.	69
4.4	History of the backward error as a function of the computational cost, for different dimensions q of the <i>near</i> -invariant subspace. The case $q = 0$ corresponds to the classical CG algorithm, for comparison.	70
5.1	Pattern of the coefficient matrix A	76
5.2	Distribution of the 100 smallest and 5 largest eigenvalues of the coefficient matrix.	77
5.3	Effect of the accuracy of the spectral information on the convergence of INIT-CG.	78
5.4	Convergence in number of iterations of the restarted INIT-CG.	78
5.5	Convergence in floating points operations of the restarted INIT-CG.	79
5.6	Costs ($\mathcal{C}_{BCGSI}$) of pre-computing the <i>near</i> -invariant subspace W with different block sizes s	81
5.7	Solution costs and amortization right-hand sides of INIT-CG.	82
5.8	Total costs of the two-phase approach (block size $s = 10$ used for the pre-computation).	83

List of Tables

2.1	Properties of the coefficient matrices of the velocity and pressure systems with Dirichlet boundary conditions, at the first SIMPLE iteration.	11
2.2	Extremal eigenvalues of the pressure system solved in the first global iteration, before and after preconditioning.	12
2.3	Cosine of the angles between the invariant subspaces associated with the 20 smallest eigenvalues.	12
2.4	Total number of matrix-vector products performed by the BlockCGSI algorithm as a function of the number of inverse iterations.	19
2.5	Estimate of the relative residual associated with the smallest eigenvalue λ_1 as a function of the number of inverse iterations and of the different threshold values ϵ	19
2.6	Costs and benefits of CG accelerated with the SLRU preconditioner.	23
2.7	Costs and benefits of CG accelerated by the deflation on the starting guess.	24
4.1	Basic operations counts in BlockCGSI algorithm	60
4.2	Cost in floating-point operations for different methods.	63
4.3	Properties of the test matrix $M^{-1}A$ with two different preconditioners.	64
4.4	Iterations and operations counts as a function of the filtering level ξ . Number of correct digits ($t = 1$), $q = \text{numb. of eigs.} \in]0, \mu[$	66
4.5	Cost-benefit of CG accelerated with the spectral information	71
5.1	Cost-benefit of the two-phase accelerating technique.	84

Nomenclature

Roman Characters

A	coefficient matrix with size $n \times n$
A^T	transposed matrix
A^{-1}	inverse matrix
b	vector with size n
B	right-hand sides matrix in the blockCG algorithm with size $n \times s$
$\mathcal{F}_m$	polynomial function of degree m
I	identity matrix
$\mathcal{K}_i$	i -dimensional Krylov subspace
L	lower triangular matrix
M	preconditioning matrix with size $n \times n$
n	size of the coefficient matrix A
nnz	non-zero elements in the matrix
$\mathbb{N}$	natural numbers set
$P^{(i)}$	conjugates directions matrix in the i -iteration of the blockCG algorithm with size $n \times s$
R	upper triangular matrix
$R^{(i)}$	residuals matrix in the i -iteration of the blockCG algorithm with size $n \times s$
$\mathbb{R}$	real numbers set
s	block size in the blockCG algorithm

$S^{(k)}$	spectral residual vector associated with the eigenpair (δ, v) in the k -outer (inverse) iteration
t	upper bound on the number of correct digits in the converged Ritz value δ (e.g. $ \lambda - \delta / \delta < 10^{-t}$)
T_m	Chebyshev polynomial of degree m
u_j	eigenvector of a symmetric matrix A
v_j	Ritz vectors
$w(\lambda)$	eigenvalue mapping function
x^*	exact solution vector of the system $Ax = b$
$x^{(i)}$	approximate solution vector in the i -th iteration of the Conjugate Gradient algorithm
$z_j^{[i]}$	approximate solution vector of the j -th system, in the i -inner iteration of the blockCG algorithm

Greek Characters

α	square matrices of size $s \times s$
β	square matrices of size $s \times s$
γ	square matrices of size $s \times s$
δ_j	Ritz values
ϵ	threshold used to stop the inner iteration
ζ	square matrices of size $s \times s$
θ	angle
κ	spectral condition number
λ_j	eigenvalues of a SPD matrix A
$\lambda_{\max}$	largest eigenvalue of matrix A (also denoted by λ_n)
$\lambda_{\min}$	smallest eigenvalue of matrix A (also denoted by λ_1)
μ	cut-off value of the BlockCGSI algorithm (in the range $]0, \lambda_{\max}[$)
μ_f	cut-off value of the Chebyshev polynomial filter (in the range $]0, \lambda_{\max}[$)

ν	a scalar, or an arbitrary shift
ξ	filtering level of the Chebyshev polynomial filter (in the range $]0, 1[$)
$\sigma(A)$	set of all the eigenvalues (spectrum) of the matrix A
$\phi_j^{[i]}$	measure of proximity of the iterate $z_j^{[i]}$ from the corresponding eigenvector u_j in the i -th inner iterations
$\omega_j^{[i]}$	backward error measure associated with the approximate solution vector $z_j^{[i]}$

Subscripts and Superscripts

i	inner iteration or iteration of the CG, blockCG and SIMPLE algorithms
j	position of the corresponding eigenvalue in $\sigma(A)$
k	outer iteration or iteration of the blockCG algorithms

Norms and Operators

$\ x\ _2$	Euclidean norm of vector x
$\langle x, y \rangle$	inner (dot) product (e.g. $x^T y$)
$\ x\ _M$	M -norm of vector x (e.g. $x^T M x$), with M SPD
$\ A\ _2$	2-norm of matrix A
$\vec{\text{grad}}$	gradient
$\vec{\text{rot}}$	rotational
div	rotational
∇^2	Laplacian

Abbreviations

blockCG	<i>block Conjugate Gradient algorithm</i>
blockCGSI	<i>block Conjugate Gradient within the Subspace Iteration algorithm</i>
CG	<i>Conjugate Gradient algorithm</i>
CFD	<i>Computational Fluid Dynamics</i>
INIT-CG	<i>CG accelerated by a deflated initial guess</i>
PDEs	<i>Partial Differential Equations</i>
SIMPLE	<i>Semi Implicit Pressure Linked Equations</i>
SIP	<i>Strongly Implicit Procedure</i>
SLRU	<i>Spectral Low Rank Update preconditioner</i>
SLRU-CG	<i>CG accelerated by the SLRU preconditioner</i>
Span{...}	<i>Subspace generated by linear combinations of a set of vectors</i>
SPD	<i>Symmetric and Positive Definite</i>
SVD	<i>Singular Values Decomposition</i>
TDMA	<i>Tri-Diagonal Matrix algorithm</i>

Chapter 1

Introduction

1.1 Objectives of this thesis

Partial spectral information, associated with the smallest eigenvalues of a given matrix A , can be used to improve the solution of successive linear systems of equations with very similar coefficient matrices. This is specially the case 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. We propose a two-phase approach to address this kind of problems, in particular when the coefficient matrices are Symmetric and Positive Definite (SPD). The idea is to perform some partial spectral decomposition of the first matrix A in the sequence of linear systems, and to exploit this information to improve the convergence in the iterative solution of the following linear systems.

The proposed two-phase approach is similar to a direct method. It includes a preliminary “*factorization*” phase, also denoted as “*partial spectral factorization phase*”, followed by a “*cheap*” solution phase. As opposed to direct methods, both phases are based only on numerical tools that are usually exploited in iterative methods and, in particular, require only matrix-vector products. This offers the possibility of keeping the matrix in implicit form. This approach is also suitable for large scale problems where the direct factorization of the coefficient matrix may be difficult to achieve.

To perform the partial spectral factorization, which computes and stores a basis of a *near*-invariant subspace linked with the smallest eigenvalue of the given SPD matrix A , we propose an algorithm called BlockCGSI. This algorithm combines the (inverse) subspace iteration (see for instance (Parlett, 1998)) with the block Conjugate Gradient (blockCG) developed by Arioli et al. (1995).

For the computation of any further solution, we propose two different techniques where this pre-computed basis is used in different ways. The first one consists in deflating the

initial residual to compute a starting guess, before computing the remaining part of the solution with CG (Hestenes and Stiefel, 1952). The second one is to build a second level of preconditioning that shifts the smallest eigenvalues from close to zero to near one.

1.2 Motivation and preliminary work

The two-phase approach is a general purpose acceleration technique that can be applied in a large class of problems coming from different computational areas, like for instance fluid or structural mechanics. However our first steps in this direction were basically motivated by some applications coming from Computational Fluid Dynamics (CFD). We have indeed focused our attention on two particular fluid flow algorithms: the SIMPLE method (Patankar and Spalding, 1972) and the Fractional Step method (see for instance Ferziger (1998)). Both of them are used for the numerical simulation of the Navier-Stokes equations, where the time and space variables are discretized in different ways.

In each time step or global iteration of these algorithms, there are several linear systems of equations to be solved, corresponding also to an important part of the total computing time. Iterative solvers are normally used in such algorithms because the coefficient matrix is generally not assembled explicitly in a sparse matrix structure. Instead of that, only the non-zero diagonals are stored in separate arrays and a particular matrix-vector product subroutine is built on top of that. The main motivation for this choice is linked with the memory requirements for large scale simulations. However, this way of doing also induces some loss of efficiency in the matrix-vector multiplications, since the elements of the coefficient matrix must be accessed from different arrays, and this disables vectorization partly.

Most of the iterative methods used in these algorithms have been specially designed to solve systems resulting from the discretization of partial differential equations (see Ferziger (1998)), like for instance the Tri-Diagonal Matrix (TDA) algorithm or the Strongly Implicit Procedure (SIP). These methods are still in use because they match very well with the data structures from the simulation codes and are fancily implemented.

In previous work (Balsa, 2000), we compared the performance of some of these methods with the techniques based on Krylov subspaces. We studied, in particular, the systems originated by the SIMPLE method (Patankar and Spalding, 1972). In this algorithm, two kinds of linear systems $A_j x = b_j$ are solved in each main iteration j : one is related with the velocity field, and the other with the correction of the pressure field. In both, the coefficient matrix A_j and the right-hand side b_j are updated at each iteration. Despite this update, the spectral properties of the matrices A_j are approximately constant. Additionally, the systems derived from the pressure correction are symmetric indefinite and ill-conditioned. This occurs because the boundary conditions are of Von-Neumann type, in which case solvers like MINRES or SYMMLQ (Paige and Saunders, 1975) are preferable. It was

also observed that most of the Krylov solvers present some plateaus in their convergence history at the beginning, before reaching a good linear rate of convergence. We also tried the stationary iterative methods, like TDMA, which in turn always reduce the backward error by a constant factor all through the iterations, but that can still present a very slow convergence rate in general.

In Balsa and Palma (2003), we perform the same kind of comparison in an algorithm based on the Fractional Step method. In this algorithm, all the linear systems related with the pressure that are solved in each main iteration have the same coefficient matrix A . Just the right-hand side b_j is updated. It was observed that, despite the fact this matrix is not very much ill-conditioned, the CG method has some difficulties to converge, in particular at the beginning of the run. As opposed to that, the Strongly Implicit Procedure (SIP) converges very fast in the first iterations and stagnates afterwards. It was then shown, that if just a small reduction in the residual norm is enough, the SIP method can be preferable to CG.

After this initial experimental study we came to the conclusion that the computational time spent in the successive solution of linear systems could be reduced by a two-phase approach strategy including a partial spectral factorization. In the SIMPLE algorithm, in particular, the various coefficient matrices arising from the pressure correction step share an almost constant invariant subspace linked with the smallest eigenvalues in these matrices. In the Fractional-step algorithm, the coefficient matrices do not change from one time step to the other. In both cases, CG presents the same type of convergence behavior, with difficulties at the beginning of the run, and this is mostly related to the distribution of the smallest eigenvalues in these matrices (van der Sluis and van der Vorst, 1986).

1.3 Short bibliographic review

The Krylov solver of choice for the solution of large sparse and symmetric positive definite linear systems of the form

$$Ax = b \quad (1.1)$$

is the CG algorithm. This method constructs the i^{th} iterate $x^{(i)}$ as an element of

$$x^{(0)} + \text{Span}\{r^{(0)}, Ar^{(0)}, \dots, A^{(i-1)}r^{(0)}\} \quad (1.2)$$

so that the A -norm of the error, $\|e^{(i)}\|_A = \|x^{(i)} - x^*\|_A$, is minimized, where x^* is the exact solution of (1.1).

The convergence of the Conjugate Gradient method is governed by the eigenvalues distribution in A (see van der Sluis and van der Vorst (1986)). A coarse characterization is given by the bound on the rate of convergence of the CG given by Concus et al. (1976)

$$\|e^{(i)}\|_A \leq 2 \|e^{(0)}\|_A \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^i, \quad (1.3)$$

where $\kappa = \lambda_{\max}/\lambda_{\min}$ denotes the condition number of A . From this bound, it can be seen that increasing the size of the smallest eigenvalues will decrease substantially the upper bound (1.3) on the convergence rate of the Conjugate Gradient.

Additionally, in the presence of clusters of eigenvalues at the extreme of the spectrum of A , combined with some ill-conditioning, the classical Conjugate Gradient may show a convergence history with sequences of plateaus and sharp drops. These plateaus (see van der Sluis and van der Vorst (1986)) are well understood and correspond to the discovery of separate eigenvalues within a cluster. The problem is that these plateaus can be rather large, in terms of number of iterations, even when these extreme clusters incorporate only a few eigenvalues. Thus removing the bad effect of small eigenvalues in the matrix A might have a beneficial effect on the convergence. Preconditioning the system (1.1) helps to reduce the effect of the extremal eigenvalues. Unfortunately, several of the most commonly used preconditioners transform the matrix A to a matrix $M^{-1}A$ which has eigenvalues in a relatively small number of clusters, but which is still ill conditioned.

Several techniques have been proposed in the literature to remove the effect of the smallest eigenvalues on the convergence of the CG (see for instance (Giraud et al., 2004)). Most of these techniques consist in enforcing the CG to work in the orthogonal complement of an invariant subspace W associated with the smallest eigenvalues. All through the iterations, the computed residuals are maintained, implicitly or explicitly, in this orthogonal complement. In this way, it is expected that the reduced condition number of A , restricted to the orthogonal complement of W , is much lower than the original condition number. Consequently, the preconditioned Conjugate Gradient may behave as if the spectrum of A where reduced to only those eigenvalues with eigenvectors not lying in this subspace W . Some of the main techniques based on this general idea are discussed and compared in Giraud et al. (2004).

A first idea can be to decompose the solution of (1.1) in two parts. The first part can be obtained directly with a projection of the right hand side b onto the orthogonal complement of W . Therefore, the second part of this solution can be obtained with the use of Conjugate Gradient with the first part of this solution as a starting guess.

The other techniques make use of a spectral projector explicitly all through the iterations. The first one, called Deflated-Conjugate Gradient, is to augment the Krylov space generated in the Conjugate Gradient iterations with the basis W , and resumes in a deflation at each iteration of the generated Krylov vectors. The second one, also called Projected-Conjugate Gradient, is to run the Conjugate Gradient algorithm onto the projected system $\mathcal{P}Ax = \mathcal{P}b$, where $\mathcal{P}$ is a projector onto the orthogonal complement of W that commutes in some way with A . Details about these two previous techniques can be found in Saad

et al. (2000). The last technique, proposed by Carpentieri et al. (2003), aims at building a preconditioner in A that shifts, by adding the value 1, those eigenvalues linked to the *near*-invariant subspace W , and to run a preconditioned Conjugate Gradient in the usual way to compute the solution of (1.1).

All the techniques that improve the convergence of the Conjugate Gradient method by removing the effect of the smallest eigenvalues imply the pre-computation of a basis W of a *near*-invariant subspace linked with these eigenvalues. Normally, this pre-computation has a cost that is larger than the benefit resulting from the acceleration of the system solution. Thus, in order to amortize the pre-computational costs, these acceleration techniques are recommended only when several linear systems with the same (or approximately the same) coefficient matrix have to be solved consecutively.

The two-phase approach was initially proposed by Arioli and Ruiz (1995), to improve the consecutive solutions of linear systems with the same coefficient matrix A and changing right-hand sides. The matrix A is assumed to be SPD or convertible to an SPD matrix by some kind of preconditioner. In a first phase, a basis of a *near*-invariant subspace linked with the smallest eigenvalues is computed and stored. The computation of the basis of this subspace is made with an algorithm, called BlockCGSI, that combines the (inverse) subspace iteration (see for instance Parlett (1998)) with a stabilized version of the block Conjugate Gradient (blockCG), developed by Arioli et al. (1995). For the computation of any further solutions in the second phase of the approach, the computed basis is used to perform a deflation on the initial residual and the Conjugate Gradient computes the remaining part of the solution.

In the BlockCGSI algorithm proposed by Arioli and Ruiz (1995), the inversion of the coefficient matrix is made implicitly by means of an iterative solution with the blockCG. This method is particularly well designed for simultaneously computing multiple solution and suffers less from the particular numerical properties of the linear systems under consideration (e.g. presents a convergence history with plateaus of much reduced length). It also provides only approximate solutions, as opposed to a direct solver, which may affect the global convergence in the subspace iteration. Therefore, the difficulty is to find an appropriate stopping criterion for the blockCG that enables to reach sufficient accuracy to ensure the convergence of the subspace iteration towards the targeted invariant subspace, and, if possible, that contributes to minimize the total amount of work. Another limitation of this algorithm is that the dimension of the targeted approximate invariant subspace must be set at the beginning, through the choice of the working block size in the blockCG. Unfortunately, it is difficult to know *a priori* the best block size without additional information on the spectrum of the given matrix.

An alternative version of the two-phase approach was proposed in Arioli and Ruiz (2002), where the first phase is performed with an algorithm called Chebyshev-PSF (Chebyshev based Partial Spectral Factorization). This algorithm combines a filtering tool based on Chebyshev polynomials with a block-Lanczos type of algorithm. The Chebyshev poly-

nomials are used to “*damp*”, in a set of vectors, the eigencomponents associated with all the eigenvalues in some predetermined range. This increases the degree of collinearity of the filtered vectors with respect to the eigenvectors linked to the eigenvalues outside this predetermined range. Starting with these filtered vectors, the block-Lanczos process is then used to build a Krylov basis that is very rich with respect to the spectral information associated to the targeted eigenvectors.

Compared with the BlockCGSI algorithm, the Chebyshev-PSF method is more robust because it does not depend on the choice of the dimension of the initial basis. The approximate invariant basis is indeed enlarged iteratively with all the computed Krylov vectors until *near*-invariance is detected. However, memory requirements are not controlled/fixed explicitly and, in some extreme cases, where the number of eigenvalues outside the filtering/damping interval is large, this process may be out of interest if not enough storage is available. As opposed to that, the BlockCGSI algorithm always works with the same basis dimension and enables a precise control of the memory requirements.

1.4 Contributions of this thesis

We propose a two-phase approach technique to improve the successive solutions of linear systems with the same spectral properties and changing right-hand sides. For each phase in this technique, we have implemented algorithms and analyzed their behavior in function of the different input parameters. From both theoretical and experimental analyzes, we indicate how to monitor these two algorithms. We also present some applications, coming from the computational fluid mechanics area, to illustrate both the potential and the effectiveness of the strategy that is used to monitor the propose technique, regardless the distribution of the smallest eigenvalues in the given matrix.

The eigencomputation of a *near*-invariant subspace, linked with the smallest eigenvalues, is performed with a reformulated version of the BlockCGSI algorithm that combines the subspace iteration with the block Conjugate Gradient (blockCG). We analyze, from an inner-outer point of view, the convergence of the subspace iteration combined with the blockCG, and establish the relation between the reduction of the inner residual norm and the convergence of the outer process. From this, we propose a residual measure for the blockCG that is directly linked with the convergence of the subspace iteration, and that helps to minimize the total amount of work. We monitor the inner (blockCG) iteration with a threshold that is automatically set with respect to the accuracy wanted for the *near*-invariant subspace.

The developed theoretical analysis shows that the convergence of the subspace iteration does not require at all an extremely accurate solution in the linear systems. We prove that the spectral residual norm (that measures the invariance in the outer loop) decreases along the inner iterations with respect to the angle between the right-hand side and the current

approximate solution of the linear system. When this angle stagnates, no improvements can be obtained in the convergence of the outer process. This result differs from most of the inexact inverse iteration theoretical analyzes, which usually derive expressions involving the angle between the right-hand side and the targeted eigenvector.

To improve the global convergence of the BlockCGSI algorithm, we introduce a spectral filtering tool based on Chebyshev polynomials to build starting vectors with better spectral properties. With this tool, the subspace iteration starts to work with a set of vectors that are already purged from the eigeninformation linked with the largest eigenvalues. We also include the concept of “*sliding window*” as an algorithmic feature that enables the computation of a *near*-invariant subspace whose dimension is not directly fixed by the choice of the working block size in the blockCG.

After computing the *near*-invariant subspace, we accelerate the convergence of the CG algorithm in any consecutive run (time step) in two different ways, namely the INIT-CG and the SLRU-CG. We also analyze experimentally how to take advantage of this spectral information even when it is poorly accurate.

1.5 Brief outline of the thesis

We begin, in Chapter 2, by applying the two-phase accelerating technique in the pressure correction systems coming from the SIMPLE algorithm. Using the original version of the BlockCGSI algorithm, we compute, with different accuracies, the eigenspace associated with the smallest eigenvalues, and we improve the convergence of the CG by either building a spectral preconditioner that shift these eigenvalues from almost zero close to the unit value, or performing a deflation of the initial residual in order to remove part of the solution corresponding to the smallest eigenvalues. The efficiency of the two resulting techniques is then compared experimentally. We also highlight that the success of the two-phase approach depends strongly on the appropriate combined monitoring of the subspace iteration and of the blockCG convergence within these iterations.

In Chapter 3 we review the main idea behind the inverse iteration, which is the basis of the BlockCGSI algorithm. We focus in particular on the inexact inverse iteration where the inversion of the coefficient matrix A is performed implicitly through an iterative solution of the linear system of equations. This method has been studied in several publications in recent years, like for instance Golub and Ye (2000); Simoncini and Eldén (2002); Berns-Mueller et al. (2005), where many important results were established. For instance, it has been proved that the inexact inverse iteration can converge linearly at the same rate as the exact case, even if the system is not solved very accurately, and that the convergence rate can be given by the threshold parameter used to stop the inner iterative solver. It was also shown that the growth of the norm of the solution vector, when solving iteratively the linear system, is directly linked to the reduction of the eigenvalue residual norm in the

inverse iteration. Based on these ideas, different stopping criteria for the iterative method inside the inverse iteration have been proposed: based either on the observation of the stagnation of the approximate solution vector norm or on the standard backward error measure.

In Chapter 4, we exploit some of the ideas developed in Chapter 3 to understand the effect of the inaccurate solution of the linear systems on the global convergence of the BlockCGSI algorithm towards the targeted invariant subspace. We develop an analysis of the convergence of the BlockCGSI algorithm and indicate how to monitor the algorithm to achieve some precise accuracy in the computed spectral information. We also introduce some algorithmic features that improve the convergence of the algorithm, and overtake some limitations of the subspace iteration. Namely, a spectral filtering tool based on Chebyshev polynomials, that helps to speed the convergence and improves the monitoring of the iterations, and the concept of “*sliding window*” that enables to incorporate as many eigenvalues as required. We devote the section 4.7 of Chapter 4 to numerical experiments concerning the computational costs involved in the different steps of the new version of the BlockCGSI algorithm, and to illustrate the behavior of this two-phase approach in general. The spectral information computed with the new version of the BlockCGSI algorithm is used to improve the solution of SPD linear systems resulting from a PDE problem. As in Chapter 2, the extracted spectral information is used either to deflate the starting residual from the part of the solution corresponding to the smallest eigenvalues, or to build a second level of preconditioning called Spectral Low Rank Update preconditioner (see Carpentieri et al. (2003)). The differences between these two acceleration techniques are analysed and discussed in detail.

In Chapter 5, we validate the proposed two-phase strategy in the context of an airflow simulation based on the ICARE code (c.f. Martinat (2003)), developed at IMFT-*Institut de Mécanique des Fluides de Toulouse*. The code simulates the transition of the flow, around the wing of a plane, from the laminar to the turbulent state. The Navier-Stokes equations are solved by a semi-implicit finite element discretization in a 2D field, through a prediction-correction scheme, very close to the one used in the Fractional Step algorithm. We show how the Conjugate Gradient can be accelerated even when the computed spectral information is not accurate, and how the total amount of work evolves as a function of the dimension of the computed *near*-invariant subspace. Based on these observations, we suggest an adaptive strategy to determine the dimension of this subspace.

Chapter 6 is concerned with some final observations on the two-phase approach developed in this work and with some possible issues concerning future developments.

Chapter 2

Use of partial spectral information on a CFD application

2.1 Introduction.

The starting point for this work was the iterative solution of multiple systems of linear equations, all Symmetric and Positive Definite (SPD), that occur in a Computational Fluid Dynamic (CFD) code based on the SIMPLE fluid flow algorithm (Patankar and Spalding, 1972). This is what has been called a segregated approach, where the mass and momentum conservation equations are solved in sequence and in an iterative manner, as separate equation systems.

In every iteration i of the SIMPLE algorithm, a new system of linear equations

$$A_i x = b_i \tag{2.1}$$

must be solved, where both the coefficient matrix A_i and its right-hand side b_i are updated. We have observed that, despite of the updating of A_i and b_i between iterations, the characteristics of the invariant subspaces associated with the smallest eigenvalues in these varying linear systems did not change much. This is very important, because it is well-known that eigenvalues near the origin slow down the convergence of Krylov based methods, as with CG for instance, in the case of SPD systems (see Golub and van Loan (1996)). The idea was then to take advantage of this near-stability of the invariant subspace associated with the smallest eigenvalues, computing it only once and using it to improve the convergence of the CG method in subsequent iterations of the SIMPLE fluid flow algorithm.

The chapter is organized as follows. In section 2.2 we analyze the characteristics and the origin of the systems to be solved. Section 2.3 includes a short explanation of the techniques being used to evaluate and cancel the effect of the smallest eigenvalues. Section 2.4 is concerned with the results and their discussion. Finally, in section 2.5, we present the

costs and benefits of the proposed technique and conclude with some observations about the experiments and a list of other issues, still requiring further analysis.

2.2 The test problem

As a basis for our study, we used the coefficient matrices of the pressure p which result from the classical problem of the flow inside a square box with a sliding lid, for a Reynolds number ($Re = U_{lid}L/\nu$) of 100, based on lid velocity (U_{lid}) and box size ($L = 1$ m).

2.2.1 Fluid flow equations

The fluid flow governing equations are the continuity,

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \quad (2.2)$$

and the Navier-Stokes equations,

$$\frac{\partial(uu)}{\partial x} + \frac{\partial(uv)}{\partial y} = -\frac{1}{\rho} \frac{\partial p}{\partial x} + \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \quad (2.3a)$$

$$\frac{\partial(uv)}{\partial x} + \frac{\partial(vv)}{\partial y} = -\frac{1}{\rho} \frac{\partial p}{\partial y} + \nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) \quad (2.3b)$$

which establish the principles of mass (2.2) and momentum conservation (2.3) in a two-dimensional stationary flow of an incompressible constant-property Newtonian fluid. u and v are the velocity components along the x and y Cartesian coordinate directions, p is the pressure and ρ and ν are the density and the fluid's kinematic viscosity.

The value of the unknowns u, v and p was determined using the SIMPLE fluid flow algorithm (Patankar and Spalding, 1972), where the momentum and mass conservation principles are enforced by alternately solving, until convergence, equations (2.3) and a Poisson-type pressure-correction equation derived by algebraic manipulation of the continuity (2.2) and momentum (or velocity) equations (2.3). The advective and diffusive terms are discretized in a non-staggered grid (Rhie and Chow, 1983), using hybrid and second order central differencing, in a finite volume approach (see Ferziger and Perić (2002)), which yields a five-diagonal coefficient matrix. The hybrid scheme switches between upwind and central differencing schemes, depending on whether the Reynolds number based on the numerical cell size is higher or lower than 2.

The total number of global iterations performed by the SIMPLE algorithm, to reach convergence, is between 100 and 450, depending on the kind of formulation and of the under-relaxation factors in use (see for instance McGuirk and Palma (1993)). In our test problem it converge in 250 iterations.

2.2.2 Properties of the coefficient matrices

The domain was discretized with a regular mesh of 120×120 nodes, with Dirichlet's boundary conditions. The result was a five-diagonal matrix of dimension $n = 13924$, with $nnz = 69148$ non-zero elements. In table 2.1, we present some properties of the coefficient matrices for the momentum and pressure correction system, solved in the first global iteration.

Table 2.1: Properties of the coefficient matrices of the velocity and pressure systems with Dirichlet boundary conditions, at the first SIMPLE iteration.

System	Momentum u	Pressure p
n	13924	13924
nnz	69148	69148
$\ A\ _\infty$	$1.057e - 01$	$1.006e - 02$
$\ A\ _2$	$9.971e - 02$	$1.005e - 02$
$\kappa_2(A)$	$4.233e + 00$	$3.131e + 05$

The test systems used in this work are derived from the pressure-correction equation. As we can see in table 2.1, these systems are ill-conditioned, compared with the velocity systems, because their smallest eigenvalues are much closer to the origin. The linear systems involving the velocities appear to be rather well conditioned, even with other grid sizes that we have tried in some preliminary tests (Balsa, 2000). The pressure systems are positive definite because we considered Dirichlet's boundary conditions for the pressure-correction equation. In the case of Neumann conditions on all boundaries, these systems would have been singular and more difficult to solve.

Some of the eigenvalues of the pressure system, computed with ARPACK (Lehoucq et al., 1998), are presented in table 2.2. These values belong to the systems solved at the first global iteration ($A_1 x = b_1$), and to the same system preconditioned with M_1 obtained with the standard Incomplete Cholesky (IC). As we can see, the IC(0) preconditioner clusters the eigenvalues near one, but it is not very effective in reducing the spectral condition number, which changes by one order of magnitude only.

In table 2.3, we present the minimum cosine of the angles between the two subspaces generated by the eigenvectors associated with the 20 smallest eigenvalues of $M_1^{-1}A_1$ and $M_p^{-1}A_p$. The subscripts 1 and p refer to the global iterations in the SIMPLE algorithm and M_1 and M_p are the corresponding IC preconditioners. The computation is made using the Singular Value Decomposition (SVD), as proposed in Golub and van Loan (1996), where $SVD(V_1^T V_p)$ gives the cosines of the angles between the two subspaces $Range(V_1)$ and $Range(V_p)$ (provided the columns of $Range(V_1)$ and $Range(V_p)$ are orthonormal sets of vectors), and the minimum of these values denotes the maximum angle. The eigenvectors, located in the columns of the matrices V_1 and V_p , have been computed with ARPACK.

Table 2.2: Extremal eigenvalues of the pressure system solved in the first global iteration, before and after preconditioning.

Eigenvalue	A_1	$M_1^{-1}A_1$
λ_1	$3.211006e-08$	$4.428912e-05$
λ_2	$8.877776e-07$	$1.232523e-03$
λ_3	$9.572325e-07$	$1.328789e-03$
λ_4	$1.769538e-06$	$2.473183e-03$
λ_5	$3.550269e-06$	$4.915760e-03$
λ_{10}	$7.985443e-06$	$1.100848e-02$
λ_{15}	$1.164407e-05$	$1.611350e-02$
λ_{20}	$1.591812e-05$	$2.193773e-02$
λ_{max}	$1.005278e-02$	$1.226147e-00$
κ_2	$3.130730e+05$	$2.768510e+04$

Despite the changes in the systems in each global iteration, the spectral characteristics of the coefficient matrices A_i do not change much along the global iterations. Indeed, as we can see in table 2.3, the angles between the invariant subspaces associated with the smallest eigenvalues stay very close to zero.

Table 2.3: Cosine of the angles between the invariant subspaces associated with the 20 smallest eigenvalues.

$\min\{SVD(V_1^T V_p)\}$	$M_1 = M_p = I$	$M_1 = IC(A_1), M_p = IC(A_p)$	$M_1 = M_p = IC(A_1)$
$\min\{SVD(V_1^T V_{10})\}$	0.9999999211680	0.9999999106341	0.9999999123822
$\min\{SVD(V_1^T V_{100})\}$	0.9999999704539	0.9999999236925	0.9999999693204
$\min\{SVD(V_1^T V_{200})\}$	0.9999999025570	0.9999999240168	0.9999999946030

2.3 Methods to obtain and use the spectral information

As mentioned before, we are concerned with the consecutive solution of several linear systems with different right-hand sides and different coefficient matrices, but having spectral characteristics approximately constant. To benefit from this, we propose a two phase approach that firstly computes a part of the spectral information associated with the smallest eigenvalues in the first system, and which afterwards uses this information to compute the solution of the remaining systems.

In this section, we describe the method for computing the eigenvalues and the eigenvectors of the matrix A_1 , which is based on the (inverse) subspace iteration. Next, we discuss

two techniques, both exploiting the precomputed invariant subspace, that can be used to improve the CG convergence: first, the deflation of the starting guess and second, the use of a spectral preconditioner.

2.3.1 Computation of the invariant subspaces

To compute the spectral information we have implemented the Subspace Iteration in conjunction with the stabilized block Conjugate Gradient method (blockCG) to compute the required sets of multiple solutions, as proposed in Arioli and Ruiz (1995) where this combination is called BlockCGSI algorithm. This algorithm approximates s eigenvectors associated to the s smallest eigenvalues of the preconditioned pressure matrix $M^{-1}A$, at the first SIMPLE iteration.

ALGORITHM 2.3.1: BLOCKCGSI ALGORITHM

Inputs: $A, M = R^T R \in \mathbb{R}^{n \times n}, s \in \mathbb{N}, m \in \mathbb{N}$

Output: a *near*-invariant subspace W with dimension s

Begin

Generate the initial subspace

$Z^{(0)} = \text{RANDOM}(n, s)$

$V^{(0)} \Gamma = Z^{(0)}$ such that $V^{(0)T} M V^{(0)} = I_{s \times s}$

For $k = 1, \dots, m$ Do:

Solve $M^{-1} A Z^{(k)} = V^{(k-1)}$ with blockCG

$Q^{(k)} \Gamma_k = Z^{(k)}$ such that $Q^{(k)T} M Q^{(k)} = I_{s \times s}$

$\beta_k = Q^{(k)T} A Q^{(k)}$

Diagonalize $\beta_k = U_k \Delta_k U_k^T$

where $U_k^T = U_k^{-1}$

and $\Delta_k = \text{Diag}(\delta_1, \dots, \delta_s)$ (Ritz Values)

$V^{(k)} = Q^{(k)} U_k$ (Ritz Vectors)

EndDo

End

In algorithm 2.3.1, the preconditioned blockCG algorithm is used to solve the s linear systems $M^{-1} A z_j^{(k)} = v_j^{(k-1)}$, with $j = 1, \dots, s$, simultaneously, where the matrix A is

preconditioned with M (standard IC preconditioner). The block Conjugate Gradient algorithm under concern is a numerically stable variant that avoids the numerical problems that can occur when some of the s systems are about to converge (Arioli et al., 1995). This stabilized variant is detailed in chapter 3.2.3.

The block solution matrix $Z^{(k)}$, corresponding to the s right-hand sides (columns of $V^{(k-1)}$), is computed with an accuracy determined by a threshold value ϵ , which is an input of the algorithm. The other input variables are m , the number of inverse iterations of the BlockCGSI algorithm, and s , the block size, which defines the size of the computed invariant subspace. These parameters can not be set in an optimal manner *a priori* because this depends on the kind of problem and on the desired final accuracy. This depends also on how this computed spectral information will be exploited in the following consecutive solutions. In section 2.4, we investigate experimentally the impact of varying the values of m and s in the context of the current problem.

2.3.2 Deflation

Once the invariant subspace linked to the smallest eigenvalues of the linear system solved in the first global iteration is obtained, we can use it for the computation of the solution of the systems in each of the following global iterations. The idea is to use this spectral information to remove the effect of the ill conditioning in these linear systems.

One of the possible methods is to perform a deflation on the initial residual. This enables to obtain the eigencomponents of the solution corresponding to the smallest eigenvalues. 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. In this way, the Conjugate Gradient should reach linear convergence immediately (van der Sluis and van der Vorst, 1986).

In each global iteration i of the SIMPLE algorithm, we run the Conjugate Gradient to solve $M_1^{-1}A_i x = M_1^{-1}b_i$ with the starting guess

$$x^{(0)} = V^{(m)} \Delta_m^{-1} V^{(m)T} b_i, \quad (2.4)$$

where $V^{(m)}$ and Δ_m are the matrices corresponding to the Ritz vectors and values obtained after m inverse iterations of the BlockCGSI algorithm on the preconditioned matrix $M_1^{-1}A_1$.

2.3.3 Spectral Low Rank Update (SLRU) preconditioner

Another way of exploiting spectral information of the coefficient matrix is to perform a deflation at each CG iteration, instead of just at the beginning. This approach, proposed in Carpentieri et al. (2003), is called Spectral Low Rank Update (SLRU) preconditioning.

The computation of the solution of the systems $M_1^{-1}A_i x = M_1^{-1}b_i$, is obtained by running the CG algorithm on an equivalent system $MA_i x = Mb_i$, where the preconditioner M is given by

$$M = M_1^{-1} + V^{(m)}\Delta_m^{-1}V^{(m)T}. \quad (2.5)$$

This preconditioner will shift the smallest eigenvalues in the coefficient matrix $M_1^{-1}A_i$ close to one (see Carpentieri et al. (2003)). In some cases, it can be interesting to shift the smallest eigenvalues close to some predetermined value λ (with $\lambda = \lambda_{max}$, for instance), in which case the spectral preconditioner must be set to

$$M = M_1^{-1} + \lambda V^{(m)}\Delta_m^{-1}V^{(m)T}. \quad (2.6)$$

This is not useful in our test problem, since the spectrum is previously clustered near one with the first level of preconditioning M_1 (see table 2.2).

In comparison with the deflation on the starting guess, presented in equation (2.4), the SLRU preconditioner exploits the precomputed spectral information at each CG iteration, which may have an effect in terms of robustness but also in terms of total amount of work.

2.4 Experiments

In this section, we illustrate the potential of the methods presented above, in the context of the SIMPLE algorithm. We analyse the computation phase of the spectral information and its consequences on the acceleration of the CG convergence. All the tests have been performed on a Personal Computer with an AMD® Athlon® Processor, with clock frequency of 1.6 MHz and 250 MB of RAM memory. All runs have been performed with Matlab® R12.

We compute, using algorithm 2.3.1, the s smallest eigenvalues in the system $A_1 x = b_1$, with s varying from 5 to 20, and then we run the CG algorithm on all the sample systems. In each run of the classical Conjugate Gradient algorithm, preconditioned with Incomplete Cholesky, we compare the performance improvements obtained with the two different techniques introduced above, namely the SLRU preconditioner and the deflation of the smallest eigenvalues in the starting guess.

2.4.1 Monitoring the BlockCGSI algorithm

The main parameters of algorithm 2.3.1, that must be carefully chosen, are the threshold value (ϵ) for the stopping criterion of the blockCG algorithm, the dimension of the computed invariant subspace (block size s), and the number of inverse iterations (m).

The accuracy in the computation of the multiple solutions required in the subspace inverse iterations, obtained by means of the stabilized blockCG, is set by the threshold

value ϵ used in the stopping criterion. The choice of this stopping criterion is indeed a crucial point because it can lead to a great amount of unnecessary extra work, proportional to the block size s . We recall that the linear systems that we need to solve at each inverse iteration is of the type

$$M^{-1}Az = v,$$

which is equivalent to the symmetrized system $R^{-T}AR^{-1}Rz = Rv$, assuming that the SPD preconditioner is factorized as $M = R^T R$. Based on the ideas developed by Rigal and Gaches (1967), to monitor the convergence we have chosen the normwise backward error

$$\omega_1 = \frac{\|Rv - R^{-T}Az\|_2}{\|R^{-T}AR^{-1}\|_2\|Rz\|_2 + \|Rv\|_2}, \quad (2.7)$$

where z denotes the current iterate approximating the solution of the system above, and we stop the iterations when this backward error is smaller than some threshold ϵ . Assuming that the preconditioned matrix $R^{-T}AR^{-1}$ has a maximum eigenvalue close to one – which is the case in general with classical preconditioning techniques, and can at any rate be achieved with some additional scaling factor – and considering that the right-hand side vector v has an M -norm equal to one, we also mention that this backward error measure can be simplified to

$$\omega_1 = \frac{\|v - M^{-1}Az\|_M}{\|z\|_M + 1}. \quad (2.8)$$

The reason for this choice is that it is the measure of the residual norms in the symmetrized system $R^{-T}AR^{-1}Rz = Rv$ that bounds effectively the error on the approximated eigenvalues. Indeed, at each inverse subspace iteration k , the Ritz values $\delta_1^{(k)}, \dots, \delta_s^{(k)}$ (diagonal elements of Δ_k) ranged in increasing order, and the corresponding Ritz vectors $v_1^{(k)}, \dots, v_s^{(k)}$ (columns of $V^{(k)}$), approximate the eigenvalues $\lambda_1, \dots, \lambda_s$ and the eigenvectors $u_1, \dots, u_s$, of $M^{-1}A$. The error bound on each eigenpair, given by (Parlett, 1998, p. 357), is:

$$|\lambda_j - \delta_j^{(k)}| \leq \frac{\|Av_j^{(k)} - \delta_j^{(k)}Mv_j^{(k)}\|_{M^{-1}}}{\|Mv_j^{(k)}\|_{M^{-1}}}, \quad (2.9)$$

which also corresponds to

$$|\lambda_j - \delta_j^{(k)}| \leq \|M^{-1}Av_j^{(k)} - \delta_j^{(k)}v_j^{(k)}\|_M \quad (2.10)$$

(since $v_j^{(k)}$ has an M -norm equal to 1), and incorporates the same matrix-norm as in the backward error measure (2.8). In that respect, we may expect a better agreement between the choice of the threshold value ϵ used to stop the blockCG iterations and the final level of accuracy that can be reached in the approximated eigenvalues.

To start with the illustration of the different issues discussed above, we first show in figure 2.1 the convergence history of the backward error ω_1 within the blockCG run at the first inverse iteration ($k = 1$) in the BlockCGSI algorithm. We indicate in the

X-axis the number of matrix-vectors multiplications, which correspond to the iteration count multiplied by the block size s (fixed to $s = 5$ and $s = 10$ in the two plots). The convergence history that is shown is relative to the solution vector z_1 associated with the smallest eigenvalue λ_1 , because its computation is the most sensitive to round-off error propagation. Just for sake of comparison, we also plot the convergence history of the classical standard relative residual ratio:

$$\omega_2 = \frac{\|M^{-1}Az - v\|_2}{\|v\|_2}. \quad (2.11)$$

We can observe that the choice of the criterion based on (2.11) can increase substantially the number of blockCG iterations for the same fixed level ϵ . For example, with a threshold

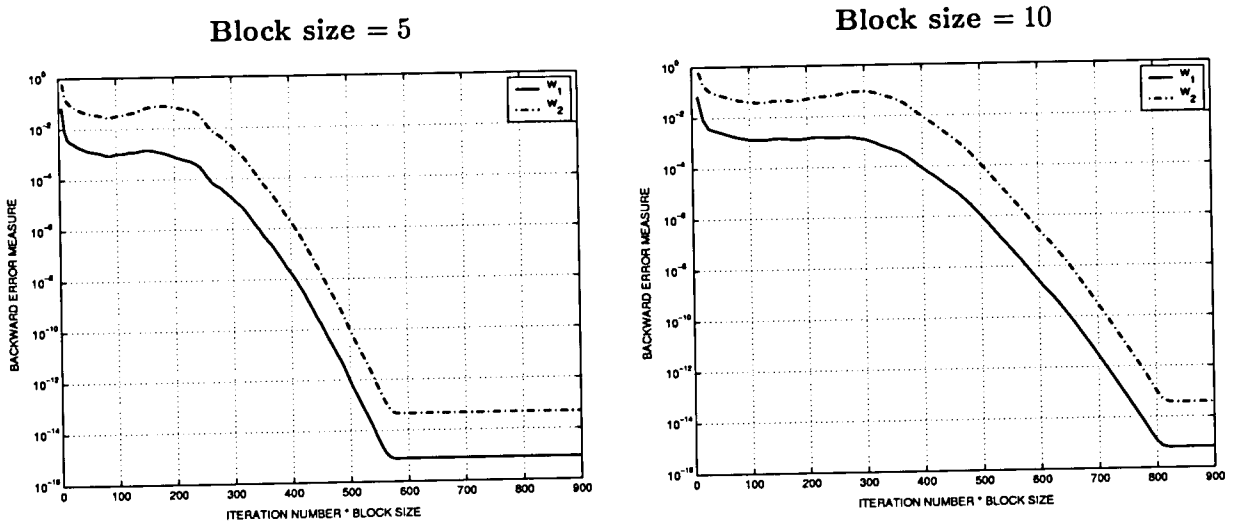


Figure 2.1: Two different residual measure for the convergence of the blockCG with block size $s = 5$ and $s = 10$.

value $\epsilon = 10^{-4}$ and a block size $s = 10$, the use of a residual norm (2.11) instead of (2.8) increases the total number of matrix-vector multiplications from 400 to 500 (see figure 2.1). It is also important to note that the amount of matrix-vector multiplications does not grow proportionally with the block size s .

One of the most crucial issues in these experiments is the good adequation between the threshold value ϵ and the level of stagnation of the error bound (2.10) in the inverse iterations. In figure 2.2, we can actually see that this error bound decreases very fast as the number of inverse iterations grows, and stagnates around a level very close to the threshold value ϵ . This justifies experimentally the discussion we had before concerning our choice for the backward error measure (2.8), and gives the potential for an easy control of the numerical behavior of algorithm 2.3.1. Finally, the stagnation of this bound after a certain number of inverse iterations is a direct consequence of the level of accuracy in the computed solutions. For example, if we want to approximate the five smallest eigenvalues

with a threshold value $\epsilon = 10^{-10}$, the BlockCGSI algorithm stagnates after four inverse iterations (see figure 2.2), and after this stage, the blockCG needs only one iteration to reach the level ϵ (see table 2.4). This occurs simply because there is no more refinement possible with the same tolerance ϵ .

Dividing equation (2.10) by $\delta_j^{(k)}$ instead of λ_j , we can get an estimate of the relative residual associated to λ_j , which indicates the number of correct digits in each approximated eigenvalue $\delta_j^{(k)}$. In table 2.5, we give the value of this estimate corresponding to $\delta_1^{(k)}$ (obtained with a block size $s = 5$), as a function of the number of inverse iterations and for different values of the threshold ϵ . For instance, with $\epsilon = 10^{-10}$, the BlockCGSI algorithm gives an approximation of the smallest eigenvalue with six correct digits after four inverse iterations.

Concerning the choice of the block size s , we must consider different conflicting aspects in which there is some compromise to be reached. On the one hand, the more eigenvectors associated to the smallest eigenvalues are approximated, the faster should be the convergence of the CG algorithm in the solution phase, because we expect that the CG algorithm will work as if the condition number was reduced to about $\lambda_{\max}/\lambda_{s+1}$. But on the other hand, as s grows, the work to compute the spectral information also grows. However, and this has been mentioned in Arioli and Ruiz (1995), this computational work does not grow proportionally with the block size, and it is thus important to find out the appropriate number of eigenvalues to approximate.

The last important aspect concerning the choice of the block size s , is that the convergence of the inverse iterations in the BlockCGSI algorithm can also be improved with

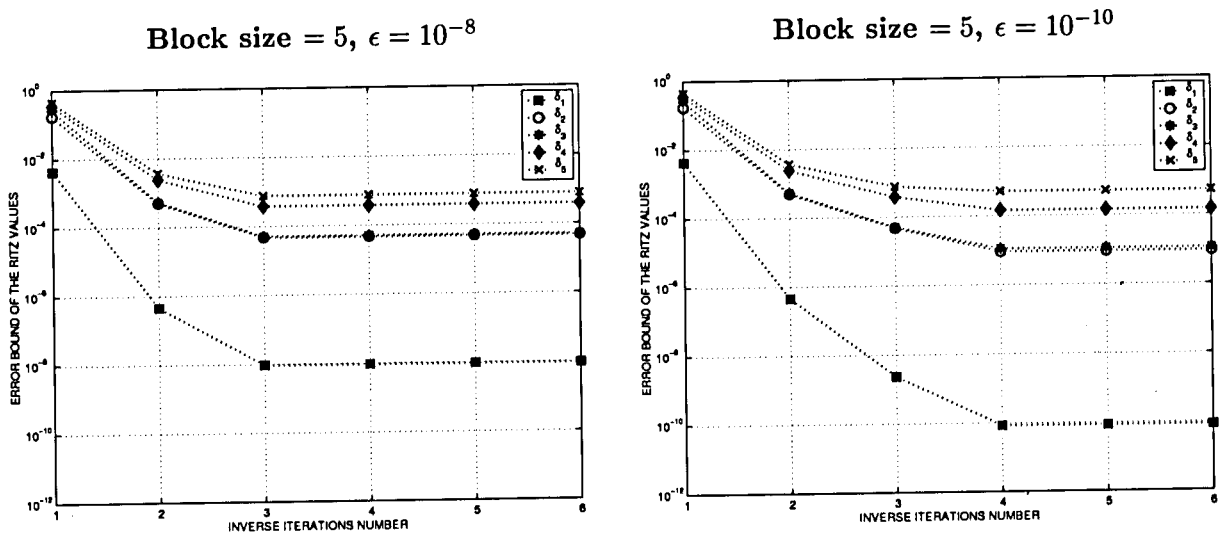


Figure 2.2: Evolution of the error bound (2.10) associated to the five smallest eigenvalues, computed with the BlockCGSI algorithm.

2.4. Experiments

Table 2.4: Total number of matrix-vector products performed by the BlockCGSI algorithm as a function of the number of inverse iterations.

Inv. Iters.	Number of BlockCG Iterations * Block Size															
	$\epsilon = 10^{-4}$				$\epsilon = 10^{-6}$				$\epsilon = 10^{-8}$				$\epsilon = 10^{-10}$			
	Block Size				Block Size				Block Size				Block Size			
	5	10	15	20	5	10	15	20	5	10	15	20	5	10	15	20
1	260	400	480	560	345	500	600	600	410	580	600	600	455	600	600	600
2	50	70	90	120	215	310	360	420	305	430	495	600	365	510	600	600
3	10	10	15	20	10	20	15	20	165	220	210	260	255	350	390	460
4	5	10	15	20	5	10	15	20	5	10	15	20	120	130	15	100
5	5	10	15	20	5	10	15	20	5	10	15	20	5	10	15	20
6	5	10	15	20	5	10	15	20	5	10	15	20	5	10	15	20

Table 2.5: Estimate of the relative residual associated with the smallest eigenvalue λ_1 as a function of the number of inverse iterations and of the different threshold values ϵ .

Inverse Iterations	$\ M^{-1}Av_1^{(k)} - \delta_1^{(k)}v_1^{(k)}\ _M/\delta_1^{(k)}$			
	$\epsilon = 10^{-4}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-8}$	$\epsilon = 10^{-10}$
1	6.71101e + 01	6.71317e + 01	6.71317e + 01	6.71317e + 01
2	2.25445e + 00	2.35220e - 02	1.47802e - 02	1.02337e - 02
3	1.60105e + 00	1.86759e - 02	2.81274e - 04	5.23205e - 05
4	1.60105e + 00	1.86759e - 02	2.15853e - 04	1.85662e - 06
5	1.60105e + 00	1.86759e - 02	2.15853e - 04	1.85662e - 06
6	1.60105e + 00	1.86759e - 02	2.15853e - 04	1.85662e - 06

larger values of s . Indeed, the subspace computed in the BlockCGSI algorithm converges towards an invariant subspace of the iteration matrix A with a rate that depends on the relative gap between the approximated eigenvalues and the closest in the remaining ones. As it was shown in Parlett (1998), this rate of convergence is governed by the relation:

$$|\lambda_j - \delta_j^{(k)}| \sim \left(\frac{\lambda_j}{\lambda_{s+1}} \right)^k, \quad 1 \leq j \leq s, \quad (2.12)$$

and it may be useful in some cases to increase the block size s just to benefit from a better gap in the relation above. This technique is also denoted “Guard Vectors”. Nevertheless, the relation (2.12) relies on the fact that the solutions in each inverse iteration are obtained as exactly as possible. This is not the case in our experiments where we actually play with rather large values for the threshold ϵ , expecting to reduce quite reasonably the total amount of work for very improvements in the solution phase, as this will be illustrated in the following sections.

2.4.2 Improving the convergence of the CG algorithm

Based on the precomputed spectral information, we can improve the convergence of the CG algorithm at each of the consecutive runs in the SIMPLE algorithm. To illustrate this, we have selected two pressure-correction systems coming from the global iterations 10 and 100 in the SIMPLE algorithm, and have solved these systems with the CG algorithm accelerated by means of the SLRU preconditioner or the deflation of the initial solution vector. The relative residual measure in these plots is a standard backward error measure based on $\|r^{(i)}\|_2/\|r^{(0)}\|_2$, mainly for reasons of simplicity in the discussion and also because very similar improvements can be observed with other type of residual measures that can be found in literature (see Arioli (2004) for instance).

The results show that, as expected, the spectral information calculated in the first

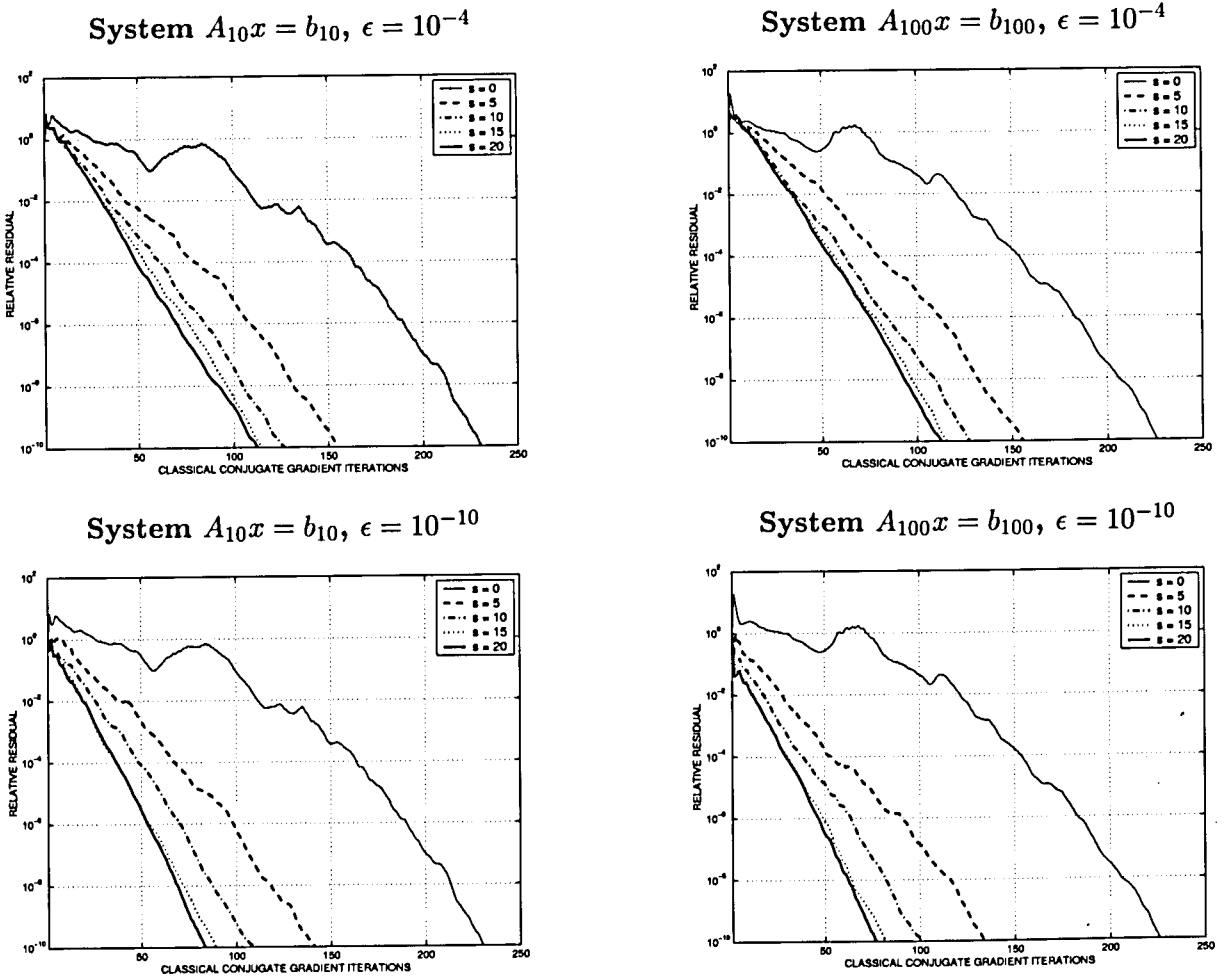


Figure 2.3: Classical Conjugate Gradient with the SLRU preconditioner.

global iteration is valid during the whole iterative process of the SIMPLE algorithm. In previous experiments, we observed that the reduction in the number of CG iterations is similar to the one obtained in the case of the two systems $A_{10}x = b_{10}$ and $A_{100}x = b_{100}$ presented here.

The Incomplete Cholesky preconditioner clusters the eigenvalues near one but still leaves a few ones remote (see table 2.2). These are shifted to the right of the unity by the SLRU preconditioner introduced in equation (2.5), yielding a lower condition number and, therefore, the performance of CG is improved. However, we can observe in figure 2.3 that, after removing the 15 smallest eigenvalues, no much better improvements are obtained on the convergence rate. This occurs because the reduced condition number $\lambda_{\max}/\lambda_{s+1}$ remains at the level of 10^2 when $s \geq 15$ (see table 2.2).

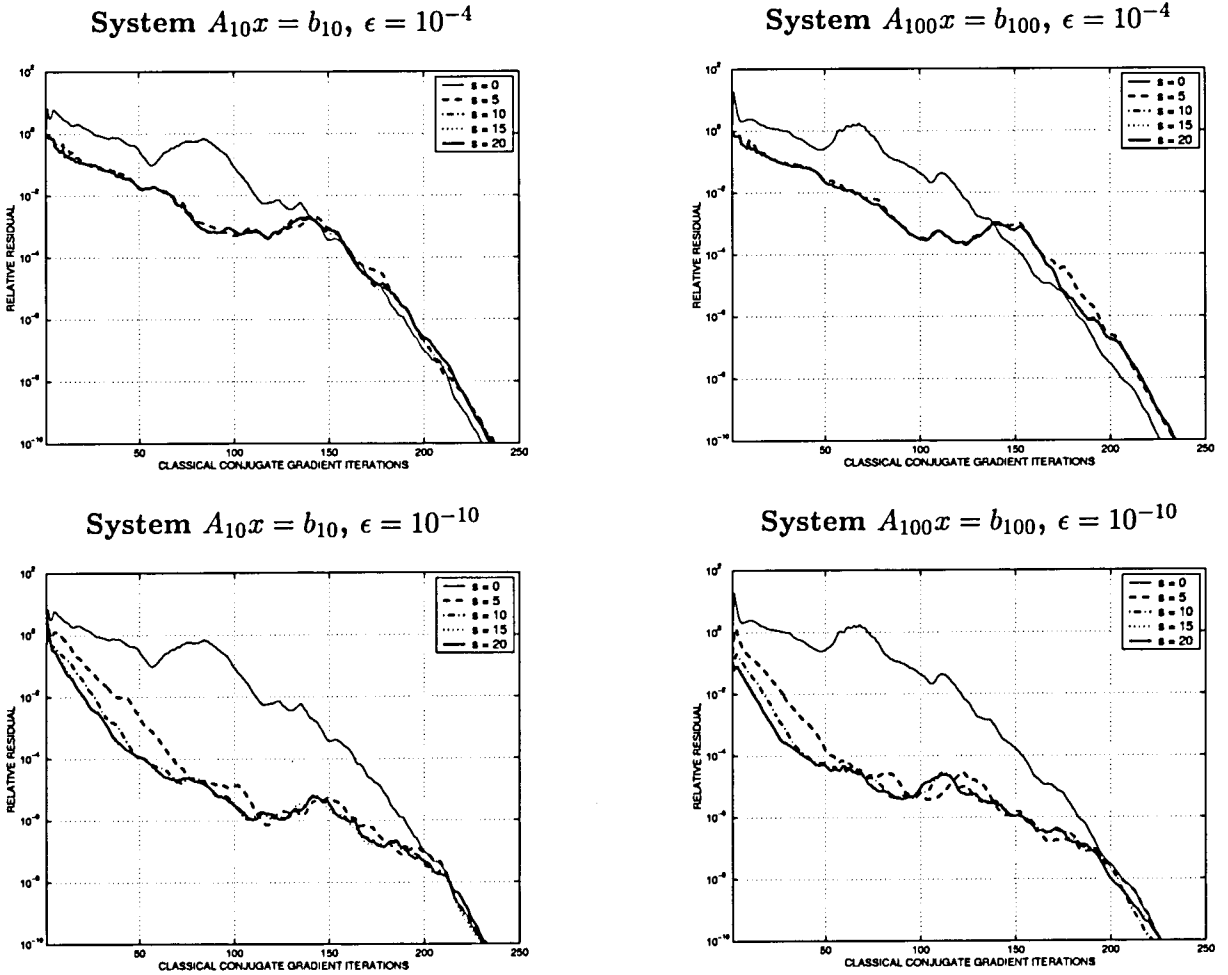


Figure 2.4: Classical Conjugate Gradient with the deflation of the starting guess.

An important property of the SLRU preconditioner is that it does not need a high accuracy in the spectral information to be efficient. As we can observe in figure 2.3, the curves obtained with $\epsilon = 10^{-4}$ and $\epsilon = 10^{-10}$ are very similar, despite the estimates in table 2.5 tend to indicate that δ_1 has no correct digit in the first case and 6 correct digits in the second case.

The alternative approach that we have experimented, to remove the effect of the smallest eigenvalues, is the deflation technique (see figure 2.4). As it was mentioned before, the idea is to include in the starting guess the information corresponding to the smallest eigenvalues. As opposed to the SLRU preconditioner the deflation of the starting guess shows a linear convergence only during the first few iterations, and requires also that the spectral information be computed with better accuracy (see figure 2.4). This suggests that the deflation technique is numerically more unstable than the SLRU preconditioner. Nevertheless, in these two acceleration techniques, the linear convergence is identical and is basically linked to the reduced condition number $\lambda_{\max}/\lambda_{s+1}$ as expected from the theory (see Giraud et al. (2004)).

2.5 Cost-benefit and concluding remarks

The framework in this chapter was to introduce some techniques that could help to reduce the computing time in a fluid dynamic code based on the SIMPLE fluid flow algorithm. We have stressed that the difficulties mostly occur when solving the SIMPLE pressure-correction equation. In general the linear systems that arise from the discretization of this equation present some eigenvalues very close to zero, but with associated invariant subspaces that do not vary much during the whole SIMPLE iterative process. This opened the way to strategies based on spectral information for improving the solution of these linear systems. 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.

Concerning the computation of the spectral information associated with the smallest eigenvalues in our linear systems, we have chosen a combination of the (inverse) subspace iteration algorithm with a stabilized version of the block Conjugate Gradient for the solution of the systems with multiple right-hand sides. The main reasons for the choice of this combination were to enable a precise control of the memory requirements, and to open the possibility of reducing the total amount of work by controlling the accuracy when solving the systems at each inverse iteration. We also mention that the blockCG algorithm can incorporate BLAS 3 Dongarra et al. (1990) operations and therefore be efficiently implemented on modern computers.

This eigencomputation has a cost that depends on the dimension and on the accuracy of the computed invariant subspace. Once this information is obtained, we can use it to accelerate the CG in the successive global iterations of the SIMPLE algorithm. The success of all this implies that the initial cost can be recovered by the reduction of the number of CG iterations in the following runs.

In tables 2.6 and 2.7, the cost-benefit of the two proposed techniques are illustrated. The system used is $A_{100}x = b_{100}$, and we stop the CG iterations when the relative residual norm $\|r^{(k)}\|_2/\|r^{(0)}\|_2$ is below 10^{-4} . This is a value which is not too low and is normally used in this type of simulation. In this context, the classical Conjugate Gradient algorithm converges in 153 iterations and 11.68 seconds. In these tables, we indicate first the total number of matrix-vector multiplications performed to get the spectral information, which we denote by “*Equivalent Iterations*”, i.e. an rough measure of the cost for computing the spectral information. In each case, the number of CG iteration and the time to converge are indicated, as well as the benefits with respect to the convergence without any acceleration (153 iterations and 11.68 sec.). Finally, the “*SIMPLE Amortization*” denotes the number of global iterations after which the extra cost in terms of matrix-vector multiplications for computing the spectral information is actually fully recovered. For instance, in table 2.6, for $s = 10$ and $\epsilon = 10^{-10}$, 1590 equivalent iterations are needed for the spectral pre-computation, out of which CG convergence is achieved in 38 iterations, i.e. a reduction of 75% compared to the run without spectral preconditioning. The 1590 extra equivalent iterations are payed back after 14 global iterations, and in the remaining global iterations of the SIMPLE algorithm, the total time spent in the solver will be halved.

Table 2.6: Costs and benefits of CG accelerated with the SLRU preconditioner.

s	ϵ	Eq. Iter. (costs)	CG + SLRU		Benefits		SIMPLE Amortization
			Its	Time	Its	Time	
5	10^{-4}	320	79	7.65	74	4.03	5
	10^{-10}	1195	53	5.18	100	6.50	12
10	10^{-4}	470	62	8.65	91	3.03	6
	10^{-10}	1590	38	5.32	115	6.36	14
15	10^{-4}	580	56	9.32	97	2.36	6
	10^{-10}	1590	30	4.96	123	6.72	13
20	10^{-4}	680	54	9.96	99	1.72	7
	10^{-10}	1760	29	5.32	124	6.36	15

As we can see in table 2.7, the efficient use of the deflation technique requires a greater initial cost, since it needs higher accuracy in the computation of the spectral information. However, the deflation technique just implies an initial extra operation, and is less costly than the SLRU preconditioner. For example, in the same situation ($s = 10$ and $\epsilon = 10^{-10}$), the deflation leads to the same reduction in iterations, but in terms of computing time, the reduction is approximately of 80% and bigger compared to the SLRU preconditioner.

Therefore, depending on the total number of global iterations that will be performed, the deflation technique can still be a good option.

Table 2.7: Costs and benefits of CG accelerated by the deflation on the starting guess.

s	ϵ	Eq. Iter. (costs)	CG + Defl.		Benefits		SIMPLE Amortization
			Its	Time	Its	Time	
5	10^{-8}	880	54	3.19	99	8.49	9
	10^{-10}	1195	53	3.16	100	8.52	12
10	10^{-8}	1230	46	2.83	107	8.85	12
	10^{-10}	1590	39	2.35	114	9.33	14
15	10^{-8}	1350	46	2.83	107	8.85	13
	10^{-10}	1590	38	2.35	113	9.33	15
20	10^{-8}	1460	46	2.83	107	8.85	14
	10^{-10}	1760	38	2.35	113	9.33	16

Finally, the success of this approach also depends on the appropriate combined monitoring of the inverse iterations and of the blockCG convergence within these iterations. We have proposed and validated experimentally a stopping criterion for the blockCG algorithm that presents good numerical properties to achieve such a goal. Of course, we still need to investigate this issue from the theoretical point of view, and this is done in chapter 4, in order to build an algorithm that can adapt automatically to the properties of the given matrices and compute invariant subspaces with some *a priori* fixed accuracy.

The results presented here show the effectiveness in reducing the computing time in what can be considered as a not too difficult fluid flow calculation. Our plan is also to investigate the potential of such an approach in the context of more complex numerical simulations, where the stability of spectral properties of the linear systems will be not so strong.

Chapter 3

Inverse iteration and related methods.

3.1 Introduction.

Assuming that $A \in \mathbb{R}^{n \times n}$ is real and symmetric positive definite (SPD), the spectrum of this matrix (denoted by $\sigma(A)$) has the following properties

$$\begin{aligned} Au_j &= \lambda_j u_j, \quad \|u_j\|_2 = 1, \quad \text{and} \\ 0 &\leq \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n. \end{aligned} \tag{3.1}$$

The problem of finding the eigenpairs (u_j, λ_j) by an iterative method is known as a symmetric eigenvalue problem. An other related problem, known as the generalized eigenvalue problem, is based in the search of the solution of

$$Au_j - \lambda_j Mu_j = 0, \quad \text{with } u_j \neq 0, \tag{3.2}$$

given two symmetric matrices A and M . If we multiply (3.2) by M^{-1} , the problem is reduced to the standard eigenvalue problem (3.1), where A is replaced by $M^{-1}A$ (provided M is invertible). If in addition, the matrix M is positive definite, we can compute the Cholesky decomposition $M = R^T R$, and the problem can be written as

$$\begin{aligned} R^{-T} A R^{-1} R u_j &= \lambda_j R u_j, \\ \Leftrightarrow \quad \tilde{A} \tilde{u}_j &= \lambda_j \tilde{u}_j. \end{aligned} \tag{3.3}$$

The eigenvalues of $\tilde{A}$ are the same as those of $M^{-1}A$, and the eigenvectors will be $\tilde{u}_j = Ru_j$.

In this Chapter, we will review some of the numerical methods related with the inverse iteration that are used to solve the symmetric eigenvalue problem. As these methods compute some part of the spectrum $\sigma(A)$, we say that they perform a partial spectral factorization of the matrix A (or $\tilde{A}$). We start in Section 3.2 by a short description of the single eigenvalue computation methods, directly related with the inverse iteration. In section 3.3, we review the main contributions to the study of the inexact inverse iteration. We continue in Section 3.4 with an overview of the Jacobi-Davidson method, which is a type of inexact inverse iteration method. Some topics on the subspace iteration, that is the generalization of the power (or inverse) iteration, are resumed in Section 3.5. We conclude in Section 3.6, with a complementary description of the BlockCGSI algorithm from the point of view of an inexact subspace iteration method.

3.2 Direct and inverse iteration methods

We recall from the books of Parlett (1998); Golub and van Loan (1996) and Stewart (2001) the main components of the direct and inverse iteration methods. The power method and the inverse iterations are simple vector iterations that consist in the successive multiplication of the iteration vector $x^{(k)}$ by the matrix A , in the first case, or by A^{-1} in the second one. In the case of the direct iteration (or power method), the resulting vector is more and more pointed in the direction of the eigenvector corresponding to the largest eigenvalue. In the case of inverse iteration, the resulting vector becomes close to the eigenvector corresponding to the smallest eigenvalue. This old idea has been perfected along the years, and has given rise to sophisticated and reliable methods that are still used in certain domains like for instance in structural engineering.

3.2.1 Rayleigh quotient

Once we have an approximated eigenvector $v \in \mathbb{R}^n$ we can find the corresponding approximated eigenvalue by the Rayleigh quotient

$$\delta(v) = \frac{v^T A v}{v^T v}, \quad (3.4)$$

because $\delta(v)$ minimizes the residual norm $\|Av - \alpha v\|_2$ over all the scalars α .

From a geometrical point of view, the minimization property of the Rayleigh quotient means that, for any $v \neq 0$, the spectral residual $S(v) = Av - v\delta(v)$ is orthogonal to v , i.e. $v^T S(v) = 0$. Other basic property of the Rayleigh quotient, called *Boundedness* (Parlett,

1998), says that $\delta(v)$ ranges over the interval $[\lambda_1, \lambda_n]$ as v ranges over all nonzero vectors in $\mathbb{R}^n$.

Given an approximated eigenvector v and an approximated eigenvalue δ (Rayleigh quotient for instance), we can have an estimation the quality of v and δ by knowing that there is an eigenvalue λ of A satisfying

$$|\lambda - \delta| \leq \frac{\|Av - \delta v\|_2}{\|v\|_2}. \quad (3.5)$$

3.2.2 Power iteration

If we take a vector $v^{(0)}$ and we multiply it by A^k , as $k \rightarrow \infty$, the resulting vector $v^{(k)} = A^k v^{(0)}$ will be close (after scaling) to the eigenvector (u_n) corresponding to the largest eigenvalue (λ_n). This observation is the basis of the power iteration method described in algorithm 3.2.2.

ALGORITHM 3.2.2: POWER ITERATION
$v^{(0)}$ = some vector with $\ v^{(0)}\ _2 = 1$ For $k = 1, 2, \dots$, until convergence Do : 1. $w = Av^{(k-1)}$ (apply A) 2. $v^{(k)} = w/\ w\ _2$ (normalize) 3. $\delta^{(k)} = (v^{(k)})^T Av^{(k)}$ (Rayleigh quotient) 4. test for convergence (invariance) EndDo

In Parlett (1998) it is proved that the power iteration converges linearly to λ_n with a convergence rate of λ_{n-1}/λ_n as $k \rightarrow \infty$. For that, λ_n must be the unique dominant eigenvalue of A and the starting vector $v^{(0)}$ must not be orthogonal to u_n , i.e. $u_n^T v^{(0)} \neq 0$. The upper bounds error associated with the iterate vector $v^{(k)}$ and the corresponding Rayleigh quotient $\delta^{(k)}$ that can be found in Golub and van Loan (1996) are

$$|\sin(\theta_k)| \leq \tan(\theta_0) \left(\frac{\lambda_{n-1}}{\lambda_n} \right)^k, \quad (3.6)$$

$$|\lambda_n - \delta^{(k)}| \leq |\lambda_1 - \lambda_n| \tan^2(\theta_0) \left(\frac{\lambda_{n-1}}{\lambda_n} \right)^{2k},$$

where θ_k is the angle between the iterate vector $v^{(k)}$ and the eigenvector u_n . These formulas show that the Rayleigh quotient $\delta^{(k)}$ converges twice as fast as the approximate eigenvector

$v^{(k)}$. This is of particular importance because when we monitor the convergence, in step 4 of Algorithm 3.2.2, the threshold in the stopping criterion must be adapted to the targeted output, for either $v^{(k)}$ or $\delta^{(k)}$. From (3.6), we can also conclude that rapid convergence can come in two ways: small ratio λ_{n-1}/λ_n or small θ_0 .

3.2.3 Inverse iteration

The inverse iteration is the power iteration involving A^{-1} instead of A in step 1 of algorithm 3.2.2. There is no need to invert A because the step 1 is replaced by the solution of the system $Aw = v^{(k-1)}$ for w . For $k \rightarrow \infty$, and assuming that λ_1 is the eigenvalue closest to 0 and $u_1^T v^{(0)} \neq 0$, the vector $v^{(k)}$ in the inverse iteration converges linearly to u_1 with a convergence rate of λ_1/λ_2 at worst.

If there is an approximation ν to an eigenvalue λ_j , we may apply the inverse iteration with $(A - \nu I)^{-1}$ instead of A^{-1} . The next iterate vector $v^{(k)}$ will have component in the direction of u_j multiplied by $1/(\lambda_j - \nu)$ and this component will be much larger when compared to the other eigencomponents. This idea is incorporated in the shifted version of inverse iteration presented in the algorithm 3.2.3.

ALGORITHM 3.2.3: SHIFTED INVERSE ITERATION	
$v^{(0)}$: some vector with $\ v^{(0)}\ _2 = 1$	
a shift value ν : an approximation of λ_j	
For $k = 1, 2, \dots$, until convergence Do :	
1. solve $(A - \nu I)w = v^{(k-1)}$ for w	(apply $(A - \nu I)^{-1}$)
2. $v^{(k)} = w/\ w\ _2$	(normalize)
3. $\delta^{(k)} = (v^{(k)})^T A v^{(k)}$	(Rayleigh quotient)
4. test for convergence	(invariance)
EndDo	

The rate of convergence of the shifted version of inverse iteration is proportional to $|\nu - \lambda_J|/|\nu - \lambda_K|$ where λ_J is the closest eigenvalue to ν and λ_K is the second closest. For this reason, the inverse iteration converges more rapidly when ν is well chosen, as for instance when one or more eigenvalues are already known and when we want to compute the corresponding eigenvectors. In this case, one step of inverse iteration can be sufficient for convergence.

In each inverse iteration (k), we have (in exact arithmetic) $(A - \nu I)w = v^{(k-1)}$ with $\|v^{(k-1)}\|_2 = 1$. The normalized spectral residual is then given by

$$\begin{aligned}
\|\hat{S}^{(k)}\|_2 &= \|(A - \nu I)w\|_2 / \|w\|_2 \\
&= \|v^{(k-1)}\|_2 / \|w\|_2 \\
&= 1 / \|w\|_2.
\end{aligned} \tag{3.7}$$

By (3.5), this means that ν differs from an exact eigenvalue of A by no more than $1/\|w\|_2$.

The principal cost in the inverse iteration comes from the solution of the system of linear equations $(A - \nu I)w = v^{(k-1)}$. If a triangular factorization $(A - \nu I) = LDL^T$ is possible at the beginning, the new iterate $v^{(k)}$ can be computed at each iteration with the cost of a matrix-vector multiplication. In this case, there is an additional initial cost in this factorization, but the solution phase at each iteration can be rather cheap depending on the sparsity of the factors L .

Additionally, if ν is close to some eigenvalue λ_j , the system can be very ill conditioned and roundoff errors can give rise to an erroneous solutions. In practice, this is not a problem because, as shown in Parlett (1998), the error resulting from the inaccurate solution of the system is almost entirely in the direction of u_j , and no significant damage is introduced in the convergence process.

3.2.4 Rayleigh quotient iteration

We have introduced in section 3.2.1 the Rayleigh quotient that gives an eigenvalue estimate δ associated with an eigenvector estimate v , which can be obtained, for instance, with the shifted inverse iteration. Combining these two ideas gives rise to an algorithm called the Rayleigh quotient iteration (Algorithm 3.2.4).

ALGORITHM 3.2.4: RAYLEIGH QUOTIENT ITERATION	
$v^{(0)}$ = some vector with $\ v^{(0)}\ _2 = 1$	
$\delta^{(0)} = v^{(0)T} A v^{(0)}$	
For $k = 1, 2, \dots$, until convergence Do :	
1. solve $(A - \delta^{(k-1)}I)w = v^{(k-1)}$ for w	(apply $(A - \delta^{(k-1)}I)^{-1}$)
2. $v^{(k)} = w / \ w\ _2$	(normalize)
3. $\delta^{(k)} = v^{(k)T} A v^{(k)}$	(Rayleigh quotient)
4. test for convergence	(invariance)
EndDo	

The Rayleigh quotient iteration improves iteratively the eigenvalue estimates to increase the rate of convergence. The convergence to an eigenpair (u_j, λ_j) almost always

occur, and when it does, the rate of convergence is cubic. This means that each iteration triples the number of correct digits of accuracy of $\delta^{(k)}$, and the angle θ_k between u_j and $v^{(k)}$, has the following asymptotic behavior

$$\lim_{k \rightarrow \infty} \left| \frac{\theta_{k+1}}{\theta_k^3} \right| \leq 1, \quad (3.8)$$

the inequality being sharp (Parlett, 1998).

Compared to the shifted inverse iteration, the Rayleigh quotient iteration presents a better convergence rate but also increases the amount of work required at each iteration. Because the coefficient matrix $(A - \delta^{(k-1)}I)$ changes, the solution of the system implies a new factorization at each step.

3.3 Inexact inverse iteration

In the inverse iteration and related methods, the system of linear equation is traditionally solved through the factorization of the coefficient matrix (step 1 of Algorithms 3.2.3 and 3.2.4). This can be expensive or impractical if n is large. 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 its multiplication with any vector. However, an iterative method also introduces an error, when computing the approximate solutions, that may affect the linear convergence rate in the inverse iteration (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 global computational work.

The inverse iteration combined with an iterative solver, also called inexact inverse iteration (see for instance Lai et al. (1997)), includes two levels of iterations. One is the outer iteration, and corresponds to the loop k of the Algorithm 3.2.3. The other is the inner iteration and corresponds to the iterative solution of the system in step 1 of Algorithm 3.2.3. 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.

- Ruhe and Wiberg (1972). In this early paper, an algorithm that combines the Conjugate Gradient with the inverse iteration is proposed. It focuses on two different ways to prevent a possible breakdown in the solution of the *nearly* singular and indefinite system $(A - \nu I)w = v^{(k-1)}$ with the CG algorithm. The first one is designed for the case of a fixed shift value ν , that is very close to the targeted eigenvalue. In this case, only one inverse iteration is sufficient, in which the CG algorithm performs a double

step at the beginning and continues as usual. The second proposes a correction to the Rayleigh quotient δ in order to get less problematic shifts.

The spectral residual is also reformulated as

$$\|\hat{S}^{(k)}\|_2 = \frac{\|v^{(k-1)} - r^{(k)}\|_2}{\|w\|_2}, \quad (3.9)$$

including the residual $r^{(k)} = v^{(k-1)} - (A - \nu I)w$ which comes from the inexact solution of the system $(A - \nu I)w = v^{(k-1)}$. They have additionally incorporated in the stopping criterion of the inner iteration a condition that prevents the inner residual to become too small. In practice they considered that after $\|r^{(k)}\|_2$ has reached a certain level, proportional to $1/\|w\|_2$, no further reduction of the norm of the spectral residual $\hat{S}^{(k)}$ can be expected.

- Lai et al. (1997). The authors propose the following inexact inverse iteration algorithm (Algorithm 3.3).

ALGORITHM 3.3: INEXACT INVERSE ITERATION (LAI ET AL. (1997) VARIANT)
$v^{(0)}$ = some vector with $\ v^{(0)}\ _2 = 1$ $\epsilon_0 \ll 1$ For $k = 1, 2, \dots$, until convergence Do : 1. iterate to solve $Aw = v^{(k-1)}$, until $\ Aw - v^{(k-1)}\ _2 \leq \epsilon_{k-1}$ 2. $\beta_k = \ w\ _2$ 3. $v^{(k)} = w/\beta_k$ 4. If $\ Av^{(k)} - v^{(k)}/\beta_k\ _2 < TOL$ Then stop 5. compute ϵ_k (see explanations below) EndDo

It is shown that if $v^{(0)}$ is not too close to be orthogonal to u_1 , and if ϵ_k is bounded by $1/|\prod_{i=1}^k \beta_i \lambda_2|$ for all k in Algorithm 3.3, then $v^{(k)}$ and $1/\beta_k$ converge linearly to u_1 and $1/\lambda_1$. The authors also give two practical ways to estimate the above bound for ϵ_k , in which λ_2 is absent:

$$\epsilon_k = \frac{1}{k} \left| \frac{\beta_k - \beta_{k-1}}{\beta_k} \right| \quad \text{or} \quad \epsilon_k = \frac{1}{k} \frac{\|v^{(k)} - v^{(k-1)}\|_2}{|\beta_k|}. \quad (3.10)$$

The proposed inexact inverse iteration (Algorithm 3.3) with the second option of the inner stopping threshold (3.10) is very efficient in reducing the total amount of inner iterations, without degrading the global convergence.

These results clearly indicate that if one wants to maintain the same convergence rate as in the exact inverse iteration, the stopping threshold ϵ_k must decrease proportionally to the number of outer iteration k . For instance, the linear systems can

be solved more or less roughly at the beginning, and more and more accurately as the number of outer iterations increases.

- Smit and Paardekooper (1999). Here, a geometric interpretation is used to prove the linear convergence of the inexact inverse iteration. The authors also propose an algorithm, similar to Algorithm 3.3, in which the shift ν is fixed. Assuming that the system in step 1 is solved approximately with a tolerance ϵ_k , the resulting residual norm is $\|(A - \nu I)w - v^{(k-1)}\|_2 \leq \epsilon_k$, or equivalently

$$(A - \nu I)w = v^{(k-1)} + e_v^{(k-1)} \quad \text{for some } e_v^{(k-1)} \quad \text{with } \|e_v^{(k-1)}\|_2 \leq \epsilon_k. \quad (3.11)$$

It is analyzed how the perturbation vector $e_v^{(k-1)}$ affects the reduction of the angle between $v^{(k-1)}$ and the corresponding eigenvector u_1 , which is given by, in the exact case,

$$\tan \theta_w = \frac{|\lambda_1 - \nu|}{|\lambda_2 - \nu|} \tan \theta_{v^{(k-1)}}. \quad (3.12)$$

They conclude that, in the inexact case, this reduction can be bounded by

$$\tan \theta_w \leq \gamma \tan \theta_{v^{(k-1)}} \quad \text{if } \forall k : \epsilon_k \leq \frac{(\gamma\eta - 2) \sin \theta_{v^{(k-1)}} \cos \theta_{v^{(k-1)}}}{\sqrt{\cos^2 \theta_{v^{(k-1)}} + \gamma^2 \eta^2 \sin^2 \theta_{v^{(k-1)}}}}, \quad (3.13)$$

where $\eta = |\lambda_2 - \nu|/|\lambda_1 - \nu|$ and $\eta^{-1} < \gamma < 1$. The formula (3.13) indicates that the upper bound on the tolerance threshold ϵ_k depends on the linear rate of convergence γ and on the angle between the right-hand side $v^{(k-1)}$ and the targeted eigenvector u_1 . As the upper bound on ϵ_k decreases with $\theta_{v^{(k-1)}}$, the authors propose a decreasing sequence of values for ϵ_k that ensures linear convergence. These values are computed by

$$\epsilon_k = \left(1 - \gamma^{-1} \frac{|\lambda_1 - \nu|}{|\lambda_2 - \nu|}\right) \sin \theta_{v^{(k-1)}} \cos \theta_{v^{(k-1)}}. \quad (3.14)$$

Based on (3.14), a practical strategy for the choice of ϵ_k is given:

$$\epsilon_k = \min \left\{ \frac{(1 - \gamma^{-1} q_k) q_k}{(1 + q_k) |\delta^{(k)} - \nu|} \|r^{(k)}\|_2, \epsilon_{\max} \right\}, \quad (3.15)$$

where $r^{(k)}$ is the current residual, $q_k = \|r^{(k)}\|_2 / \|r^{(k-1)}\|_2$ is the reduction factor that approaches the rate of convergence $|\lambda_1 - \nu|/|\lambda_2 - \nu|$, $\delta^{(k)}$ is the Rayleigh quotient associated with $v^{(k)}$. To ensure convergence even at the beginning, some maximum value for the tolerance $\epsilon_{\max}$ is also included.

The paper finishes with an analysis of the inexact Rayleigh quotient iteration, using the same approach as above in the case of the inexact inverse iteration. They conclude that the appropriate choice for ϵ_k is mostly critical at the beginning, and if it is small

enough to start the convergence process, then the inexact Rayleigh quotient iteration will converge at least quadratically.

- Golub and Ye (2000). The authors present a theoretical analysis of the inexact inverse iteration for the generalized eigenvalue problem $Au = \lambda Mu$. The study follows an inexact inverse iteration formulation similar to the Jacobi-Davidson method (see section 3.4). In each outer iteration, an approximate solution to $Aw^{(k)} = Mv^{(k-1)}$ is sought in which the inner residual $r^{(k)} = Aw^{(k)} - Mv^{(k-1)}$ satisfies a certain termination criterion. Assuming that $w^{(k-1)}$ is an approximation of the eigenvector u_1 , the problem is addressed in solving the *correction equation* $Ad^{(k)} = Mv^{(k-1)} - Aw^{(k-1)}$, before updating $w^{(k)} = w^{(k-1)} + d^{(k)}$.

Two versions, that differ from each other only in the choice of the stopping criterion in the inner iteration, are analyzed separately. The inner iteration in the first version is stopped when

$$\|r^{(k)}\|_2 \leq \epsilon \|Mv^{(k-1)} - Aw^{(k-1)}\|_2, \quad (3.16)$$

which corresponds to the standard relative residual measure (assuming that the iterative solver starts with an initial iterate $d = 0$), and in the second version when

$$\|r^{(k)}\|_2 \leq \epsilon_k \|w^{(k-1)}\|_2. \quad (3.17)$$

In (3.16), the stopping threshold ϵ is fixed, and in (3.17) the value of ϵ_k varies from one iteration to another iteration.

The proposed inexact inverse iteration method is summarized in Algorithm 3.3. We can observe that the outer residual $S^{(k-1)} = Mv^{(k-1)} - \beta_{k-1}Aw^{(k-1)}$, computed in step 1, is linked to the standard spectral residual (scaled by β_{k-1}) because at convergence $1/\beta_{k-1}$ approximates the smallest eigenvalue.

The authors develop a theoretical convergence analysis of Algorithm 3.3, in the case of the stopping criterion given by (3.17). They prove that, if we take $\epsilon_k \approx \gamma^k$ and

ALGORITHM 3.3: INEXACT INVERSE ITERATION (GOLUB AND YE (2000) VARIANT)

$v^{(0)}$ = some vector with $\ v^{(0)}\ _\infty = 1$ and $w^{(0)} = 0$

For $k = 1, 2, \dots$, until converge Do :
--

1. $S^{(k-1)} = Mv^{(k-1)} - Aw^{(k-1)}$
--

2. iterate to solve $Ad^{(k)} = S^{(k-1)}$ until $r^{(k)} = Ad^{(k)} - S^{(k-1)}$ satisfies (3.16) (or (3.17))
--

3. $w^{(k)} = w^{(k-1)} + d^{(k)}$

4. $\beta_k = \ w^{(k)}\ _\infty$

5. $v^{(k)} = w^{(k)} / \beta_k$

EndDo

$\rho \leq \gamma < 1$, with $\rho = \lambda_1/\lambda_2$, then $v^{(k)}$ will converge linearly to u_1 with a convergence rate directly given γ . Additionally, if $\gamma \leq \rho$, the convergence rate is always ρ , and if $\gamma \geq 1$, convergence may not be expected.

With respect to the computational work, they show experimentally that the total number of inner iterations is close to a minimum when γ is in the range $[\rho, 1[$, and increases slightly as γ varies from ρ to 1. Their conclusion is to simply take γ greater than ρ but of the same order of magnitude. They also observed that taking γ close to ρ tends to minimize the number of outer iterations.

The authors verify also experimentally the behavior of the algorithm with the stopping criterion (3.16). The numerical experiments tend to indicate that there is a rather wide range of values for ϵ for which the total number of inner iterations is minimum. However, unlike in (3.17), this range is not explicitly linked to ρ .

- **Simoncini and Eldén (2002).** The authors analyze efficient implementations of inexact Rayleigh quotient-type methods, which involve the approximate solution of a linear system at each iteration by means of the Conjugate Residual algorithm (Saad, 2000). Throughout the paper, it is supported that the reduction of the norm of the residual $r^{(k)} = v^{(k-1)} - (A - \delta^{(k-1)}I)w$, in the inner iteration, does only matter when the current approximation $v^{(k-1)}$ of the eigenvector is weak, and helps to reduce the norm of the spectral residual $\|S^{(k)}\|_2 = \|Av^{(k)} - \delta^{(k)}v^{(k)}\|_2$. On the opposite, when this approximation $v^{(k-1)}$ is good, improvements on the approximate eigenvector $v^{(k)}$ can be obtained long before the residual $r^{(k)}$ has been reduced substantially. In this case, it is the growth of the norm of the approximate solution vector w , along the inner iterations, that matters in this reduction of the spectral residual norm.

The authors begin by showing that the reduction of the angle θ_k between the iterate $v^{(k)}$ and the corresponding eigenvector u_j is strongly related to the reduction of the eigencomponent in the residual $r^{(k)}$ relative to that eigenvector. When the right-hand side $v^{(k-1)}$ of the linear system has a large component along that eigenvector, the iterative method will reduce significantly the residual along this direction already in the first inner iterations. After that, the authors highlight the dependence between the spectral residual $S^{(k)}$ and the growth of the norm of the iterate w approximating the solution of the linear system, and in particular that:

$$\|Av^{(k)} - \delta^{(k)}v^{(k)}\|_2 \leq \frac{\sqrt{1 - \|r^{(k)}\|_2}}{\|w\|_2}. \quad (3.18)$$

It is proved that the Rayleigh quotient iteration converges linearly as soon as $\|w\|_2 \geq \|Av^{(k-1)} - \delta^{(k-1)}v^{(k-1)}\|_2^{-1}$ and that, if a good approximation $v^{(k-1)}$ to u_j is available, the growth of $\|w\|_2$ is directly linked to the variation of $S^{(k)}$ in the direction of the wanted eigenvector u_j .

Based on the theoretical developments, the authors propose a stopping criterion for the inner iteration that is based on the variation of the magnitude of $\|w\|_2$:

$$\text{if } \frac{\|w^{[i]}\|_2 - \|w^{[i-1]}\|_2}{\|w^{[i]}\|_2} \leq \epsilon_{inner} \ \& \ \|w^{[i]}\|_2 > \frac{1}{\|Av^{(k-1)} - \delta^{(k-1)}v^{(k-1)}\|_2} \text{ then stop,} \quad (3.19)$$

where the superscript $[i]$ denotes the inner iteration i . The additional condition on the right of (3.19) is included in order to ensure that a smaller spectral residual norm is always obtained, and to prevent the inner solver from stopping too early. This also helps to avoid useless outer iterations. It is also evidenced that other stopping criteria like for instance (3.16) and (3.17) proposed in Golub and Ye (2000), are related to monitoring the quantities $\|w\|_2$.

The stopping criterion (3.19) is combined with

$$\frac{\|Av^{(k)} - \delta^{(k)}v^{(k)}\|_2}{\delta^{(k)}} \leq \epsilon_{outer}, \quad (3.20)$$

for the outer iteration, in order to detect when the approximate eigenpair $(v^{(k)}, \delta^{(k)})$ has converged within a given tolerance ϵ_{outer} . The numerical experiments show that (3.19) is not very sensitive to this tolerance ϵ_{inner} as long as the tolerance is not too loose, and that a lot of computational effort can be wasted when this tolerance is too tight. The only case where it can still be advantageous to consider very small values for this tolerance is when the wanted eigenvalue λ_j is well separated from the next one λ_{j+1} .

The authors also propose a preconditioning technique suitable for the solution of the indefinite system involved in the Rayleigh quotient iteration (step 1 of Algorithm 3.2.4). They first show that a standard symmetric preconditioning based on the incomplete factorization $M = R^T R$ of A (for instance $M = IC(A)$), viz.

$$R^{-T}(A - \delta I)R^{-1}\hat{w} = R^{-T}v \quad \text{and} \quad w = R^{-1}\hat{w}, \quad (3.21)$$

yields a very slow convergence in the outer iteration. That is so because the right-hand side $R^{-T}v$ is no longer an approximate eigenvector of the coefficient matrix $R^{-T}(A - \delta I)R^{-1}$. To overtake this difficulty, they first show that, if (v, δ) is close (with respect to some specific conditions) to an eigenpair of A , then Rv approximates the eigenvector of $R^{-T}(A - \delta I)R^{-1}$ associated with the eigenvalue closest to zero in this preconditioned matrix. Consequently, they propose to solve instead the following preconditioned system:

$$R^{-T}(A - \delta I)R^{-1}\hat{w} = Rv \quad \text{and} \quad w = R^{-1}\hat{w}. \quad (3.22)$$

This maintains the nice convergence properties of the Rayleigh quotient iteration. The numerical experiments show that this preconditioning technique (3.22) is very

effective to reduce to total amount of work compared with the standard preconditioning (3.21).

- Berns-Mueller et al. (2005). This paper gives a general convergence theory of the inexact inverse iteration, that is independent of the choice of the inexact solver and of the shift. It is proved that the inexact Rayleigh quotient iteration can converge cubically if the linear system is solved with a decreasing tolerance, and quadratically with a fixed tolerance. The main theoretical result also shows that, with a fixed shift value strategy, convergence occurs only with decreasing tolerance values and that convergence cannot be expected with a fixed tolerance.

The convergence of the preconditioned inner iteration, as proposed in (3.22), is also analyzed from the theoretical point of view. With a sufficiently good starting guess, the preconditioned Rayleigh quotient iteration is quadratically convergent. The preconditioning (3.22) breaks the property of cubic convergence (except in a very special case), but is very effective in reducing the total number of inner iterations.

An important study of the Krylov method MINRES, used as inner solver in the inexact inverse iteration, is also presented. MINRES is a good choice to solve $A - \delta^{(k-1)}I = v^{(k-1)}$, because as the inverse iteration proceeds, the symmetric matrix $A - \delta^{(k-1)}I$ becomes more and more indefinite. MINRES also has the important property of minimizing the residual norm $\|v^{(k-1)} - (A - \delta^{(k-1)}I)\|_2$. An analysis of the number of iterations needed for the unpreconditioned and preconditioned inner solves is also provided, and descriptive bounds on the number of inner iterations needed to reach linear convergence are derived. They indicate that a quadratic convergence of $\delta^{(k-1)}$ to λ_1 induces only a logarithmic growth in the number of inner iterations required to achieve such convergence. The *a priori* analysis indicates that the preconditioning (3.22) need less inner MINRES iterations per outer iteration, as opposed to the case with the standard preconditioning (3.21). The *a posteriori* bounds confirm the superiority of (3.22) in terms of overall costs to achieve a given accuracy.

A numerical study, comparing a stopping criterion based on the standard relative residual with the one proposed by Simoncini and Eldén (3.19), is also given. If the inner iteration is preconditioned as shown in (3.22), the standard relative residual stopping condition is given by

$$\text{if } \frac{\|Mv^{(k-1)} - (A - \delta^{(k-1)}I)\|_2}{\|Mv^{(k-1)}\|_2} \leq \epsilon_{inner} \text{ then stop.} \quad (3.23)$$

Combined with the outer stopping criterion (3.20) (the same one as used in Simoncini and Eldén (2002)), the performances observed with the use of (3.19) appear to very sensitive to the choice of the fixed tolerance ϵ_{inner} . If ϵ_{inner} is chosen too large, then (3.19) may fail to provide convergence for an interior eigenvalue. Therefore, it is very difficult in practice to find a tolerance ϵ_{inner} that ensures convergence. In

contrast, the stopping criterion (3.23) seems to provide rather stable performances for a large range of values of ϵ_{inner} .

3.4 Jacobi-Davidson method

The Jacobi-Davidson method (see reference Sleijpen and van der Vorst (1996)) is an inexact inverse iteration type method, that has been developed in order to circumvent the ill-conditioning of the shifted systems that must be solved at each inverse iteration. Given an approximate eigenvector v of norm unity and associated residual $S = Av - \delta v$, where $\delta = v^T Av$ is the Rayleigh quotient, the Jacobi-Davidson method basically computes a correction d to the current approximation v by solving approximately the *correction equation*

$$(I - vv^T)(A - \nu I)(I - vv^T)d = -S, \quad d \perp v \quad (3.24)$$

where ν either equals some fixed target value τ ($\tau = 0$ if A is positive definite) or is given by the Rayleigh quotient δ . Thanks to the projection onto the space orthogonal to v , the system (3.24) is expected to remain in general reasonably well conditioned, even when $A - \delta I$ becomes closer to a singular matrix as δ converges to an eigenvalue. If equation (3.24) is solved exactly, the method converges as fast as the inverse or Rayleigh quotient iteration, depending to the choice of ν . However, a modest accuracy in these solutions seems to be sufficient in general to ensure convergence. A simplified version of the Jacobi-Davidson procedure is given in Algorithm 3.4.

ALGORITHM 3.4: JACOBI-DAVIDSON METHOD (SIMPLIFIED VERSION)

$v^{(0)}$ = some vector with $\|v^{(0)}\|_2 = 1$

$\delta^{(0)} = v^{(0)T} A v^{(0)}$

For $k = 1, 2, \dots$, **until convergence** **Do**:

1. $S^{(k)} = Av^{(k-1)} - \delta^{(k-1)}v^{(k-1)}$

if $\|S^{(k)}\|_2 < \epsilon$ exit

2. select $\nu = \tau$ or $\nu = \delta^{(k-1)}$ and solve

$(I - v^{(k-1)}v^{(k-1)T})(A - \nu I)(I - v^{(k-1)}v^{(k-1)T})d^{(k)} = -S^{(k)}$ for $d \perp v^{(k-1)}$

3. $v^{(k)} = \frac{v^{(k-1)} + d^{(k)}}{\|v^{(k-1)} + d^{(k)}\|_2}$

4. $\delta^{(k)} = v^{(k)T} A v^{(k)}$

EndDo

In the paper written by Notay (2002), the convergence of the Jacobi-Davidson method combined with the CG algorithm, as the inner solver, is analyzed in the context of the

computation of the smallest eigenvalue of a SPD matrix. 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. Assuming that the starting guess $d_0^{(k)}$ is set to zero, and that i inner iterations are performed, the spectral residual at the next inverse iteration will be given by

$$\|S_i^{(k+1)}\|_2^2 = \frac{\|r_i^{(k)}\|_2^2}{1 + \|d_i^{(k)}\|_2^2} + \left(\frac{\|d_i^{(k)}\|_2^2}{1 + \|d_i^{(k)}\|_2^2} (\delta^{(k)} - \nu + \beta^{(k)}) \right)^2, \quad (3.25)$$

where $\beta^{(k)} = (v^{(k)}, (A - \nu I)d_i^{(k)})$, $d_i^{(k)}$ is the approximate solution of the correction equation 3.24 obtained after i CG inner iterations, and $r_i^{(k)}$ the corresponding residual. It is also proved that the second term in (3.25) converges quickly to $\|S^{(k+1)}\|_2^2$, that is the value that one would obtain if the correction equation (3.24) was exactly solved. This result show that the outer process can be monitored at the same time as convergence of the inner loop is evaluated. As the first term in (3.25) vanishes along with the residual of the linear system, it is proposed to stop the inner iteration when the first term does not longer dominate the second one. The following inner stopping criterion is proposed

$$\text{if } \frac{\|S_i^{(k+1)}\|_2}{\|S_{i-1}^{(k+1)}\|_2} > \left(\frac{\|r_i^{(k+1)}\|_2}{\|r_{i-1}^{(k+1)}\|_2} \right)^a \text{ then stop,} \quad (3.26)$$

for some $a < 1$ like for instance $a = 0.9$. The reason for this is that the two ratios in (3.26) behave similarly until the point where $\|S_i^{(k+1)}\|_2$ stagnates, because it is close to its minimal value. In order to prevent a too early exit, they also add the safeguard condition

$$\frac{\|r_i^{(k)}\|_2}{\|r_0^{(k)}\|_2} \leq 0.5 \quad (3.27)$$

to ensure a minimal reduction of the inner residual. The experiments indicates that it is preferable to stop the inner iteration as soon as both the safeguard condition (3.26) and

$$\|S_i^{(k+1)}\|_2 \geq \|S_{i-1}^{(k+1)}\|_2 \quad (3.28)$$

hold, keeping then $d_{i-1}^{(k+1)}$ as approximate solution of the correction equation. This is explained by the convergence of the Conjugate Gradient algorithm when the eigenvalue distribution favor an irregular behavior. The convergence may be faster in the initial phase than in the subsequent one where quite large plateaus are possible.

3.5 Subspace iteration

Subspace iteration is a generalization of both the power method and inverse iteration which were presented in section 3.2. Consider a symmetric and positive definite matrix A . If we multiply a matrix V with size $n \times s$, successively by A^{-1} and orthonormalize the column vectors after each multiplication, the column vectors from the resulting matrix Q will converge to the invariant subspace of dimension s , associated with the smallest eigenvalues in the range $]0, \lambda_s]$. This simplest version of subspace iteration is also called orthogonal iteration. The convergence rate of the orthogonal iteration, comparable to the one of the inverse iteration, is of order λ_s/λ_{s+1} (see for instance Golub and van Loan (1996)).

In order to get better approximation to each individual eigenvector, using all the information in the base Q , the orthogonal iteration is normally followed by the Ritz (or Rayleigh-Ritz) acceleration. In Algorithm 3.5, we present the main steps involved in the (inverse) subspace iteration.

ALGORITHM 3.5: SUBSPACE ITERATION ALGORITHM

Inputs: $A \in \mathbb{R}^{n \times n}$, $Z^{(0)} \in \mathbb{R}^{n \times s}$, $s \in \mathbb{N}$
Output: an invariant subspace V
 with dimension s

Begin
 Generate the initial subspace
 $Z^{(0)}$ an initial basis of dimension s
 $V^{(0)}\Gamma = Z^{(0)}$ such that $V^{(0)T}V^{(0)} = I_{s \times s}$
For $k = 1, 2, \dots$ **until convergence** **Do:**
 Orthogonal iteration
 Solve $AZ^{(k)} = V^{(k-1)}$
 $Q^{(k)}\Gamma_k = Z^{(k)}$ such that $Q^{(k)T}Q^{(k)} = I_{s \times s}$
 Ritz acceleration
 $\beta_k = Q^{(k)T}AQ^{(k)}$
 Diagonalize $\beta_k = U_k\Delta_kU_k^T$
 where $U_k^T = U_k^{-1}$
 and $\Delta_k = \text{Diag}(\delta_1, \dots, \delta_s)$ (Ritz Values)
 $V^{(k)} = Q^{(k)}U_k$ (Ritz Vectors)
EndDo
End

The Ritz values $\text{diag}(\Delta) = \delta_1, \dots, \delta_s$ and Ritz vectors $V = [v_1, v_2, \dots, v_s]$ are approximations to the eigenpairs in A corresponding to the eigenvalues in the range $]0, \lambda_s]$. Provided

the Ritz acceleration is incorporated in the subspace iteration, the convergence rate is

$$|\lambda_j - \delta_j^{(k)}| \sim \left(\frac{\lambda_j}{\lambda_{s+1}} \right)^k, \quad (3.29)$$

where λ_{s+1} is the eigenvalue immediately following the current set of s approximated eigenvalues (see Parlett (1998) or Stewart (2001) for a complete analysis). This property shows that if λ_j is well separated from λ_{s+1} , we can have a good estimation of the eigenvalue λ_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 in the relation above. 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 (3.29).

In the subspace iteration the Ritz values δ_j converge faster to their limit λ_j than the corresponding Ritz vectors v_j to u_j . This means that one can have a converged Ritz value even if the corresponding Ritz vector is far from the wanted eigenvector. The angle between the two vectors is given by

$$|\sin \angle(v_j, u_j)| \leq \frac{\|Av_j - \delta_j v_j\|_2}{gap}, \quad (3.30)$$

where $gap = \min\{|\lambda_{j-1} - \lambda_j|, |\lambda_j - \lambda_{j+1}|\}$. In practice we cannot use (3.30) to monitor the convergence because the gap is unknown, but when the Ritz values have converged, we can use the δ_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 (3.30) will not be reliable (see Parlett (1998)).

Like in the inverse iteration, it is necessary to solve the linear systems $(A - \nu I)z = v$, either by factorizing $A - \nu I$ or with an iterative solver, to obtain in the end the eigenvalues closest to the shift value ν .

An acceleration technique that can be incorporated in the subspace iteration is the refined Ritz vectors technique proposed by Jia (2000). This technique looks for an optimal vector $\hat{v}_j$ that approaches the eigenvector u_j in:

$$\begin{aligned} &\text{minimizing } \|A\hat{v}_j - \delta_j \hat{v}_j\|_2 \\ &\text{subject to } \hat{v}_j \in \text{span}(V), \end{aligned} \quad (3.31)$$

where δ_j is the corresponding Ritz value and V is the matrix made of the Ritz vectors. The solution of problem (3.31) implies the solution of s small eigenproblems at low cost. Another improving technique that can be used with the subspace iteration is to exploit

Chebyshev polynomials to replace some consecutive steps of the power method. For instance in the computation of the subspace associated with the dominant eigenvalues, the computation of $A^k V$ is replaced by $p_k(A)V$, where p_k is a scaled Chebyshev polynomial of degree k chosen to enhance the gap between the two extremal eigenvalues λ_n and λ_{n-s} (see details in Parlett (1998) and Stewart (2001)).

Compared with other reliable Lanczos algorithms (see Saad (1992) for instance), 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 (3.29), 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 power (inverse) iterations will be needed for convergence.

3.6 Inexact subspace iteration

An inexact subspace iteration algorithm is possible with a block iterative method that solves the system with s right-hand sides $AZ = V$. Since A is symmetric and positive definite, a block Conjugate Gradient method can be suitable. However, in the case of shifted systems $(A - \nu I)Z = V$ a block MINRES or SYMMLQ may be preferable. As we have seen in the previous chapter, we are concerned with an algorithm called BlockCGSI that combines the subspace iteration with a stabilized version of the block Conjugate Gradient. In this section we review some properties of this algorithm, beginning by a short description of the stabilized block Conjugate Gradient.

3.6.1 Stabilized block Conjugate Gradient algorithm

The block Conjugate Gradient (blockCG) can be derived in the same way as the non-block analogue CG (see for instance Golub and van Loan (1996)). The only requirement is to replace, in all the discussion, the different n -vectors by $n \times s$ rectangular matrices, where the integer s corresponds to the block size. This is the case of the generalization of the Conjugate Gradient method described in Hageman and Young (1981). The blockCG algorithm is well defined as long the residual matrices $R^{(i)}$ and conjugate direction matrices $P^{(i)}$ retain full rank. As the $R^{(i)}$ matrices, which correspond to the residuals $B - AX^{(i)}$, go to zero at convergence, failure or ill conditioning might occur, specially if one residual vector converges faster than the others. Effectively, it can be derived that if a rank deficiency in the block Krylov subspaces

$$\mathcal{K}_i(A, R^{(0)}) = \text{Span}\{R^{(0)}, AR^{(0)}, \dots, A^{i-1}R^{(0)}\} \quad (3.32)$$

occurs at some stage i , then there is one or more of the s linear systems which have converged, and whose corresponding column vector in the $R^{(i)}$ and $P^{(i)}$ matrices will be the zero vector. In this case, the iteration can be continued after discarding the corresponding column vectors in the $R^{(i)}$, $P^{(i)}$ and $X^{(i)}$ matrices.

In the blockCG version proposed by O'Leary (1980), it is suggested to orthonormalize the columns of $P^{(i)}$ and to reduce the block size s when linear dependence is detected, in order to control the rank deficiency. This version, also called monitored blockCG (see Ruiz (1992)), shows a certain numerical instability characterized by a rupture in the convergence just at the moment when the superlinear convergence is reached. This occurs, despite the orthonormalization of the $P^{(i)}$ matrices, because the matrices $R^{(i)}$ become very ill conditioned when convergence is about to be reached. This strongly affects the global behavior of the algorithm.

In order to overtake the instabilities of the blockCG, a stabilized version was proposed in Ruiz (1992). It is suggested to follow the blockCG philosophy, which enables easy updates of the iterates $X^{(i)}$ to be performed, associated with the stability of the block Lanczos algorithm (see Underwood (1975)). To do that, the residual matrices $R^{(i)}$ are reorthonormalized into some $\bar{R}^{(i)}$ at each iteration. In order to minimize the amount of work, an A -orthonormalization of the $P^{(i)}$ matrices is also performed (instead of the QR factorization suggested in the monitored blockCG algorithm). This simplifies the computation of the ζ_i and α_i auxiliary matrices with size $s \times s$, and results in Algorithm 3.6.1, proposed by Ruiz (1992).

As we can observe in Algorithm 3.6.1, the $\bar{R}$ do not correspond to the residuals, and the computation of $X^{(i+1)}$ involves the residuals $R^{(i)}$. This implies an overhead since the computation of the residuals $R^{(i)}$ requires an additional call to the matrix product $AX^{(i)}$. However, it is possible to verify that

$$\beta_i^{-T} = \bar{P}^{(i)T} \bar{R}^{(i)}, \quad (3.33)$$

which enables to write

$$\bar{R}^{(i+1)} \gamma_{i+1} = \bar{R}^{(i)} - A \bar{P}^{(i)} \bar{P}^{(i)T} \bar{R}^{(i)}. \quad (3.34)$$

Using the fact that the residuals are also given by

$$R^{(i+1)} = R^{(i)} - A \bar{P}^{(i)} \bar{P}^{(i)T} R^{(i)} \quad (3.35)$$

combined with (3.34), it can be shown by recurrence, that the residuals $R^{(j)}$ are linked to the $\bar{R}^{(j)}$ matrices by the reverse product of the γ_j matrices, i.e.

$$R^{(i)} = \bar{R}^{(i)} \left(\prod_{j=i}^0 \gamma_j \right). \quad (3.36)$$

ALGORITHM 3.6.1: STABILIZED BLOCK CONJUGATE GRADIENT**Inputs:** $A \in \mathbb{R}^{n \times n}$, $B^{(0)} \in \mathbb{R}^{n \times s}$, $X^{(0)}$ (arbitrary) $\in \mathbb{R}^{n \times s}$ **Output:** a block solution vector X **Begin**

$$\bar{R}^{(0)} = B - AX^{(0)}$$

$$\bar{R}^{(0)} = R^{(0)} \gamma_0^{-1} \quad \text{such that} \quad \bar{R}^{(0)T} \bar{R}^{(0)} = I$$

$$\bar{P}^{(0)} = \bar{R}^{(0)} \beta_0^{-1} \quad \text{such that} \quad \bar{P}^{(0)T} A \bar{P}^{(0)} = I$$

For $i = 0, 1, 2, \dots$ **until converge** **Do:**

$$\zeta_i = \beta_i^{-T}$$

$$\bar{R}^{(i+1)} = (\bar{R}^{(i)} - A \bar{P}^{(i)} \zeta_i) \gamma_{i+1}^{-1} \quad \text{such that} \quad \bar{R}^{(i+1)T} \bar{R}^{(i+1)} = I$$

$$\alpha_i = \beta_i \gamma_{i+1}^T$$

$$\bar{P}^{(i+1)} = (\bar{R}^{(i+1)} + A \bar{P}^{(i)} \alpha_i) \beta_{i+1}^{-1} \quad \text{such that} \quad \bar{P}^{(i+1)T} A \bar{P}^{(i+1)} = I$$

$$X^{(i+1)} = X^{(i)} + \bar{P}^{(i)} \bar{P}^{(i)T} R^{(i)} \quad \text{where} \quad R^{(i)} = B - AX^{(i)}$$

EndDo**End**

The update of $X^{(i+1)}$ becomes

$$X^{(i+1)} = X^{(i)} + \bar{P}^{(i)} \bar{P}^{(i)T} \bar{R}^{(i)} \left(\prod_{j=i}^0 \gamma_j \right), \quad (3.37)$$

which is equivalent to the following formula

$$X^{(i+1)} = X^{(i)} + \bar{P}^{(i)} \zeta_i \left(\prod_{j=i}^0 \gamma_j \right). \quad (3.38)$$

This trick, which saves computation and storage resources is included in the blockCG version proposed in Arioli et al. (1995), and is the one that is used in this work. In this version, the ill-conditioning of the residuals is included in the reverse product of the transformation matrices γ_i . The matrix γ_i goes to zero when convergence is reached, and the iterates $X^{(i+1)}$ are not updated any more. The success key of the stabilized blockCG is the quality of the Krylov subspace basis which is not affected by the ill conditioning of the residuals. In each iteration, the basis is updated using the matrices γ_i without the need of the residuals $R^{(i)}$. The matrices γ_i are less ill conditioned than the residuals themselves because the ill conditioning of the residuals is spread out in the reverse product of all γ_i matrices. Therefore, the building of the basis of the Krylov subspace, which never involves the matrix $\prod_{j=i}^0 \gamma_j$, should be significantly improved when compared with other version of the blockCG algorithm.

If one is interested in the solution of one system only, the update of $X^{(i+1)}$ can even be simplified to

$$x_1^{(i+1)} = x_1^{(i)} + \bar{P}^{(i)} \ell_i \left(\prod_{j=i}^0 \gamma_j(1, 1) \right), \quad (3.39)$$

where ℓ_i is the first column of the matrix ζ_i . In some cases, there is some advantage in using the blockCG instead of the classical Conjugate Gradient for the solution of a unique linear system. Indeed, when clusters of eigenvalues are located at the extremes of the spectrum of the iteration matrix, the CG algorithm may present a convergence with sequences of plateaus. This misbehavior in presence of close eigenvalues has already been observed and analyzed in details in Ruiz (1992) and Arioli et al. (1995). The blockCG algorithm, as opposed to the CG algorithm, can identify multiple eigenvalues and therefore can be expected to perform better in the presence of clusters of eigenvalues. In the comparative study between the two methods presented in Ruiz (1992), it is observed that when the CG algorithm has difficulties to reach a superlinear convergence, the blockCG algorithm improves the situation since it is much less sensitive to the clusterization of the spectrum. But when the CG behaves as a powerful iterative method, the blockCG exhibits the drawback of a direct method: it gives a very accurate solution after computing global information from the linear system and is thus less efficient.

The efficiency of the blockCG depends strongly on the choice of the block size s . The A -norm of each error vector $e_j^{(i)} = x_j^* - x_j^{(i)}$, where x_j^* is the exact solution of the j^{th} system and $j = 1, \dots, s$, can be bounded (see Underwood (1975) and O'Leary (1980)) in terms of the reduced condition number $\kappa_s = \lambda_n / \lambda_s$, as

$$\|e_j^{(i)}\|_A \leq C_1 \|e_j^{(0)}\|_A \left(\frac{\sqrt{\kappa_s} - 1}{\sqrt{\kappa_s} + 1} \right)^i, \quad (3.40)$$

and not in terms of the classical condition number $\kappa = \lambda_n / \lambda_1$ as for the Conjugate Gradient algorithm (see for instance Golub and van Loan (1996)). This can considerably improve the convergence specially for matrices with a strong clustering of eigenvalues near zero. This indicates also that there is a certain optimal choice depending on the eigenvalue distribution, and that increasing the block size after a certain stage will not really improve the situation any further, but may rather increase the amount of work. The choice of the block size *a priori* is not easy because we have no information about the distribution of eigenvalues. Still, all the operations in the blockCG algorithm can involve Level 3 BLAS type of operations (see Dongarra et al. (1990)), and this can considerably improve the performance of Algorithm 3.6.1. Consequently, the appropriate block size s may not be the one realizing the minimum amount of work, but is the one realizing the best compromise between the amount of work and the efficiency of Level 3 BLAS kernels. Numerical experiments about this point, in the case of shared memory multiprocessor computers, can be found in Arioli et al. (1995).

3.6.2 Block Conjugate Gradient coupled with Subspace Iteration

An algorithm that combines subspace iteration (Algorithm 3.5) with the stabilized blockCG (Algorithm 3.6.1) was proposed in Arioli and Ruiz (1995). As we have seen in Chapter 2, this combination is called BlockCGSI algorithm and is described in Algorithm 2.3.1. The blockCG is used to solve simultaneously, at each inverse iteration, the system with s right and sides. The BlockCGSI algorithm approaches a *near*-invariant subspace associated to the eigenvalues in the range $[\lambda_1, \lambda_s]$, that is used in a second phase to improve the consecutive solution of linear systems with the same coefficient matrix and changing right and sides. In Arioli and Ruiz (1995), an important set of numerical experiments is exposed in order to understand how the computation of the spectral information, with the BlockCGSI algorithm, affects the convergence in the CG accelerated with the deflation of the starting residual, as described in Section 4.6.1. It is pointed out the importance of some working parameters like the block size s , the inner stopping threshold ϵ , and the number of inverse iteration m .

In the BlockCGSI algorithm proposed in Arioli and Ruiz (1995), as well as in our preliminary work presented in Chapter 2, the block size is fixed *a priori*. This presents some difficulties because the eigenvalue distribution is unknown and, as we have seen in sections 3.5 and 3.6.1, the subspace iteration and blockCG convergence rate depend strongly on this choice. From a theoretical point of view, we may always expect an improvement of the convergence as the block size increases. In practice, increasing the block size after a certain stage will not really improve the situation, but may rather increase the amount of work. A good choice for the block size is to consider the size of the extreme cluster of eigenvalues responsible for the ill conditioning, but unfortunately nothing is known about it *a priori*. Additionally, as mentioned in Section 3.6.1, the operations in the blockCG can involve Level 3 BLAS type of operations, and this can considerably improve the overall performance of the BlockCGSI algorithm. This opens the door to a strategy based on choosing the block size in agreement with the computer characteristics. However, the number of approximated eigenvectors, determined by this choice of the block size, may be insufficient to improve correctly the classical Conjugate Gradient in the second phase. This fact has been illustrated by the experimental results of Section 2.4.2. In the next Chapter, we propose an algorithmic feature that enables to overtake this limitation of the BlockCGSI algorithm. With this modification the dimension of the computed *near*-invariant subspace is independent of the block size s .

In the version of the BlockCGSI algorithm proposed in Arioli and Ruiz (1995), the inner stopping threshold ϵ , and the number of inverse iteration m are independent from each other. The convergence of the blockCG is monitored by measuring, at each inner iteration (i), the normwise backward error

$$\omega_{\max}^{(i)} = \max_{1 \leq j \leq s} \frac{\|Ax_j^{(i)} - b_j\|_{\infty}}{\|A\|_{\infty}\|x_j^{(i)}\|_1 + \|b_j\|_{\infty}}, \quad (3.41)$$

where x_j and b_j , $j = 1, \dots, s$, represent the j^{th} column of the block matrices $X^{(i)}$ and B respectively in Algorithm 3.6.1. The blockCG is stopped when the value of $\omega_{\max}^{(i)}$ is less than some threshold value ϵ . The accuracy of the approximated eigenvectors (Ritz vectors) is not measured. Instead of that, the number of inverse iterations is set as an input of the algorithm. The choice for these parameters constitute another limitation of the initial version of the BlockCGSI algorithm. In Chapter 2, we have tried to understand how the inaccurate solution of the system affects the convergence of the Ritz vectors to the targeted eigenvectors. To do that, we have proposed to monitor the blockCG with the backward error measure (2.8). We choose this because it is compatible with the measure of the eigenvalue error bound (2.9), used to monitor the convergence of the Ritz values $\delta_j^{(k)}$ and vectors $v^{(k)}$. We verify experimentally that the upper bound on the spectral error $|\lambda_j - \delta_j^{(k)}|$, given by (2.9), stagnates on a level very close to the stopping threshold ϵ used to stop the blockCG in each inverse iteration. Note that this observation is in agreement with the theoretical analysis developed in Golub and Ye (2000), for the fixed tolerance in the stopping criterion (3.16). In the next Chapter, we proceed to an inner-outer analysis of the BlockCGSI algorithm and derive some important properties about the evolution of the spectral residual (4.2) along the blockCG run. Based on these properties, we propose a stopping criterion for the inner iteration that guaranties that the algorithm converges to the targeted eigenspace and reduces the total number of inner iterations to the minimum necessary. With these considerations incorporated in the BlockCGSI algorithm, the threshold ϵ and the number of inverse iterations m disappear from the inputs parameters. Instead of them, will only be supplied one parameter giving the accuracy with which the *near*-invariant subspace should be computed.

Chapter 4

Monitoring and improving the BlockCGSI algorithm

4.1 Introduction

As we have seen in Chapter 1, we are concerned with the computation of a *near*-invariant subspace associated with the smallest eigenvalues in the iteration matrix $M^{-1}A$, and to do that we propose an algorithm, called BlockCGSI, based on the inexact subspace iteration (see section 3.6). In this algorithm, the set of multiple solutions required in each inverse iteration is computed iteratively using a stabilized version of the block Conjugate Gradient algorithm, that we call blockCG (see section 3.6.1). The implicit use of the inverse of the coefficient matrix by means of an iterative solution (inner iteration) introduces an error when computing the approximate solutions that may affect the convergence rate of the inverse iteration (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.

In section 3.3, we have seen that the problem resulting from the use of an inexact solver has been studied in several papers, some of which have appeared recently. In Smit and Paardekooper (1999); Lai et al. (1997), for instance, it is proved that the inexact inverse iteration can converge linearly at the same rate as the exact case even if the system is not solved accurately, and some practical ways to choose the inner stopping threshold are given. More recently, in Golub and Ye (2000), a general convergence analysis of the correlation between the convergence rate and the threshold parameter is shown. Considering some specific error measure, they proved 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 Simoncini and Eldén (2002), 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 of

the spectral 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 the approximate solution. Following this recommendation, Berns-Mueller et al. (2005) highlight 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 Notay (2002) (see section 3.4). 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 this Chapter, we analyze, from a geometrical point of view, the convergence of the subspace iteration combined with the blockCG inner solver, and we derive an expression that relates the two residual norms used in the two different iteration levels. From this, we propose some residual measure for the blockCG that is directly linked with the convergence of the outer process towards the desired eigenvectors. We also propose a stopping threshold parameter that minimizes the total amount of computational work, and enables to monitor the algorithm in function of the targeted accuracy.

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 that enables the computation of a *near*-invariant subspace of any dimension.

In section 4.2, we present the new version of the BlockCGSI algorithm. Section 4.3 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. In section 4.4, we describe the Chebyshev polynomial filters and the “*sliding window*” technique. We finish the analysis of the BlockCGSI algorithm in section 4.5, with a review of its main algebraic operations and computational costs. In section 4.6, we summarize some practical properties of the two techniques that we have used to exploit the computed spectral information in the solution of SPD linear systems, namely the deflation on the starting residual and the Spectral Low Rank Update preconditioner. Section 4.7 is concerned with numerical results illustrating the good behavior of the algorithm in general. Finally, in section 4.8, we include a cost-benefit analysis, and present some concluding remarks.

4.2 The BlockCGSI algorithm

In this section, we present and detail partly the BlockCGSI algorithm used to compute an M -orthonormal basis 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 4.2, $\lambda_{\max}$ and μ 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 $M^{-1}A$.

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 inexact subspace iteration (Algorithm 2.3.1), 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 basis of dimension s . The closer are these 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 4.2, 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 section 4.4.1.

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 ε . The appropriate choice of ε is detailed in section 4.3.3. 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).

To improve the rate of convergence of the inverse subspace iteration, the orthogonal iteration is followed by the Ritz acceleration (steps iv to vii), as suggested by Parlett (1998). The new vectors coming from the orthogonal iteration are gathered together with $W^{(k-1)}$ in the matrix $Q^{(k)}$, and 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 $\text{diag}(\Delta) = \delta_1, \dots, \delta_{p+s}$ 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

ALGORITHM 4.2: 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 in the range $]0, \mu]$ **Begin****Generate the initial subspace (with filtering)**

1. $Z^{(0)} = \text{RANDOM}(n, s)$
2. $Y^{(0)} = R^{-1} Z^{(0)} \Psi$ such that $Y^{(0)T} M Y^{(0)} = I_{s \times s}$
3. $Q^{(0)} = \text{Chebyshev-Filter}(Y^{(0)}, \xi, [\mu_f, \lambda_{\max}], A, R)$
4. $V^{(0)} = Q^{(0)} \Gamma$ such that $V^{(0)T} M V^{(0)} = I_{s \times s}$
5. $W^{(0)} = \text{empty}$
6. **For** $k = 1, \dots$, **until convergence** **Do**:
 - Orthogonal iteration**
 - i. Solve $M^{-1} A Z^{(k)} = V^{(k-1)}$ 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 $Q^{(k)T} M Q^{(k)} = I_{s \times s}$
 - Ritz acceleration**
 - iv. $Q^{(k)} = [W^{(k-1)} \quad Q^{(k)}]$
 - 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 $\Delta_k = \text{Diag}(\delta_1, \dots, \delta_{p+s})$ (Ritz Values)
 - vii. $V^{(k)} = Q^{(k)} U_k$ (Ritz Vectors)
 - "Sliding window"**
 - viii. $W^{(k)} = \text{converged columns of } V^{(k)}$
 - ix. $V^{(k)} = \text{non-converged columns of } V^{(k)}$
 - x. $(n, p) = \text{size}(W^{(k)})$
 - xi. Update the computational window ($V^{(k)}$)
 - xii. $(n, s) = \text{size}(V^{(k)})$

7. EndDo**End**

is the current block size. At step vii of the algorithm, the Ritz values and Ritz vectors $V = [v_1, v_2, \dots, v_p, \dots, v_{p+s}]$ are approximations to the eigenpairs in $M^{-1}A$ corresponding to the eigenvalues in the range $]0, \lambda_{p+s}]$. With the Ritz acceleration, the convergence rate of each individual eigenvalue in the subspace inverse iteration is of order $\lambda_i / \lambda_{p+s+1}$, with $1 \leq i \leq p + s$ (see Golub and van Loan (1996)). This is a good improvement compared to

the convergence rate of the orthogonal iteration, that is of order $\lambda_{p+s}/\lambda_{p+s+1}$.

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 section 4.3.1. 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 section 4.4.2.

4.3 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) subspace iteration, and the second loop, or inner iteration, is in the call to the blockCG algorithm (at step i in Algorithm 4.2) for the iterative solution of the linear system with multiple right-hand sides, $M^{-1}AZ^{(k)} = V^{(k-1)}$. These two iterative levels require each a specific stopping criterion in order to monitor the convergence of the algorithm. Sections 4.3.1 and 4.3.2 are devoted to and analyze some properties associated with these aspects. In section 4.3.3, 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.

4.3.1 Inverse subspace iteration (outer loop)

At each inverse iteration (k) in Algorithm 4.2, the blockCG algorithm solves the s linear systems $M^{-1}Az_j^{(k)} = v_j^{(k-1)}$, $j = 1, \dots, s$, 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

$$R^{-T}AR^{-1}Rz_j^{(k)} = Rv_j^{(k-1)} \iff \tilde{A}\tilde{z}_j^{(k)} = \tilde{v}_j^{(k-1)}, \quad j = 1, \dots, s. \quad (4.1)$$

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 corresponding eigenvalue in the current working set. The superscript (k) denotes the inverse iteration number, and the tilde refers to the symmetrized system (4.1).

The outer iteration produces a sequence of Ritz vectors $\tilde{v}_j^{(1)}, \tilde{v}_j^{(2)}, \dots, \tilde{v}_j^{(k)}$, that converge to the eigenvector $\tilde{u}_j = Ru_j$ corresponding to the eigenvalue λ_j of 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 4.2) are M -orthonormal. These two are linked by the simple change of basis,

$$v_j^{(k)} = R^{-1}\tilde{v}_j^{(k)},$$

which expresses the relation between the symmetrized system 4.1 and the preconditioned one. The corresponding Ritz values (diagonal elements of the matrix $\Delta^{(k)}$) are given by $\delta_j^{(k)} = v_j^{(k)T} A v_j^{(k)} = \tilde{v}_j^{(k)T} \tilde{A} \tilde{v}_j^{(k)}$.

The error bound on each approximate eigenpair, given by (Parlett, 1998, p. 357) is:

$$|\lambda_j - \delta_j^{(k)}| \leq \frac{\|A v_j^{(k)} - \delta_j^{(k)} M v_j^{(k)}\|_{M^{-1}}}{\|M v_j^{(k)}\|_{M^{-1}}} = \|M^{-1} A v_j^{(k)} - \delta_j^{(k)} v_j^{(k)}\|_M. \quad (4.2)$$

Dividing (4.2) by $\delta_j^{(k)}$ (as an approximation of λ_j) we obtain the estimates of the relative residual upper bound and of the number of correct digits in $\delta_j^{(k)}$. This estimate is used in step viii of Algorithm 4.2 to decide if a Ritz vector $v_j^{(k)}$ has converged or not, as for instance when

$$\frac{\|M^{-1} A v_j^{(k)} - \delta_j^{(k)} v_j^{(k)}\|_M}{\delta_j^{(k)}} \leq 10^{-1}, \quad (4.3)$$

if one correct digit is enough. However, the effective number of correct digits will be greater than one because the stopping criterion (4.3) is based on the invariance of the Ritz vectors v_j , and it is well known that the Ritz values δ_j converge faster than the corresponding vectors. We use this stopping criterion because, as we will see in section 4.6, 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.

With the Ritz acceleration incorporated in the algorithm, the convergence rate of the (inverse) subspace iteration (see section 3.5) is then given by

$$|\lambda_j - \delta_j^{(k)}| \sim \left(\frac{\lambda_j}{\lambda_{p+s+1}} \right)^k, \quad (4.4)$$

where λ_{p+s+1} is the eigenvalue immediately following the current set of $p+s$ approximated eigenvalues. As discussed in section 3.5, this property shows that if the inner iteration is accurate enough and if λ_j is well separated from λ_{p+s+1} , we can have a good estimation of the eigenvalue λ_j in a few number of inverse iterations.

4.3.2 The blockCG iteration (inner loop)

As we have seen in section 3.6.1, the block Conjugate Gradient (blockCG) algorithm under concern is a numerically stable variant 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 equation (4.1). For each system, $j = 1, \dots, s$, 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,

$$\tilde{z}_j^{[1]}, \tilde{z}_j^{[2]}, \dots, \tilde{z}_j^{[i]} \rightarrow \tilde{z}_j^{(k)}, \quad (4.5)$$

where the superscript $[i]$ stands for the blockCG (inner) iteration number, and the residual vector associated with each iterate $\tilde{z}_j^{[i]}$

$$\tilde{r}_j^{[i]} = \tilde{v}_j^{(k-1)} - \tilde{A}\tilde{z}_j^{[i]}. \quad (4.6)$$

The starting point of the following study comes from the theoretical analysis given in (Parlett, 1998, pp.332–335), which shows that the Ritz vectors $V^{(k)}$ actually converge to the solution vectors $Z^{(k)}$ at the same rate as these $Z^{(k)}$ vectors converge to the set of targeted eigenvectors. Therefore, we plan to use the pair $(\tilde{z}_j^{[i]}, \tilde{\delta}_j^{[i]})$, 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 $\tilde{z}_j^{[i]}$ at each inner iteration i , as an approximation of the eigenpair $(\tilde{u}_{p+j}, \lambda_{p+j})$ in the symmetrized system (4.1), and to change the upper bound in (4.2) using this approximate eigenpair instead. Note that the subscript index $p+j$ takes in fact into account the “sliding window” feature detailed in section 4.4.2.

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

$$\tilde{S}_j^{[i]} = \tilde{A}\tilde{z}_j^{[i]} - \tilde{\delta}_j^{[i]}\tilde{z}_j^{[i]} = \tilde{v}_j^{(k-1)} - \tilde{\delta}_j^{[i]}\tilde{z}_j^{[i]} - \tilde{r}_j^{[i]}. \quad (4.7)$$

The error bound (4.2) applied on the symmetrized system, using the approximate eigenpair $(\tilde{z}_j^{[i]}, \tilde{\delta}_j^{[i]})$ available at each inner iteration i , yields

$$|\lambda_{p+j} - \delta_j^{[i]}| \leq \frac{\|\tilde{S}_j^{[i]}\|_2}{\|\tilde{z}_j^{[i]}\|_2} = \frac{\|\tilde{v}_j^{(k-1)} - \delta_j^{[i]}\tilde{z}_j^{[i]} - \tilde{r}_j^{[i]}\|_2}{\|\tilde{z}_j^{[i]}\|_2}. \quad (4.8)$$

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

$$\|\tilde{v}_j^{(k-1)} - \delta_j^{[i]}\tilde{z}_j^{[i]} - \tilde{r}_j^{[i]}\|_2^2 = \|\tilde{v}_j^{(k-1)} - \delta_j^{[i]}\tilde{z}_j^{[i]}\|_2^2 + \|\tilde{r}_j^{[i]}\|_2^2. \quad (4.9)$$

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

$$\begin{aligned} |\lambda_{p+j} - \delta_j^{[i]}| &\leq \frac{\|M^{-1}Az_j^{[i]} - \delta_j^{[i]}z_j^{[i]}\|_M}{\|z_j^{[i]}\|_M} = \sqrt{\frac{\|v_j^{(k-1)} - \delta_j^{[i]}z_j^{[i]}\|_M^2 + \|r_j^{[i]}\|_M^2}{\|z_j^{[i]}\|_M^2}} \\ &\stackrel{def}{=} \sqrt{\phi_j^{[i]2} + \omega_j^{[i]2}}, \end{aligned} \quad (4.10)$$

where we can see that the error bound associated with each Rayleigh quotient in 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_{p+j} . Therefore, the bound in (4.10) 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 θ_j in the M -norm between z_j^* and $v_j^{(k-1)}$, whose cosine is given by

$$\cos(\theta_j) = \frac{\langle z_j^*, v_j^{(k-1)} \rangle_M}{\|z_j^*\|_M \|v_j^{(k-1)}\|_M} = \delta_j^* \|z_j^*\|_M. \quad (4.11)$$

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

$$\sin(\theta_j) = \|v_j^{(k-1)} - \delta_j^* z_j^*\|_M, \quad (4.12)$$

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 (4.10) is

$$\phi_j^{[i]} \xrightarrow{i \rightarrow \infty} \frac{\sin(\theta_j)}{\|z_j^*\|_M} = \delta_j^* \tan(\theta_j). \quad (4.13)$$

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 θ_j should be very small, as well as the corresponding asymptotic limit of $\phi_j^{[i]}$. With respect to the bound in (4.10), 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.

Finally, we would also like to mention that the theoretical analysis developed in this section, which tries to understand the basic components in the spectral residual associated with the blockCG iterates, does not depend on the fact that blockCG is used within the subspace iteration, and can be generalized in any other situation to measure the proximity of the blockCG iterates with the closest eigenpair (u_ℓ, λ_ℓ) of the given linear system.

4.3.3 Stopping criterion for the blockCG

The stopping criterion for the blockCG defines the approximation degree of $\tilde{z}_j^{(k)} \approx \tilde{A}^{-1} \tilde{v}_j^{(k-1)}$ or equivalently of $z_j^{(k)} \approx A^{-1}Mv_j^{(k-1)}$. 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 (4.4)).

We propose to monitor only the convergence of the iterates $z_1^{[i]}$, corresponding to the smallest non-converged 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

$$\omega_1^{[i]} = \frac{\|v_1^{(k-1)} - M^{-1}Az_1^{[i]}\|_M}{\|z_1^{[i]}\|_M}, \quad (4.14)$$

and is readily available in the blockCG iteration (see section 3.6.1). Notice also that $\omega_1^{[i]}$ is very close to (2.8), i.e.

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

which is based on the usual Rigel and Gaches (1967) backward error measure, 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 eigenvalues through the approximate relative residual upper bound, as indicated in (4.2) and (4.3). At inverse iteration $(k-1)$, we consider that a Ritz value $\delta_j^{(k-1)}$ has converged when it has at least t correct digits, which implies that

$$\frac{\|M^{-1}Av_j^{(k-1)} - \delta_j^{(k-1)}v_j^{(k-1)}\|_M}{\delta_j^{(k-1)}} \leq 10^{-t}, \quad (4.15)$$

where $\delta_j^{(k-1)}$ is used in the denominator as an estimate of the corresponding real value of λ_{p+j} .

In order to satisfy (4.15) in the current inverse iteration (k) , the stopping criterion in the blockCG is set as

$$\omega_1^{[i]} \leq \varepsilon, \quad \text{with} \quad \varepsilon = 10^{-t}\delta_1^{(k-1)}, \quad (4.16)$$

where $\delta_1^{(k-1)}$ is the smallest of the current set of non converged Ritz values. This stopping criterion is based on the upper bound (4.10), assuming that the value of $w_1^{[i]}$ governs the absolute error measure $|\lambda_{p+1} - \delta_1^{[i]}|$ in the inner iteration because $\phi_1^{[i]}$ is not dominant. 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, as we will see in the experiments, when the starting vectors are previously filtered with Chebyshev polynomials.

The second assumption, already motivated at the beginning of the analysis in section 4.3.2, is that the upper bound of $|\lambda_{p+1} - \delta_1^{(k)}|$ in the outer iteration, given by equa-

tion (4.2), will be close to (and can be replaced by) the upper bound of $|\lambda_{p+1} - \delta_1^{[i]}|$ from the inner iteration, and given by equation (4.10). We recall indeed that the Ritz vectors $V^{(k)}$ actually converge to the solution vectors $Z^{(k)}$ at the same rate as these $Z^{(k)}$ vectors converge to the set of targeted eigenvectors (Parlett, 1998, pp.332-335).

Under these assumptions, the idea in (4.16) is to achieve a given accuracy $\varepsilon_{outer} = 10^{-t}$ in (4.15) while minimizing the number of blockCG iterations. Note also that the strategy (4.16) for the stopping criterion is in agreement with other analysis of the inexact inverse iteration, like for instance in Lai et al. (1997); Smit and Paardekooper (1999); Golub and Ye (2000); Berns-Mueller et al. (2005), where it is suggested to use a decreasing value for the inner threshold parameter ε . This is the case, indeed, in (4.16) since the value of $\delta_1^{(k)}$ decreases gradually towards λ_{p+1} along with the convergence of the outer iteration.

As we have seen in section 4.3.2, when $\phi_1^{[i]}$ has reached its stagnation level defined in (4.13), the bound in (4.10) 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 $\delta_1^{(k)}$ with respect to λ_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 Notay (2002) and, in some way, to the one also proposed in Simoncini and Eldén (2002), 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 chosen relative precision 10^{-t} not too small ($t = 1$ or 2 , for instance). 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 matrix. However, if one is interested in computing an accurate invariant subspace, the inner threshold parameter ε in (4.16) should be set to the maximum between $10^{-t}\delta_1^{(k-1)}$ and the asymptotic value of $\phi_1^{[i]}$ in (4.13). From the analysis in 4.3.2, 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 (4.13) tends to zero proportionally to the tangent of the angle θ_j . Because of that, it is ensured that after some appropriate number of inverse (outer) iterations, the maximum between $10^{-t}\delta_1^{(k-1)}$ and this asymptotic value of $\phi_1^{[i]}$ will always remain the first of these two, and the targeted accuracy in inverse iteration can then be expected to be achieved in the end.

4.4 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 4.2.

4.4.1 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 the 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 Arioli and Ruiz (2002).

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

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

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

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

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

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

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

where ξ , 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 μ_f , $\lambda_{\max}$ and ξ , the degree m depends on the ratio $\lambda_{\max}/\mu_f$ and is inversely proportional to ξ .

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 μ_f will minimize the number of inverse iterations but will increase strongly the computational efforts in the Chebyshev filtering step. This will be analysed in more details in the experiments.

4.4.2 Sliding window

The original version of the BlockCGSI algorithm (see Algorithm 2.3.1) 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 dimension 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 (Dongarra et al., 1990) 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 4.2), 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 4.2). 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. This technique is also described as “*guard vectors*” in the literature (see section 3.5).

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, after the discovery of the extreme eigenvalues, some of the al-

ALGORITHM 4.4.2: INCORPORATE NEW VECTORS**Inputs:** $A, M = R^T R \in \mathbb{R}^{n \times n}, V^{(k)} \in \mathbb{R}^{n \times (s-\ell)}, W^{(k)} \in \mathbb{R}^{n \times p}, \mu_f, \lambda_{\max} \in \mathbb{R}, \ell \in \mathbb{N}$ **Begin**

- a) $P = \text{RANDOM}(n, \ell)$
- b) $Q = R^{-1} P \Psi$ such that $Q^T M Q = I_{\ell \times \ell}$
- c) $P = \text{Chebyshev-Filter}(Q, \xi, [\mu_f, \lambda_{\max}], A, R)$
- d) $Q = P \Gamma$ 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

ready converged Ritz vectors may appear as not enough invariant, 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 4.2). 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 4.2. The operation that consists in introducing new vectors, after a set of ℓ Ritz vectors has converged, is detailed in Algorithm 4.4.2. 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 $V^{(k)}$.

4.5 Operation counts

The pre-computation of the basis W of a *near*-invariant subspace with the BlockCGSI algorithm, has a cost which we denote by $\mathcal{C}_{BCGSI}$. 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 ξ and the cut-off filtering value μ_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 4.1 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 4.2 will be expressed as a function of these basic operations. We denote the computational cost of one operation OP by the

symbol $\mathcal{C}_{OP}$, like for instance $\mathcal{C}_A$ that is the cost of one 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 $\mathcal{C}_M$. 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 4.1, where $nnz(R)$ is the number of nonzero elements in the factor R from the Incomplete Cholesky or Jacobi factorization.

Table 4.1: Basic operations counts in BlockCGSI algorithm

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

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 $\mathcal{C}_{CHEBY} \approx s\mathcal{C}_A + s\mathcal{C}_M + 3s\mathcal{C}_{AXPY}$. 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

$$\mathcal{C}_{START} \approx s\mathcal{C}_{RAND} + 2\mathcal{C}_{ORTHO(s)} + \frac{s}{2}\mathcal{C}_M + \text{ChebIt} \times \mathcal{C}_{CHEBY}, \quad (4.17)$$

where ChebIt is the total number of Chebyshev filtering iterations.

The orthogonal iteration represents the most expensive step in Algorithm 4.2 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:

$$\mathcal{C}_{bCG} \approx s\mathcal{C}_A + s\mathcal{C}_M + 3\mathcal{C}_{GEMM} + 2\mathcal{C}_{ORTHO(s)}. \quad (4.18)$$

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 orthogonal iteration at each inverse iteration resumes in:

$$\mathcal{C}_{QR} \approx \text{bCGIt}(k) \times \mathcal{C}_{bCG} + s\mathcal{C}_{PROJ(p)} + \mathcal{C}_{ORTHO(s)} + \mathcal{C}_M, \quad (4.19)$$

where $\text{bCGIt}(k)$ is the number of blockCG inner iterations performed at inverse iteration (k), and where $\mathcal{C}_{\text{PROJ}(p)} = \mathcal{C}_M + p\mathcal{C}_{\text{DOT}} + \mathcal{C}_{\text{AXPY}}$. The cost of the orthogonal iteration cannot be determined *a priori*, because the parameters $\text{bCGIt}(k)$ 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

$$\mathcal{C}_{\text{RITZ}} \approx (s + p)\mathcal{C}_A + 2(s + p)^2\mathcal{C}_{\text{DOT}} + 5(s + p)^3, \quad (4.20)$$

where $5(s + p)^3$ is the cost corresponding to the spectral decomposition of the matrix β_k at step vi in Algorithm 4.2.

The amount of work to update the computational window is induced by the convergence test, equations (4.2) and (4.3), and by the incorporation of the new vectors (see Algorithm 4.4.2). If we maintain the same block size s , the ℓ converged vectors are replaced in the computational window by ℓ 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

$$\mathcal{C}_{\text{UPDATE}} \approx \mathcal{C}_{\text{INV}} + \ell\mathcal{C}_{\text{RAND}} + \text{ChebIt} \times \mathcal{C}_{\text{CHEBY}} + 2\mathcal{C}_{\text{ORTHO}(\ell)} + \frac{\ell}{2}\mathcal{C}_M + \ell\mathcal{C}_{\text{PROJ}(p)}, \quad (4.21)$$

where

$$\mathcal{C}_{\text{INV}} \approx 2(s + p)\mathcal{C}_M + (s + p)\mathcal{C}_{\text{AXPY}} + 2(s + p)\mathcal{C}_{\text{DOT}} \quad (4.22)$$

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

$$\mathcal{C}_{\text{BCGSI}} \approx \mathcal{C}_{\text{START}} + \text{InvIt} \times (\mathcal{C}_{\text{QR}} + \mathcal{C}_{\text{RITZ}} + \mathcal{C}_{\text{UPDATE}}). \quad (4.23)$$

4.6 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 Giraud et al. (2004). Here, we summarize the two techniques introduced in chapter 2, 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.

4.6.1 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:

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

where W and Δ 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

$$x^{(1)} = W\Delta^{-1}W^Tb. \quad (4.24)$$

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 μ and $\lambda_{\max}$, as explained in Hageman and Young (1981). In this way, the Conjugate Gradient should reach linear convergence immediately (van der Sluis and van der Vorst, 1986). For clarity, we will call INIT-CG the algorithm corresponding to CG with a starting guess obtained with (4.24).

4.6.2 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 Carpentieri et al. (2003), 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

$$\widehat{M} = M^{-1} + W\Delta^{-1}W^T. \quad (4.25)$$

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 Carpentieri et al. (2003)).

In the following, we will call SLRU-CG to the algorithm corresponding to the preconditioned Conjugate Gradient with SLRU as preconditioner (4.25).

4.6.3 Practical considerations

We first consider operations count. As in section 4.5, 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 section 4.5).

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 at step vi of Algorithm 4.2. The cost of its inversion is not significant because we consider it as a diagonal matrix. We note that Δ^{-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 $W\Delta^{-1}W^T r^{(k)}$ at each iteration (see equation (4.25)). This projection can be done using common level 2 BLAS operations Dongarra et al. (1988) with a total cost roughly equal to

$$C_{\text{Proj}} \approx 4(p+1)n. \quad (4.26)$$

In Table 4.2, we indicate the total cost in floating-point operations for each algorithm, with an initial cost, and a cost per iteration.

Table 4.2: Cost in floating-point operations for different methods.

	THE COST IN FLOATING-POINT OPERATIONS	
	AT THE BEGINNING	AT EACH ITERATION
CG	$C_A + C_M + - + -$	$C_A + C_M + 3C_{\text{AXPY}} + 2C_{\text{DOT}} + -$
INIT-CG	$C_A + C_M + C_{\text{Proj}} + -$	$C_A + C_M + 3C_{\text{AXPY}} + 2C_{\text{DOT}} + -$
SLRU-CG	$C_A + C_M + C_{\text{Proj}} + -$	$C_A + C_M + 3C_{\text{AXPY}} + 2C_{\text{DOT}} + C_{\text{Proj}}$

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 (4.24)). Anyway, this also contributes to better numerical stability in the convergence process of SLRU-CG.

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 $\text{diag}(\Delta)$ the corresponding Ritz Vectors.

4.7 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 $M^{-1}A$. This is illustrated on a test matrix coming from the 2D heterogeneous diffusion equation in a L shape region, discretized by finite elements, with size $n = 7969$. We also precondition the resulting linear system with Jacobi scaling or classical Incomplete Cholesky (see table 4.3).

Table 4.3: 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	$3.07e-09$	$2.08e+00$	2	9	18	7969	55131	7969
IC(0)	$1.66e-08$	$1.55e+00$	2	2	3	31550		

In figure 4.1, we show the convergence behavior of the values ω_1 , ϕ_1 and the upper bound $\sqrt{\omega_1^2 + \phi_1^2}$ of $|\lambda_1 - \delta_1|$. As developed in section 4.3.2, these three magnitudes are related by equation (4.10). The two plots in figure 4.1 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 ϕ_1 much smaller than what it can be with a randomly generated initial set of vectors. The direct consequence is that ω_1 becomes then a good measure of the upper bound of $|\lambda_1 - \delta_1|$, 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 ϕ_1 in the first inverse iteration, allowing a larger range of values for the choice of the threshold ε in the blockCG, which is a desirable feature for the algorithm as discussed in section 4.3.3.

In table 4.4, we present both the total number of inner and outer iterations in the BlockCGSI algorithm (see Algorithm 4.2) to compute a *near*-invariant subspace associated with all eigenvalues in the range $]0, \mu[$. The requested relative precision in these approximated eigenvalues was set to one correct digit ($t = 1$), with respect to the convergence criterion for the outer loop given by equation (4.15), and the stopping criterion for the blockCG set accordingly as in (4.16). The total number of inverse iterations is indicated by InvIt, and

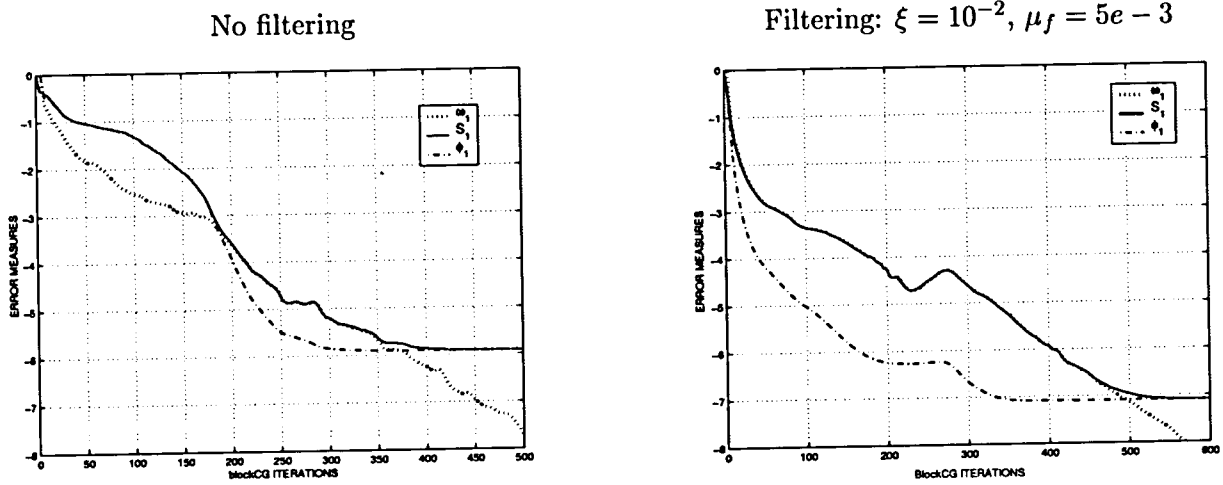


Figure 4.1: Correlation between the residual measures ω_1 , ϕ_1 and the upper bound S_1 on the spectral error $|\lambda_1 - \delta_1|$ in the blockCG with block size $s = 4$ at the first inverse iteration. Matrix A preconditioned with Jacobi scaling.

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 4.4.2).

Finally, we also include the total number of floating point operations performed by the BlockCGSI algorithm ($\mathcal{C}_{BCGSI}$), in millions (Mflops), computed as in (4.23). 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 μ_f correspond to the cases when it is greater or equal to the principal cut-off value μ in Algorithm 4.2 ($]0, \mu[$ defines the range of targeted eigenvalues, and q is the number of eigenvalues in this range).

The results in table 4.4 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 section 3.5), 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 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 q of eigenvalues in the range $]0, \mu[$ when the block size is not equal to q .

As we can observe in table 4.4, the filtering of the new vectors with Chebyshev polynomials can improve quite a lot the efficiency of the BlockCGSI algorithm. As the filtering level ξ 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 ϕ_j becomes much lower (see also figure 4.1). This also gives room for larger decrease of the error $|\lambda_j - \delta_j|$ 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 ($\text{InvIt} = 1$ 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 ξ 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 ξ , that minimizes this computational work (Mflops), also depends on the other filtering parameter μ_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 μ_f not too large with respect to the actual bound μ on the range of targeted eigenvalues, and also that when the number of targeted eigenvalues is small, a good filtering level ξ is indicated.

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 figure 4.2, we plot the backward error

$$\rho^{(i)} = \frac{\|M^{-1}r^{(i)}\|_M}{\|M^{-1}r^{(0)}\|_M} = \frac{\|R^{-T}r^{(i)}\|_2}{\|R^{-T}r^{(0)}\|_2}, \quad (4.27)$$

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 figure 4.2), 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 figure 4.2).

Regarding the results in figure 4.2, the SLRU preconditioner is numerically more stable than the deflation technique. The SLRU-CG converges linearly while the INIT-CG loses 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 figure 4.3. Indeed, with larger values for the number of correct digits t (see formula (4.15)) the irregularities

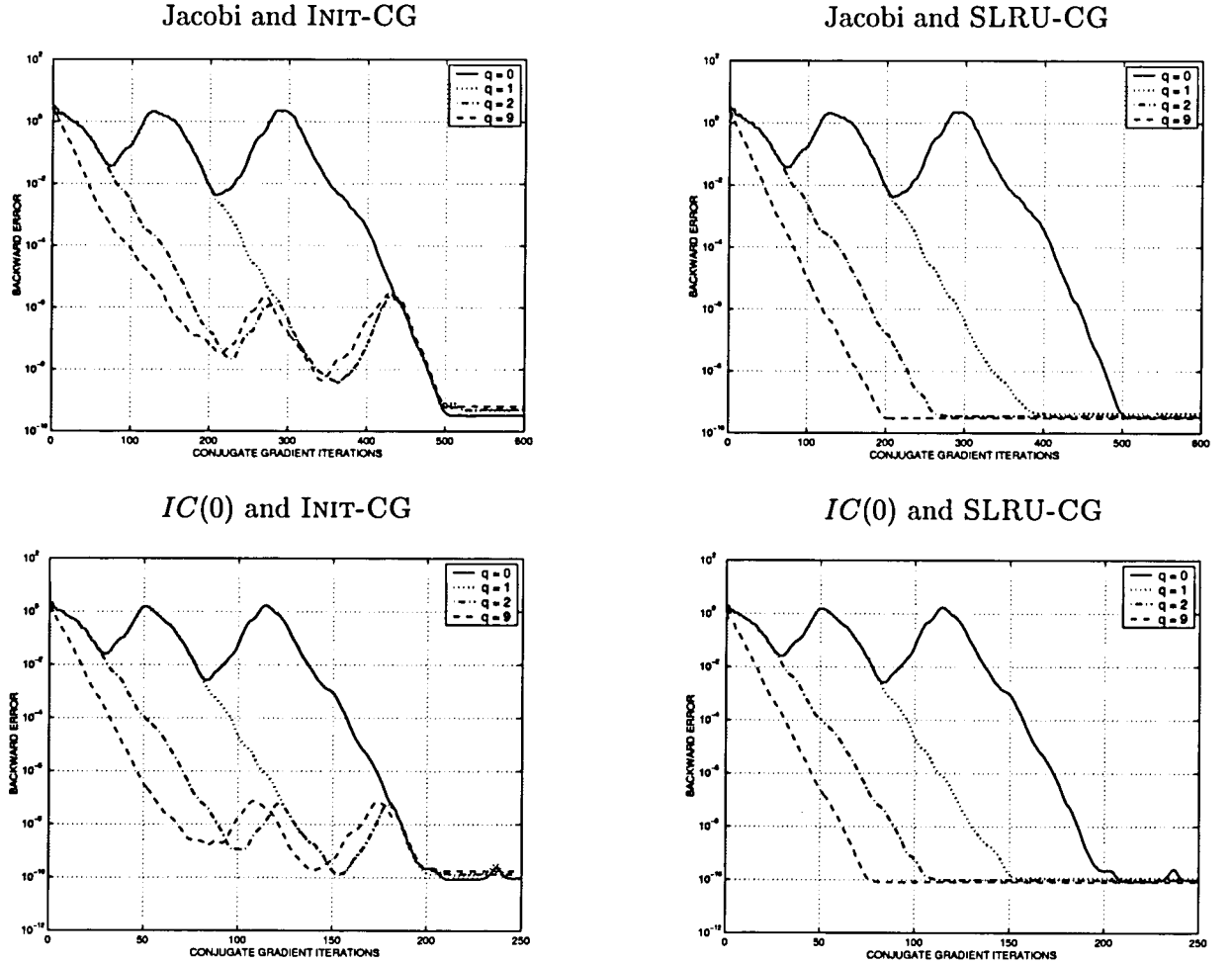


Figure 4.2: Convergence behavior of INIT-CG and SLRU-CG with different dimensions q of the pre-computed *near*-invariant subspace. The system is initially preconditioned either with Jacobi scaling (top) or with the standard Incomplete Cholesky (bottom).

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.

4.8 Cost-benefit analysis

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

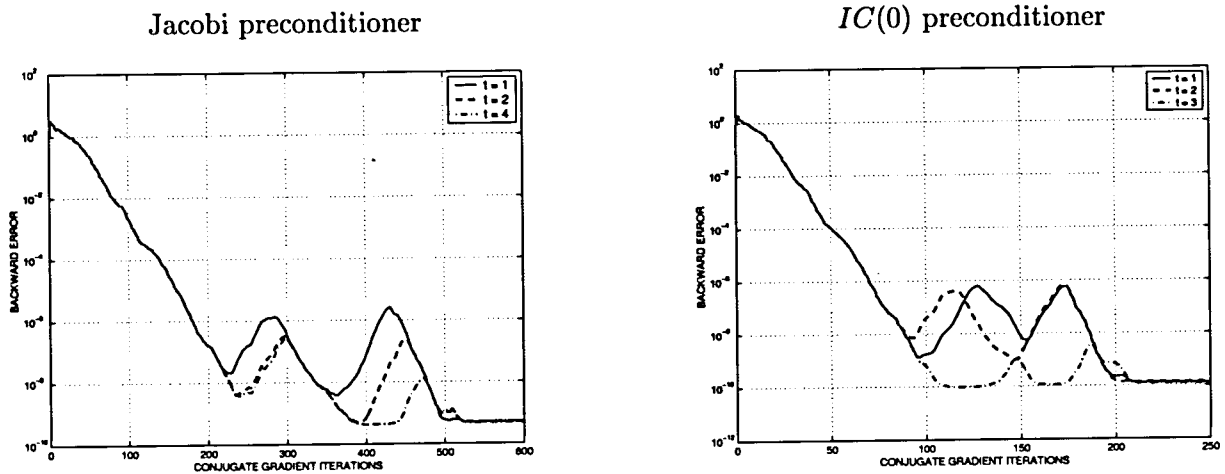


Figure 4.3: Effect of the accuracy of the spectral information on the convergence behavior of INIT-CG.

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 section 4.6.3). In figure 4.4, we plot the histogram 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 (4.25) 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 figure 4.4, 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 figure 4.2). As opposed to that, we can observe in the case of INIT-CG algorithm (left of figure 4.4) that we always get improvements with larger values of q .

As we have seen in section 4.5, the pre-computation of the *near*-invariant subspace W has a cost, that we denote by $\mathcal{C}_{BCGSI}$, 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 4.5, we present the computational costs $\mathcal{C}_{BCGSI}$ (in millions of operations, Mflops) for three different cases that correspond to different choices for the cut-off value μ . 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,

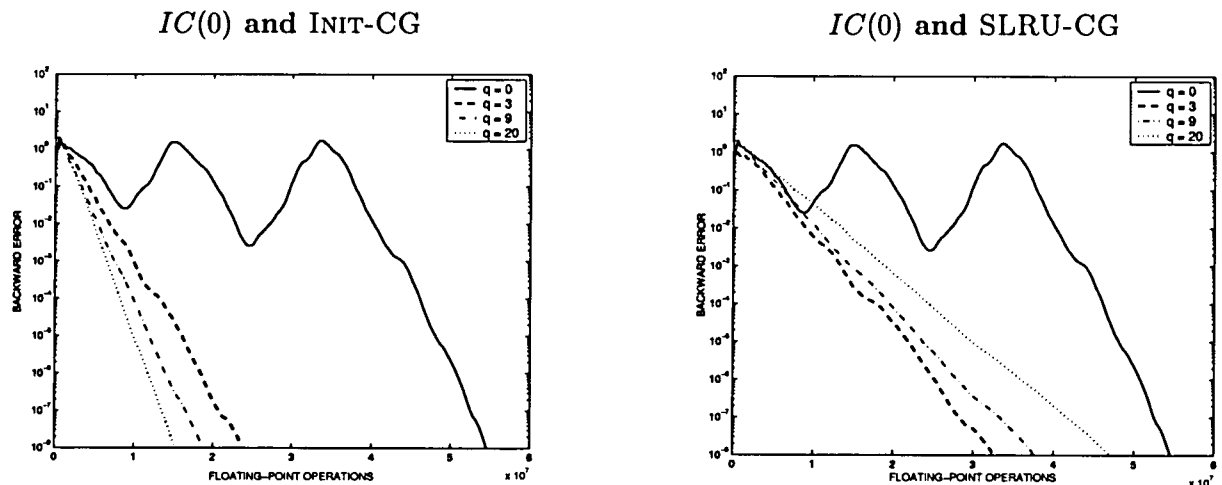


Figure 4.4: History of the backward error as a function of the computational cost, for different dimensions q of the *near*-invariant subspace. The case $q = 0$ corresponds to the classical CG algorithm, for comparison.

and with one digit ($t = 1$) in the case of $IC(0)$.

To illustrate the cost-benefits, we stop the CG iterations when the relative residual norm $\rho^{(i)}$ (see equation (4.27)) 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 $\mathcal{C}_{BCGSI}$ is compensated. The number of amortization right-hand sides is given by

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

where $\mathcal{C}_{CG}$ is the cost of CG algorithm without acceleration, and $\mathcal{C}_{aCG}$ is cost of the accelerated CG, either INIT-CG or SLRU-CG algorithm. These informations are given for each cut-off value μ . For instance, in table 4.5, 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 4.5 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

Table 4.5: Cost-benefit of CG accelerated with the spectral information

	Spectral fact.			INIT-CG			SLRU-CG		
	μ	q	C_{BCGSI} Mflops	CG		Amor. rhs.	CG		Amor. rhs.
				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

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 multiples 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 BlockCGSI 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.

Chapter 5

Validation of the strategy on an airflow problem

5.1 Description of the problem

We validate our two-phase accelerating technique in an airflow simulation code named ICARE (c.f. Braza (1986); Bergmann (2001); Martinat (2003)) that simulates the flow around a wing of plane. The Navier-Stokes equations are solved by finite elements in a 2D field, through a prediction-correction algorithm and a semi-implicit discretization scheme. The code describes the transition of the flow from the laminar to the turbulent state. At each time step iteration a system of linear equations with the same coefficient matrix and changing right-hand sides is solved. We will see that our two-phase accelerating technique can be applied with success to reduce the amount of work needed to solve all the consecutive systems.

In the first time-step the BlockCGSI algorithm computes a basis W of a *near*-invariant subspace associated with the eigenvalues in the range $]0, \mu[$ of the preconditioned matrix $M^{-1}A$. This information is used in a second phase to improve the consecutive solutions of the systems with the same coefficient matrix and changing right-hand sides.

We begin in section 5.2 with a little explanation of the numerical procedure used to solve the Navier-Stokes equations and we highlight the origin of the linear system of equations used as test problem. In section 5.3, we present some numerical properties of the coefficient matrix of this system. After that, in section 5.4 we indicate how to use correctly the INIT-CG algorithm even if the spectral information is inaccurate. It follows, in section 5.5, an *a posteriori* analysis of the computational costs involved in the proposed acceleration technique, and we indicate the optimal value for the dimension of the base W that minimizes the costs. We also propose a strategy for choosing dynamically the dimension of the base in function of the Ritz value distribution. We finish this chapter with

section 5.6 where we summarize the benefits, in computational cost, that derive from the use of the two-phase accelerating technique in the present simulation of an airflow problem.

5.2 Numerical solution of the Navier-Stokes equations

We begin by a short description of the numerical method used in the ICARE algorithm to solve the Navier-Stokes equations (for details see Braza (1986); Bergmann (2001); Martinat (2003)). As in Chapter 2 the test case comes from the simulation of a two-dimensional flow of an incompressible constant-property Newtonian fluid, but of an unsteady flow, the momentum equation includes an additional unsteady term. The flow equations in dimensionless and tensorial form are given by

$$\frac{\partial u_i}{\partial x_j} = 0, \quad (5.1)$$

and

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \frac{1}{Re} \frac{\partial^2 u_i}{\partial x_j \partial x_j}, \quad (5.2)$$

where Re is the Reynolds number. Assuming that u_i and P are known at instant $N\Delta t$ and using an implicit method,

$$\text{div} \vec{V}^N = 0, \quad (5.3)$$

$$\frac{\vec{V}^{N+1} - \vec{V}^N}{\Delta t} + \text{div}(\vec{V}^{N+1} \vec{V}^{N+1}) = -\vec{\text{grad}} P^{N+1} + \frac{1}{Re} \text{div}(\vec{\text{grad}} \vec{V}^{N+1}). \quad (5.4)$$

The convective term is linearized through a semi-implicit scheme where $\vec{V}^{N+1}$ is carried by $\vec{V}^N$:

$$\frac{\vec{V}^{N+1} - \vec{V}^N}{\Delta t} + \text{div}(\vec{V}^N \vec{V}^{N+1}) = -\vec{\text{grad}} P^{N+1} + \frac{1}{Re} \text{div}(\vec{\text{grad}} \vec{V}^{N+1}). \quad (5.5)$$

Because the pressure P^{N+1} is unknown, an intermediate (predictor) field $(\vec{V}^*, P^*)$, that also verifies the momentum equation, is introduced:

$$\frac{\vec{V}^* - \vec{V}^N}{\Delta t} + \text{div}(\vec{V}^N \vec{V}^*) = -\vec{\text{grad}} P^* + \frac{1}{Re} \text{div}(\vec{\text{grad}} \vec{V}^*). \quad (5.6)$$

As the predictor step assumes that $P^* = P^N$, the following equation is solved in order to get $\vec{V}^*$:

$$\frac{\vec{V}^* - \vec{V}^N}{\Delta t} + \text{div}(\vec{V}^N \vec{V}^*) = -\vec{\text{grad}} P^N + \frac{1}{Re} \text{div}(\vec{\text{grad}} \vec{V}^*). \quad (5.7)$$

From (5.7), $\vec{V}^*$ verifies the same boundaries conditions as $\vec{V}^{N+1}$ and, taking the rotational of equation (5.5) and equation (5.7), it is possible to prove that the field $\vec{V}^*$ transports the exact rotational value at time step $(N+1)$, i.e.

$$\text{r\ddot{ot}}(\vec{V}^*) = \text{r\ddot{ot}}(\vec{V}^{N+1}). \quad (5.8)$$

However, $\vec{V}^*$ does not verify the continuity equation. It is then necessary to introduce a correction term such that

$$\text{div}(\vec{V}^{**}) = 0. \quad (5.9)$$

The field $\vec{V}^{**}$ is introduced through an auxiliary potential function ψ , such that

$$\vec{V}^{**} = \vec{V}^* - \text{grad}\psi. \quad (5.10)$$

Taking the divergence of the equation (5.10) and using (5.9), the potential function ψ is given by the Poisson equation

$$\nabla^2\psi = \text{div}\vec{V}^*. \quad (5.11)$$

The solution of the linear system of equation (5.11) enables to get the function ψ directly from $\vec{V}^*$. The corrected velocity $\vec{V}^{**}$ is then computed from equation (5.10). Because $\text{div}(\vec{V}^{**}) = 0$, $\text{rot}(\vec{V}^{**}) = \text{rot}(\vec{V}^*) = \text{rot}(\vec{V}^{N+1})$ and $\vec{V}^{**}$ and $\vec{V}^{N+1}$ have the same boundary conditions, it is also proved that $\vec{V}^{**} = \vec{V}^{N+1}$.

The pressure at the time step $(N+1)$ is then computed by substituting $\vec{V}^{N+1}$, in (5.5), by the expression (5.10), and subtracting (5.7), we get,

$$\text{grad}P^{N+1} = \text{grad}(P^N + \frac{\psi}{\Delta t}) + \text{div}(\vec{V}^N \text{grad}\psi) - \frac{1}{Re} \nabla^2 \text{grad}\psi, \quad (5.12)$$

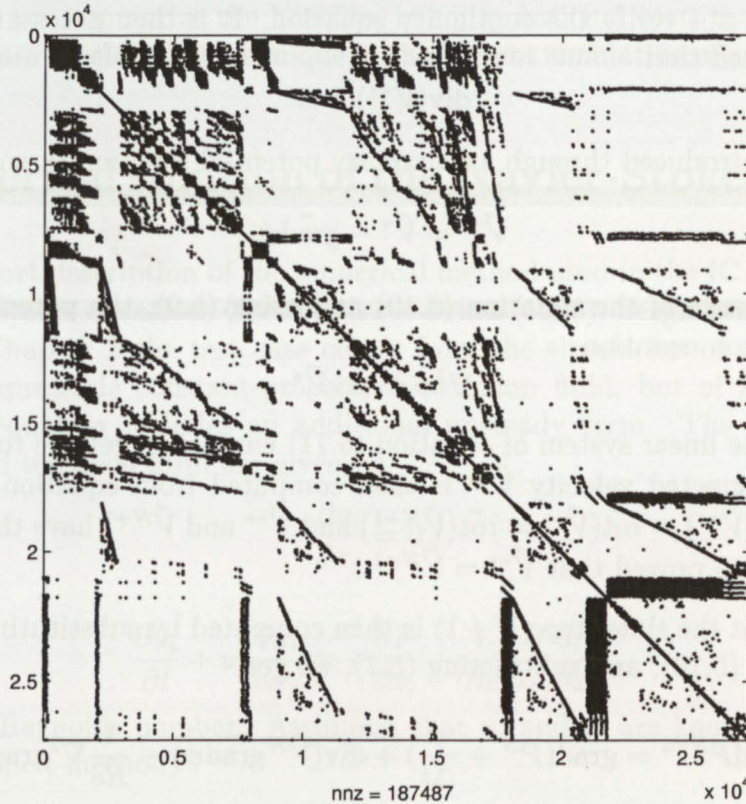
that is simplified in the following update

$$P^{N+1} = P^N + \frac{\psi}{\Delta t}. \quad (5.13)$$

5.3 Properties of the coefficient matrix

We present now some properties of the coefficient matrix A of the system $Ax = b_i$ solved at the i -th iteration in time. As we have seen in section 5.2, this system results from the Poisson equation (5.11), solved by finite elements. The matrix A of size $n = 27283$ has $nnz = 187487$ non-zeros elements distributed in agreement with the pattern presented in figure 5.1.

In figure 5.2, we plot the distribution of the 100 smallest and 5 largest eigenvalues of the coefficient matrix, with and without preconditioning. As we can observe on the left of figure 5.2, the coefficient matrix A has a spectrum distributed from $\lambda_{\min} = 5.0e - 05$ to $\lambda_{\max} = 1.1e + 01$, responsible for a condition number near $\kappa = 2.2e + 05$. After preconditioning by means of the classical Incomplete Cholesky ($M = R^T R = IC(0)$), the spectrum of the coefficient matrix is ranged from $\lambda_{\min} = 6.5e - 05$ to $\lambda_{\max} = 1.7e + 00$, which corresponds to a condition number near $\kappa = 2.6e + 04$. This first level of preconditioning helps to cluster the spectrum, specially the dominant part that is shifted from near 10

Figure 5.1: Pattern of the coefficient matrix A .

to near 1. The effect of the other end of the spectrum will be swallowed by the initial deflation in the INIT-CG algorithm.

As we can see in figure 5.2, the spectrum of $M^{-1}A$ is essentially clustered between $1.0e - 02$ and $1.7e + 00$, with 30 eigenvalues smaller than $1.0e - 02$. If we want, for instance, to reduce the condition number to about 100, we need to cancel the effect of the 48 smallest eigenvalues (e.g. all those in $]0, \lambda_{\max}/100[$).

Our purpose is to show that we can improve the global convergence of the simulation algorithm through the acceleration of the solution of the system $M^{-1}Ax = M^{-1}b_i$ in each time step iteration i . At the first time step we compute, with the BlockCGSI algorithm, a *near*-invariant subspace responsible for the ill-conditioning of the system. In the remaining time step iterations, this spectral information is used to improve the solution of the systems with the INIT-CG algorithm. The clusterization of the spectrum can appear as a limitation of the proposed acceleration technique. We will see that, by removing only the effect of the extremal eigenvalues, we can get a good improvement of the convergence rate for the solution of these systems.

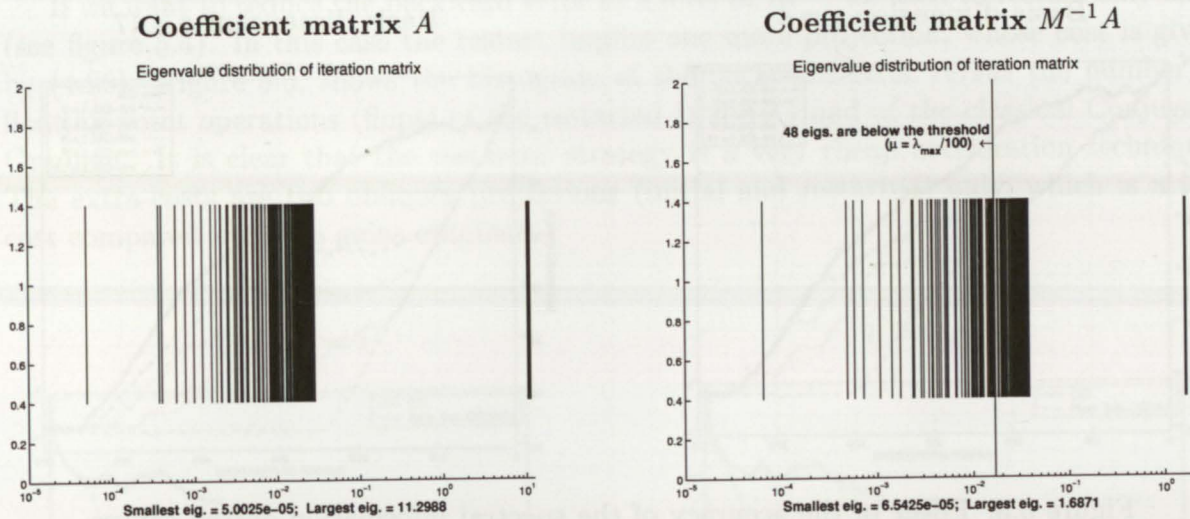


Figure 5.2: Distribution of the 100 smallest and 5 largest eigenvalues of the coefficient matrix.

5.4 Improving the systems solution with an inaccurate basis

As we have seen in the chapter 4, the INIT-CG accelerating technique is much cheaper compared to SLRU-CG. The difficulty is when the pre-computed basis vectors W are obtained with poor accuracy, as for instance when the upper bound of the number of correct digits in (4.15) is set to $t = 1$. In these cases, after some CG iterations, the *near*-invariance of the deflated residual is lost, and the initial oblique projection does not improve any more the speed of convergence of the INIT-CG algorithm, that behaves again like the classical Conjugate Gradient.

Figure 5.3 shows the convergence behavior of the INIT-CG algorithm in function of the accuracy of the spectral information, given by t , the upper bound on the number of correct digits used to monitor the convergence of the Ritz values $\delta_j^{(k)}$ in step viii of Algorithm 4.2. The plot on the left is obtained with a basis W with dimension $q = 4$, and on the right with $q = 17$. First, we observe that, even with a poor accuracy ($t = 1$), the convergence is linear until a certain level (around 10^{-6}), and as the values of t increase, the irregularities in the rate of convergence of the INIT-CG are gradually smoothed. After the phenomenon of plateaus, that occurs at a level linked to the accuracy of the basis W , the convergence history resembles closely the convergence of the classical Conjugate Gradient.

As shown in the previous chapter, one remedy to the lost of linear convergence is the SLRU-CG algorithm, which performs the oblique projection at each iteration. But this alternative is expensive and consequently not very efficient to reduce the costs. Other

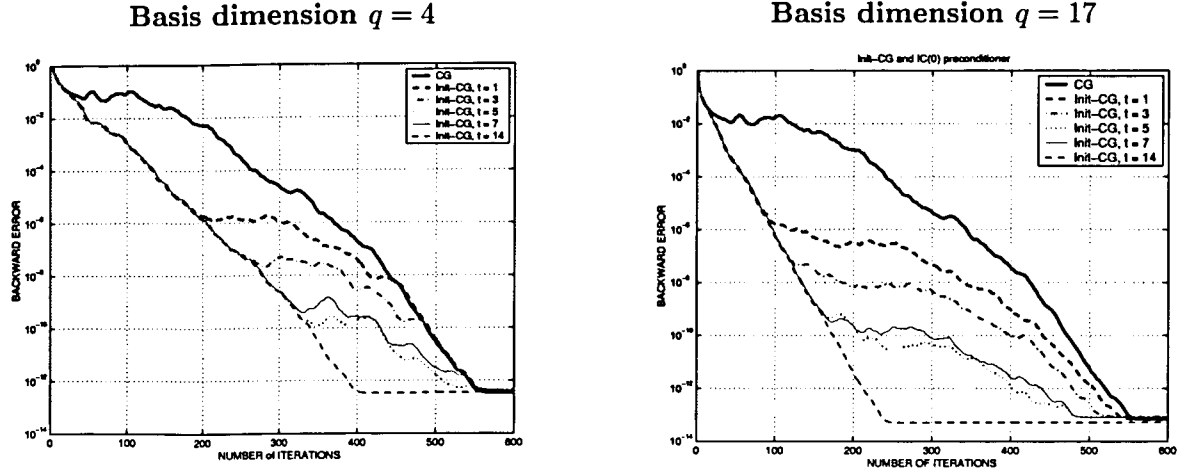


Figure 5.3: Effect of the accuracy of the spectral information on the convergence of INIT-CG.

remedy is to restart INIT-CG by updating the starting point $x^{(0)}$ in setting $x_{\text{restart}}^{(0)} = W\Delta^{-1}W^T r_{\text{restart}}$, where r_{restart} is the current residual.

In figure 5.4, we present the convergence history of INIT-CG in the current test problem, when the basis W is computed with one correct digit ($t = 1$) only. It is necessary to restart INIT-CG only twice to get a curve identical to the one obtained with a very accurate ($t = 14$) basis W (see figure 5.3). With $q = 4$, INIT-CG was restarted at iterations 200 and 300, and with $q = 17$ at iterations 80 and 180.

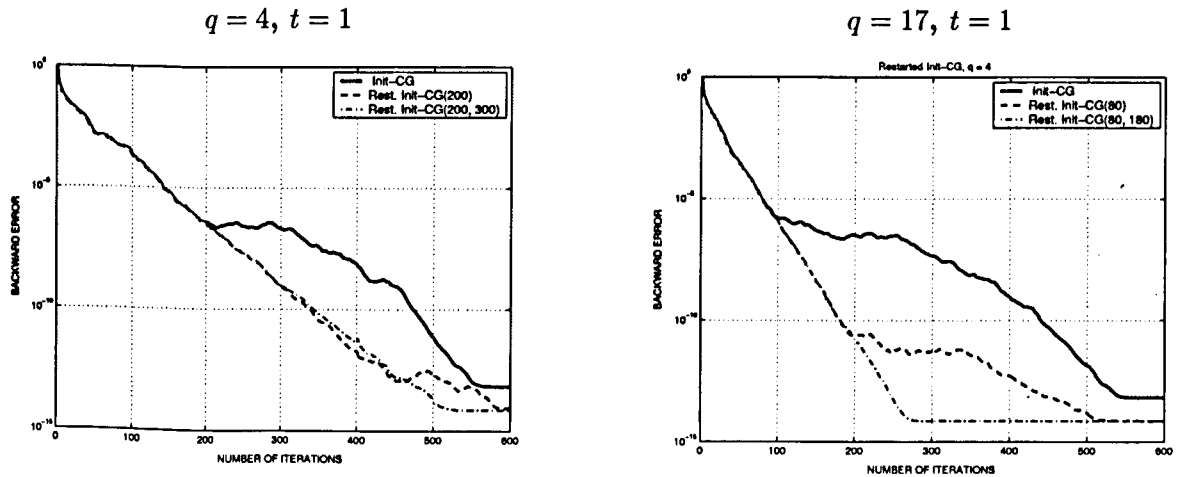


Figure 5.4: Convergence in number of iterations of the restarted INIT-CG.

If we want to reduce the backward error to a level of 10^{-8} , we need to restart once (see figure 5.4). In this case the restart implies one more projection, whose cost is given by (4.26). Figure 5.5, shows the histogram of the backward error *versus* the number of floating-point operations (flops) of the restarted INIT-CG and of the classical Conjugate Gradient. It is clear that the restarted strategy is a very cheap acceleration technique. The extra-costs are two obliques projections (initial and restarted) only, which is a low cost compared with the gains obtained.

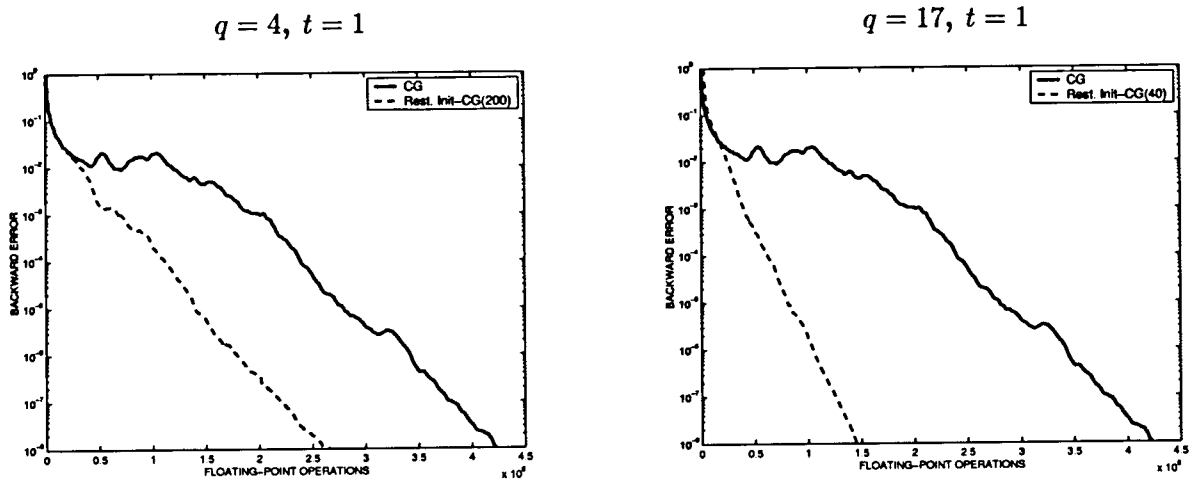


Figure 5.5: Convergence in floating points operations of the restarted INIT-CG.

5.5 Optimal dimension of the basis

The key parameter for the success of our approach is the dimension q of the basis of the *near*-invariant subspace associated with the smallest eigenvalues. There are two different ways to define it. The first one is to fix it *a priori* and stop the BlockCGSI when the basis W has reached the targeted dimension. This strategy cannot be used in a real problem where we do not know the distribution of the smallest eigenvalues. In this case, we can target instead a reduced condition number $\kappa = \lambda_{\max}/\mu$, for some given value μ , and try to compute a *near*-invariant subspace associated with all the eigenvalues smaller than μ . The desirable choice is that μ falls between two clusters. If μ is in the middle of a cluster the BlockCGSI algorithm must stop before the condition number has been reduced to this previously fixed value. For instance, if we want to reduce the condition number to $\kappa = 100$ we need to interpolate the 48 eigenvectors associated with all the eigenvalues smaller than $\mu = \lambda_{\max}/100$. As can we observe in figure 5.2, this value of μ is in the middle of a cluster

and no great benefits can be expected from the computation of a basis with this dimension. We will confirm this idea in terms of computational work.

The optimal dimension of the basis W will be the one that minimizes the total cost when solving all the systems, derived from the Poisson type equation (5.11), with our two-phase approach. This cost is given by

$$\text{Total cost} = \mathcal{C}_{BCGSI} + \mathcal{C}_{\text{Init-CG}} \times \text{NGits}, \quad (5.14)$$

where NGits is the number of global iteration (or time steps), i.e. the total number of systems to be solved. For instance, if we consider a simulation time $T = 1$ and $\Delta t = 0.01$ then $\text{NGits} = T/\Delta t = 100$. As before, the cost of pre-computing the spectral information with the BlockCGSI algorithm is given by $\mathcal{C}_{BCGSI}$ and the cost of solving one system with INIT-CG is given by $\mathcal{C}_{\text{Init-CG}}$. We investigate these costs in the following, analysing first the costs for pre-computing the basis W and after the solution costs.

5.5.1 Pre-computational cost

As we have seen in the previous chapter, the cost of pre-computing the spectral information depends on the dimension q of the basis W and on some working parameters in the BlockCGSI algorithm, as for instance the block size s , the filtering level ξ , and the cut-off value μ_f for the filtering. In our experiments with the current test problem, the value of μ_f is automatically set as $\mu_f = \mu$, and ξ is fixed to $1.0e - 10$. As we have seen in the experiments in chapter 4, the filtering level is important but does not need to be very small to reduce substantially the costs in BlockCGSI algorithm. In an efficient implementation of the BlockCGSI algorithm, Level 3 BLAS kernels can be incorporated in order to maximize the Megaflops rate, and the value of s can also be determined only on the basis of such computer aspects, keeping in mind that the *sliding window* technique adjusts the dimension of the basis W automatically.

Figure 5.6, displays the values of $\mathcal{C}_{BCGSI}$ for a block size s equal to 5, 10, 15, and 20, in function of final dimension q of the basis W . We can observe that lower pre-computational costs are obtained with larger block sizes, specially for high dimensions of q . The reasons for that are the *guard vector* effect and the practical changes that we introduced in Algorithm 4.4.2 in order to reduce the filtering costs when incorporating new vectors. As indicated in Algorithm 5.5.1, instead of generating new random vectors and filtering them (in step c of Algorithm 4.4.2), we simply inject a random linear combination of the filtered starting vectors $V^{(0)}$ generated in step 4 of Algorithm 4.2 (which are kept in memory for that purpose). With this practical simplification, we call the Chebyshev filtering routine only once and save a lot of extra work. The idea behind that, is that the starting vectors include already some information concerning all the eigenvalues in the range $]0, \mu[$, and to recover it we just need to redistribute this information over each of the newly incorporated vectors. In some cases, specially if we want an accurate spectral

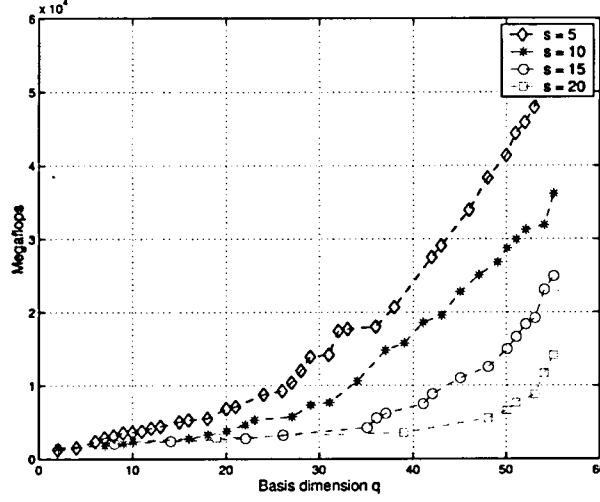


Figure 5.6: Costs ($\mathcal{C}_{BCGSI}$) of pre-computing the *near*-invariant subspace W with different block sizes s

information ($t = 3$ for instance), a breakdown can occur due to the *near*-collinearity of these new vectors relatively to the converged ones in $W^{(k)}$. Our experiments indicated that if we enforce the blockCG to do a minimum number of iterations (for instance $i_{min} = 4$), these new vectors will converge in the end to vectors different from the already converged ones.

ALGORITHM 5.5.1: INCORPORATE NEW VECTORS

Inputs: $\ell \in \mathbb{N}, M = R^T R \in \mathbb{R}^{n \times n}, V^{(0)} \in \mathbb{R}^{n \times s}, W^{(k)} \in \mathbb{R}^{n \times p}, V^{(k)} \in \mathbb{R}^{n \times (s-\ell)}$

Begin

- a) $Y = \text{RANDOM}(s, \ell)$
- b) $P = V^{(0)} Y$
- c) $P = Q \Gamma$ such that $Q^T M Q = I_{\ell \times \ell}$
- d) $P = Q - W^{(k)} W^{(k)T} M Q$
- e) $V^{(k)} = [V^{(k)} P]$

End

5.5.2 Solution cost

If we analyze now the behavior of the solution costs $\mathcal{C}_{\text{Init-CG}}$ as a function of the final dimension q , we expect a decrease of $\mathcal{C}_{\text{Init-CG}}$ as the value of q increases. On the left of figure 5.7, we plot the convergence of INIT-CG algorithm, in Megaflops, *versus* the

dimension q of the basis of the pre-computed *near*-invariant subspace. We observe at the beginning a great decrease of the convergence costs until q reaches approximately the dimension 20, and for larger dimensions of q the costs tend to stabilize. This occurs because, till q is lower than 20 we are incorporating the extremal eigenvalues (see figure 5.2) and after that, we are in the middle of a cluster. As the basis is enlarged from $q = 20$ to $q = 30$, the value of μ is shifted only from $6.63e - 03$ to $1.02e - 02$, which corresponds to a light reduction of the condition number from $\kappa = 2.54 + 02$ to $\kappa = 1.68e + 02$. Additionally, the costs of the initial and restarted oblique projections in the INIT-CG algorithm also contribute to maintain the solution costs $\mathcal{C}_{\text{Init-CG}}$ at the same level in this case. This does not mean that we cannot use the proposed two-phase approach with a dimension larger than $q = 20$. Indeed, the large number of systems to be solved allows to use a basis W with larger dimension. In the right of figure 5.7, we indicate the number of time steps (number of right-hand sides) after which the cost for pre-computing the spectral information is compensated due to gains obtained at each solve. The formula to derive this numbers is

$$\text{Amor. rhs} = \left\lfloor \frac{\mathcal{C}_{\text{BCGSI}}}{\mathcal{C}_{\text{Init-CG}} - \mathcal{C}_{\text{CG}}} \right\rfloor + 1,$$

where $\mathcal{C}_{\text{CG}}$ is the cost of classical CG algorithm, and $\mathcal{C}_{\text{BCGSI}}$ is the cost of pre-computing the spectral information with the BlockCGSI algorithm using a block size $s = 10$ (see figure 5.6). We can observe that if there are 100 systems to be solved ($\text{NGits} = 100$), the pre-computational costs are amortized for a basis dimension up to the critical value of $q = 50$. For larger number of times-steps (for instance $\text{NGits} = 1000$) this critical value of q can be even larger.

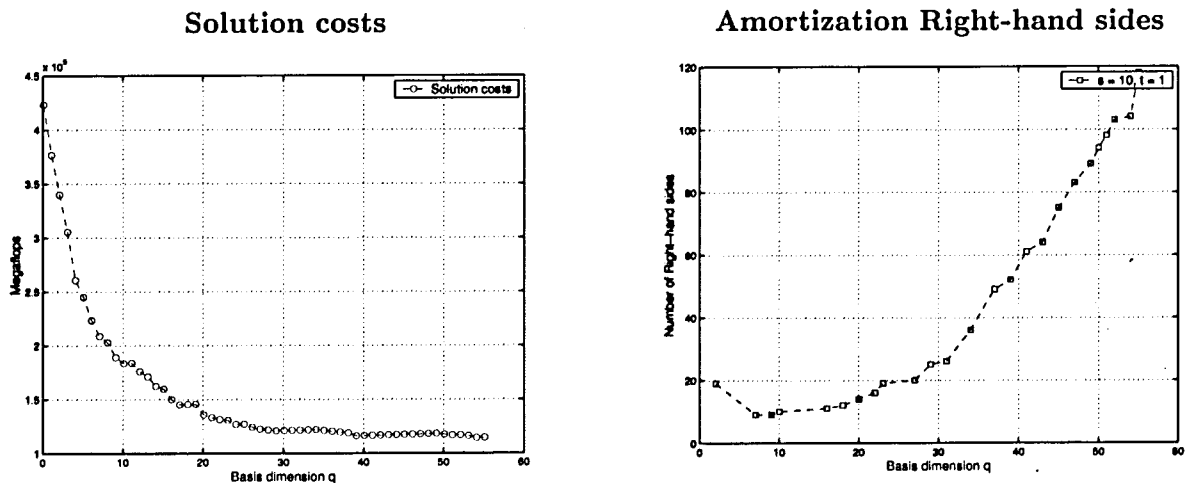


Figure 5.7: Solution costs and amortization right-hand sides of INIT-CG.

5.5.3 Minimizing the total cost function

Much more important than the critical values of q is the optimal value of q which minimizes the total cost function given by (5.14). As we have seen, the parameter **Total cost** is the addition of two other cost functions that are inversely proportional, namely C_{BCGSI} and $C_{Init-CG} \times \text{NGits}$. We have seen that the cost of pre-computing the spectral information C_{BCGSI} increases with q , while the solution costs $C_{Init-CG} \times \text{NGits}$ decrease as q increases. In figure 5.8, we plot all these two costs as well as the sum of two (the **Total cost**). The cost C_{BCGSI} was computed with block size $s = 10$, and the plot on the left corresponds to a simulation time of $T = 1$ where $\text{NGits} = T/\Delta t = 100$, and the plot on the right to $T = 10$ with $\text{NGits} = T/\Delta t = 1000$. The minimal value of **Total cost** occurs before $C_{BCGSI} \approx C_{Init-CG} \times \text{NGits}$ because the solution cost $C_{Init-CG} \times \text{NGits}$ decreases very slightly when q is greater than 20, and the cost C_{BCGSI} grows faster proportionally. The optimal value of q is near 20 when $\text{NGits} = 100$, and near 30 when $\text{NGits} = 1000$.

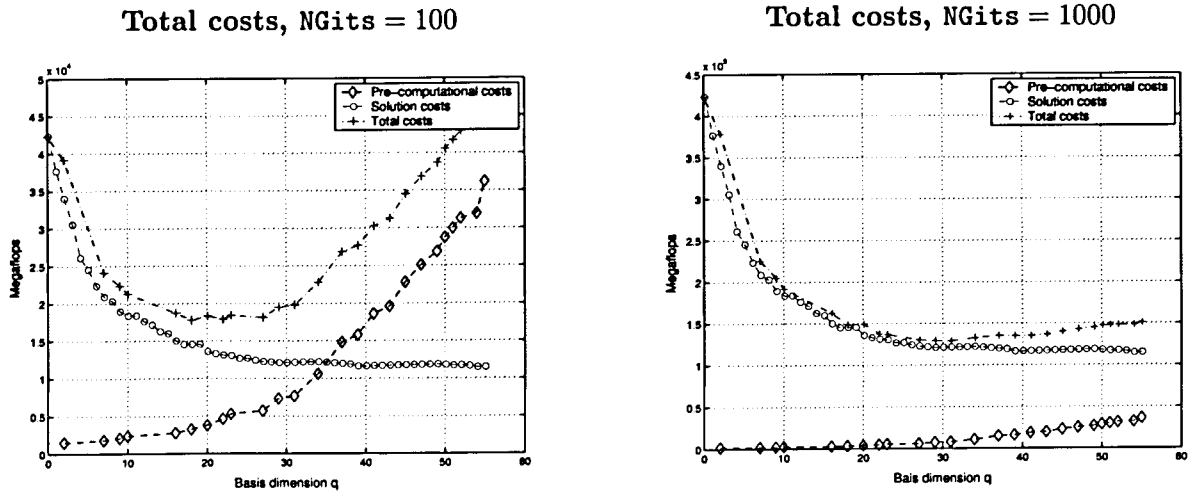


Figure 5.8: Total costs of the two-phase approach (block size $s = 10$ used for the pre-computation).

The optimal value of q confirms that it is better to stop the BlockCGSI algorithm when the Ritz values get very close to each other. There is no benefit in approximating all the 48 eigenvalues corresponding to the targeted condition number $\kappa = 100$, and the total cost increases if we continue the subspace iteration for values of q larger than 20. Even if the total number of systems to be solved NGits is large, as for instance 1000, there is no important reduction of the total cost for basis dimension larger than 20. As we can observe on the right of figure 5.8, the **Total cost** corresponding to $q = 20$ is nearly $1.5e + 05$ Mflops, and the value corresponding to $q = 30$ is nearly $1.3e + 05$ Mflops. We confirm the idea that if the spectrum is very clustered, the two phases accelerating strategy is more effective if we compute only the *near*-invariant subspace associated with

the extremal eigenvalues. Finally, we also insist on the memory requirements for keeping the basis W , that can become prohibitive in the case of very large matrices. This is also in favor for capturing the strictly necessary extremal eigenvalues.

5.6 Costs-benefits of the two-phase approach

In table 5.1, we illustrate the cost-benefits of this strategy for acceleration. We consider that the INIT-CG algorithm has converged when the backward error, computed by equation (4.27), is below 10^{-8} . In this case the classical Conjugate Gradient (INIT-CG with $q = 0$) performs 423 Mflops. As before, we indicate the number of floating point operations in Megaflops (Mflops) and the number of amortization right-hand sides by Amor. rhs. The pre-computational costs $\mathcal{C}_{BCGSI}$ are obtained with a block size of $s = 15$ in this case.

Table 5.1: Cost-benefit of the two-phase accelerating technique.

q	Spectral fact. $\mathcal{C}_{BCGSI}$	INIT-CG Mflops	Amor. rhs	Total cost	
				NGIts = 100	NGIts = 1000
0	–	423	–	42300	423000
5	2041	261	13	28141	263041
10	2377	189	11	21277	191377
15	2767	162	11	18967	164767
20	2767	145	10	17267	147767
25	3225	127	11	15925	130225
30	4242	121	15	16342	125242
35	4242	122	15	16442	126242
40	7473	116	25	19073	123473
45	11015	117	36	22715	128015
50	14972	118	50	26772	132972
55	24950	114	81	36350	138950

Firstly, we can observe that when the number of time steps NGIts = 100, the minimum value of Total cost, obtained with $q = 25$ is of 15925 Mflops. The optimal value of q is greater than in the previous section ($q = 20$) because the use of $s = 15$ instead of $s = 10$ reduces partly the costs for pre-computing the basis W . The same occurs when NGIts = 1000 where the optimal value of q is equal to 40 instead of 30. This indicate us that, if there is not computational restrictions, the BlockCGSI algorithm should be run with a block size s as large as it is possible. To analyse this point in more details on should perform a computational time analysis, which is the only way into account computer aspects like the memory accesses, parallelism,...

If we apply our accelerating technique to the current problem, and compute a basis W with optimal dimension ($q = 25$), 3225 Mflops are needed for the spectral pre-computation, out of which the INIT-CG achieves convergence in 127 Mflops, i.e. a reduction of 70%

compared with the work needed to solve one system with the classical Conjugate Gradient (423 Mflops). Consequently, the 3225 extra Mflops are paid back after 11 consecutive global iterations of the simulation code. And, in the case of $NGits = 100$, the value of **Total cost** is reduced from 42300 to 15925 Mflops, which corresponds to a reduction of 62% of the total amount of work needed to solve 100 consecutive linear systems. In the case of $NGits = 1000$, with a basis of size $q = 30$ is used, a similar computation shows that the **Total cost** is reduced from 423000 to 125242 Mflops, which is a reduction of order 70% over all the computational work needed to solve the 1000 systems.



Chapter 6

Closure

6.1 Conclusions

We have introduced a two-phase accelerating technique that reduces the total computational effort in simulation codes coming from several domains. In the first phase, a *near*-invariant subspace linked to the smallest eigenvalues is pre-computed by means of an inexact subspace iteration algorithm. The computation of this *near*-invariant subspace need not be very accurate for our purpose, which can contribute to minimize the total amount of work. In the second phase, this basis is exploited within the Conjugate Gradient algorithm to remove the effects of the smallest eigenvalues when solving iteratively the multiples systems, with varying right-hand sides, arising from the time dependent simulation.

Concerning the computation of the spectral information associated with the smallest eigenvalues in our linear systems, we have chosen an algorithm, called BlockCGSI, that combines the Subspace Iteration method with a stabilized version of the block Conjugate Gradient as an inner solver. The main reasons for this choice were to enable a precise control of the memory requirements, and the possibility of incorporating level 3 BLAS operations in the blockCG algorithm to get an efficient implementation on modern computers.

The initial experiments on the SIMPLE algorithm have shown that this strategy is viable, because the initial cost spent in the computation of the spectral information can be recovered by the reduction on the number of CG iterations in the following runs. The number of iterations can even be reduced up to 75%, and the extra cost needed to pre-compute the spectral information is payed back after a small number of global iterations in the SIMPLE algorithm.

After verifying the potential of the BlockCGSI algorithm in this context, our main objectives were to monitor in an appropriate way the two nested loops in this algorithm.

The first goal was to reduce the total amount of work in the first phase, when pre-computing the *near*-invariant subspace. The second goal was to be able to build an algorithm that can adapt automatically to the properties of the given matrices and compute a *near*-invariant subspace with some targeted accuracy. We thus proposed and validated experimentally a stopping criterion for the blockCG that was in agreement with the measure of convergence of the inverse iteration itself. It was observed that the stagnation level of the spectral error bound was directly linked to the choice of the threshold parameter used with this criterion to stop the blockCG iterations. In order to investigate this issue from a theoretical point of view, we have considered some of the ideas developed in other related studies on the inexact inverse iteration.

We further analyzed the BlockCGSI algorithm through an inner-outer iteration perspective, common to other inexact inverse iteration analysis, and we proposed to measure the residuals of the system through a Rigal-Gaches type of backward error. This measure enables the control of a bound on the absolute eigenvalue error in the inverse iteration itself, at the same time that the system is solved. This control is even more effective at the first inverse iteration if the starting vectors are previously filtered with Chebyshev polynomials. We also derived 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 with 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 initial set of vectors used to start the inverse iteration. The experiments indicated that Chebyshev filtering is useful to reduce both the total number of inverse and blockCG iterations, and consequently to reduce the total amount of work. This also had the benefit of putting our proposed monitoring in a more favorable situation even at very first inverse iterations.

Additionally, we have considered a form of dynamic adjustment of the dimension of the computed subspace in the BlockCGSI algorithm. This technique, that we called “*sliding window*”, is helpful to make the algorithm flexible and robust. Indeed, the dimension of the subspace to be computed can be enlarged during the course of the subspace iterations, and is not specifically linked to the working block size. This opened the way to efficient implementations of the algorithm, because the block size can be set in order to maximize the efficiency of Level 3 BLAS kernels for instance, while the algorithm adapts to the precise eigenvalue distribution on the fly.

This eigencomputation has a cost that depends on the dimension and on the accuracy of the computed invariant subspace. Once this information is obtained, we can use it to accelerate the Conjugate Gradient algorithm in the consecutive solutions of several linear systems with the same coefficient matrix and changing right-hand sides. The success of this two-phase approach implies that the initial cost can be recovered substantially by the

reduction of the number of CG iterations in the following runs.

In the second phase of the proposed accelerating technique, we focused our attention on two related issues that aim at accelerating the CG convergence with some available spectral information: (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 approach, also called SLRU-CG, appeared to be numerically more stable, achieving linear convergence, even when the pre-computed spectral information was obtained with poor accuracy. Nevertheless, it is also very expensive, in terms of computational work, compared to the first approach (also called INIT-CG).

In most of the experiments, the SLRU-CG is effective in reducing the total number of iterations performed by the CG, but it is not always effective in reducing the total amount of work. This occurs because the SLRU-CG uses the spectral information at each iteration, through a second level of preconditioning, and the benefits coming from the reduction of the number of iterations can be quickly overwhelmed by the extra work spent in the preconditioning. On the opposite, in the INIT-CG, the basis of the *near*-invariant subspace is just used once, at the beginning of the CG run. For this reason the reduction of the number of iterations is directly converted into a reduction of the total amount of work.

However, INIT-CG can loose rapidly linear convergence if the spectral information is not very accurate. When the backward error reaches some level proportional to the accuracy of the *near*-invariant subspace, the behavior of INIT-CG becomes similar again to that observed in the classical Conjugate Gradient without acceleration. Nevertheless, it is still possible to reduce the backward error under this level, while keeping the linear rate of convergence. To do so, one just needs to restart the INIT-CG algorithm when detecting that the linear rate of convergence is lost. The INIT-CG algorithm appeared then to be in general the right choice for improving the consecutive solution of linear systems in the second phase of the proposed acceleration technique.

The experiments showed that, if the spectrum is previously clustered, with the help for instance of a first level of preconditioning, the strategy can be very efficient in the reduction of the total cost when solving consecutive linear systems with changing right-hand sides. An efficient preconditioner clusters most of the eigenvalues near one, letting just a few of them near zero. In these cases, removing the effect of these eigenvalues can have a dramatic effect in the convergence of the CG which improves very much, and the extra work needed to compute the spectral information can be payed back after a small number of consecutive solutions.

Nevertheless, when the spectrum is not efficiently clustered by the preconditioner the key question of this strategy is the dimension q of the *near*-invariant subspace to be computed. Our experiments showed that the optimal dimension depends on a compromise to reach between the pre-computational costs and the solution costs. The cost of pre-computing the spectral information, that increases with the dimension q of the basis,

depends also on the block size s used in the BlockCGSI algorithm. On the one hand, the experiments indicate that larger block sizes can reduce these pre-computational costs. On the other hand, as the dimension of the *near*-invariant subspace increases, the solution costs decrease until q is larger than a certain value, after which these solution costs stagnate. The optimal dimension is thus the one that minimizes the sum of the two costs (pre-computing and solution) over all the systems to be solved. In our experiments, the minimum of this sum is reached for a dimension of the pre-computed basis close to the value of q after which these solution costs start to stagnate.

The stagnation of the solution costs occur because the convergence rate of CG is not sufficiently improved. This happens because the effect of all the extremal eigenvalues, separated from the main cluster, have been removed. As the remaining eigenvalues are very close from each other, deflating them we will just yield a very small reduction in the effective condition number that governs the convergence rate of the accelerated CG.

In consequence, we suggest a dynamic strategy to setup the dimension q of the *near*-invariant subspace associated with the smallest eigenvalues, without *a priori* knowledge of the spectrum. At the beginning, we target some reduced condition number $\kappa = \lambda_{\max}/\mu$, in agreement with the convergence rate of the CG, and request to approximate all the eigenvectors corresponding to the eigenvalues smaller than μ . At each iteration of the BlockCGSI algorithm, if this request is not verified, we compute the gaps between the approximated eigenvalues (Ritz values). As soon as the largest gap is below a certain pre-fixed tolerance, which means that we are in the middle of a cluster, we then stop the BlockCGSI algorithm.

6.2 Future work

Concerning future investigations, the very first and important task following naturally this work is to investigate the potential and limitations of the proposed two-phase approach on a large range of test problems, coming from different areas and with varying spectral characteristics. In the same range of ideas, it is also important to compare this technique with other existing techniques for eigencomputation, as for instance the routine in ARPACK Lehoucq et al. (1998), or some closely related approaches that have appeared very recently, see for instance in Golub et al. (2005), that share some of the ideas presented in this manuscript (the Chebyshev filtering, in particular). To complete this work, it might also be interesting to experiment and analyze the proposed monitoring strategy with respect to the various different theoretical or experimental analysis that have already been published (and summarized in section 3.3).

Other developments would be to try to incorporate multiple shifts within the inexact subspace iteration, as proposed for instance in Simoncini and Eldén (2002), to improve the first phase in this technique for the only purpose of computing some partial spectral

information associated with internal eigenvalues in a given spectrum. To do so, we must understand how to modify appropriately the blockCG inner solver, to address the case of indefinite matrices, and to analyze in parallel how to extend the proposed monitoring strategy in that respect.

In the case of large scale PDE numerical simulation (3D time dependent problems, for instance), it might be of interest to adapt and exploit this two-phase strategy in Domain Decomposition type of approaches, that can be viewed as a first layer of preconditioner, with embedded parallelism. To address this, particular efforts have to be brought on the code development and on the use of appropriate library routines to exploit efficiently all the features available on modern computers. The case of domain decomposition is very appealing indeed, because the Schur complement is almost never assembled explicitly, and one of the nice features of the proposed technique is that it requires only matrix vector products.

Finally, we plan also to investigate the case of non-linear iteration in which the Jacobian matrices do not keep constant properties in general, but may not change too rapidly. This has already received particular attention (see Loghin et al. (2005) and Giraud et al. (2005) for instance), in which some particular adaptive preconditioning strategies have been proposed to accelerate the convergence of the GMRES (Saad and Schultz, 1986) solver within the non-linear loop. In these studies, some low-rank update preconditioners are upgraded iteratively by stacking, in some different ways, Krylov information obtained at each run of the GMRES algorithm. We plan to investigate if we can exploit similar ideas within the inexact inverse subspace iteration to target SPD non-linear problems in the same way.

Bibliography

- M. Arioli. A stopping criterion for Conjugate Gradient algorithm in a finite element method framework. *Numerische Mathematike*, 97:1–24, 2004.
- M. Arioli, I. Duff, D. Ruiz, and M. Sadkane. Block Lanczos techniques for accelerating the block Cimmino method. *SIAM Journal on Scientific and Statistical Computing*, 16(6): 1478–1511, 1995.
- M. Arioli and D. Ruiz. Block conjugate gradient with subspace iteration for solving linear systems. In *Iterative Methods in Linear Algebra, Second IMACS Symposium on Iterative Methods in Linear Algebra*, pages 64–79, Blagoevgrad, Bulgaria, 1995.
- M. Arioli and D. Ruiz. A Chebyshev-based two-stage iterative method as an alternative to the direct solution of linear systems. Technical Report RAL-TR-2002-021, Rutherford Appleton Laboratory, 2002.
- C. Balsa. Resolução de sistemas de equações lineares, em mecânica dos fluidos computacional, por métodos iterativos não-estacionários. Tese de mestrado, FEUP, Porto, 2000.
- C. Balsa and J.L. Palma. Métodos iterativos do subspaço de krylov para a resolução de sistemas de equações algébricas em CFD. In J. I. Barbosa, editor, *VII Congresso de Mecânica Aplicada e Computational*, volume IV, pages 2017–2026, Evora, 2003.
- M. Bergmann. Analyse physique de la transition laminaire-turbulent 2D dans des écoulements cisailés a l'aide d'un code de Navier-Stokes en éléments finis. Rapport de stage de DEA, IMFT-ENSEEIH, Toulouse, France, 2001.
- J. Berns-Mueller, I. Graham, and A. Spence. Inexact inverse iteration for symmetric matrices. *Submitted to Linear Algebra and its Applications*, 2005.
- M. Braza. Analyse physique du comportement dynamique d'un écoulement externe, décollé, instationnaire en transition laminaire-turbulent. application: Cylindre circulaire. Thèse de docteur d'État, Institut National Polytechnique de Toulouse, France, 1986.
- B. Carpentieri, I. Duff, and L. Giraud. A class of spectral two-level preconditioners. *SIAM Journal on Scientific Computing*, 25(2):749–765, 2003.

- P. Concus, G. H. Golub, and D. O’Leary. A generalized Conjugate Gradient method for the numerical solution of elliptic partial differential equations. Technical report, Stanford University, 1976. STAN-CS-76-533.
- J. Dongarra, J. Ducroz, S. Hammarling, and I. Duff. A set of level 3 Basic Linear Algebra Subprograms. *ACM Transactions on Mathematical Software*, 16(1):1–28, 1990.
- J. Dongarra, J. Ducroz, S. Hammarling, and R. Hanson. An extended set of Fortran Basic Linear Algebra Subprograms. *ACM Transactions on Mathematical Software*, 14:1–17, 1988.
- J. Ferziger. *Numerical Methods for Engineering Application*. John Wiley & Sons, Inc, New York, 1998.
- J. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. Springer, 3rd edition, 2002.
- L. Giraud, S. Gratton, and E. Martin. Incremental spectral preconditioners for sequences of linear systems. Technical Report TR/PA/05/17, CERFACS, 2005.
- L. Giraud, D. Ruiz, and A. Touhami. A comparative study of iterative solvers exploiting spectral information for SPD systems. Technical Report RT/PA/04/40, CERFACS, 2004. Also TR/TLSE/04/03 at ENSEEIHT-IRIT and to appear in SIAM Journal on Scientific Computing.
- G. H. Golub, D. Ruiz, and A. Touhami. A hybrid approach combining Chebyshev filter and Conjugate Gradient for solving linear systems with multiple right-hand sides. *Submitted to the SIAM Journal on Matrix Analysis and Applications*, 2005.
- G. H. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore and London, 3rd edition, 1996.
- G. H. Golub and Q. Ye. Inexact inverse iteration for generalized eigenvalue problems. *BIT*, 40:671–684, 2000.
- L. A. Hageman and D. M. Young. *Applied Iterative Methods*. Academic Press, New York and London, 1981.
- M. R. Hestenes and E. Stiefel. Methods of Conjugate Gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.
- Z. Jia. A refined subspace iteration algorithm for large sparse eigenproblems. *IMACS Applied Numerical Mathematics*, 32:35–52, 2000.
- Yu-Ling Lai, Kun-Yi Lin, and Wen-Wei Lin. An inexact inverse iteration for large sparse eigenvalue problems. *Numerical Linear Algebra with Applications*, 4(5):425–437, 1997.

- R. Lehoucq, D. Sorensen, and C. Yang. *ARPACK users' guide: Solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, Philadelphia, 1998.
- D. Loghin, D. Ruiz, and A. Touhami. Adaptive preconditioners for nonlinear systems of equations. *To appear in Journal of Computational and Applied Mathematics*, 2005.
- G. Martinat. Analyse physique de la transition laminaire-turbulent sous l'effet de la rotation par un code en éléments finis. Rapport de stage de DEA, IMFT-ENSEEIH, Toulouse, France, 2003.
- J. J. McGuirk and J. L. Palma. The efficiency of alternative pressure-correction formulations for incompressible turbulent flow problems. *Computers & Fluids*, 22(1):77–87, 1993.
- Y. Notay. Combination of Jacobi-Davidson and Conjugate Gradients for the partial symmetric eigenproblem. *Numerical Linear Algebra with Applications*, 9:21–44, 2002.
- D. O'Leary. The block Conjugate Gradient algorithm and related methods. *Linear Algebra and its Applications*, 29:293–322, 1980.
- C. Paige and M. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal of Numerical Analysis*, 12:617–629, 1975.
- B. N. Parlett. *The Symmetric Eigenvalue Problem*. SIAM, Philadelphia, second edition, 1998.
- S. V. Patankar and B. B. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15:1787–1806, 1972.
- C. M. Rhie and W. L. Chow. Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA Journal*, 21:1525–1532, 1983.
- J. L. Rigal and J. Gaches. On the compatibility of a given solution with the data of a linear system. *Journal of the ACM*, 14(3):543–548, 1967.
- A. Ruhe and T. Wiberg. The method of Conjugate Gradients used in inverse iteration. *BIT*, 12:543–554, 1972.
- D. Ruiz. Résolution de grands systèmes linéaires creux non symétriques par une méthode itérative par blocs dans un environnement multiprocesseur. Thèse TH/PA/92/6, CERFACS, Toulouse, 1992.
- Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Halstead Press, New York, 1992.
- Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2nd edition, 2000.

- Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM - Journal Scientific and Statistical Computing*, 7: 856–869, 1986.
- Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'h. A deflated version of the Conjugate Gradient algorithm. *SIAM Journal on Scientific Computing*, 21:1909–1926, 2000.
- V. Simoncini and L. Eldén. Inexact Rayleigh quotient-type methods for eigenvalue computations. *BIT*, 42:159–182, 2002.
- G. Sleijpen and H. van der Vorst. A Jacobi-Davidson iteration method for linear eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 17:401–425, 1996.
- P. Smit and M.H.C. Paardekooper. The effects of inexact solvers in algorithms for symmetric eigenvalue problems. *Linear Algebra and its Applications*, 287:337–357, 1999.
- G. W. Stewart. *Matrix Algorithms. Volume II Eigensystems*. SIAM, Philadelphia, 2001.
- R. Underwood. An iterative block Lanczos method for the solution of large sparse symmetric eigenproblems. Phd thesis STAN-CS-75-496, Computer Science Department, Stanford University, California, 1975.
- A. van der Sluis and H. van der Vorst. The rate of convergence of Conjugate Gradients. *Numerische Mathematike*, 48(5):543–560, 1986.

Publications

Paper submitted to international journal

CARLOS BALSA, MICHEL DAYDÉ, JOSÉ L. PALMA, DANIEL RUIZ. Inexact Inverse Subspace Iteration to Exploit Partial Spectral Information. Submitted to *Numerical Linear Algebra with Applications*, also *Technical Report TR/TLSE/05/09* ENSEEIHT-IRIT, 2005.

International meetings with edited post-conference proceedings and scientific committee

CARLOS BALSA , JOSÉ L. PALMA, DANIEL RUIZ. Partial Spectral Information from Linear Systems to Speed-up Numerical Simulations in Computational Fluid Dynamics. *Lectures Notes on High Performance Computing for Computational Science. VECPAR'2004*, 703-719, Berlin Heidelberg, 2005.

CARLOS BALSA, MICHEL DAYDÉ, JOSÉ L. PALMA, DANIEL RUIZ. Monitoring the Block Conjugate Gradient Convergence within the Inexact Inverse Subspace Iteration. *6th International Conference on Parallel Processing and Applied Mathematics*, Institute of Computer and Information Science Czestochowa University of Technology, Poznan, Poland, 11/9/2005 (to appear in *Lectures Notes on High Performance Computing for Computational Science*).

CARLOS BALSA , MARIANNA BRAZA, MICHEL DAYDÉ, JOSÉ L. PALMA, DANIEL RUIZ. Improving the Numerical Simulation of an Airflow Problem with the BlockCGSI Algorithm. Submitted to VECPAR'2006, IMPA, Rio de Janeiro, 2006.

International meetings with scientific committee

CARLOS BALSA, MICHEL DAYDÉ, JOSÉ L. PALMA, DANIEL RUIZ. BlockCGSI Algorithm to Improve the Numerical Simulation of an Airflow Problem. *5th International Conference on Computer Science and Information Technologies*, The National Academy of Science of Armenia, Yerevan, Armenia, 19/9/2005.

CARLOS BALSA, MICHEL DAYDÉ, RONAN GUIVARC'H, JOSÉ L. PALMA, DANIEL RUIZ. Inexact Inverse Subspace Iteration to Exploit Partial Spectral Information from Linear Systems. *21st Biennial Conference on Numerical Analysis*, University of Dundee, Scotland, 28/6/2005.

National meeting with scientific committee

CARLOS BALSA, JOSÉ L. PALMA. Métodos Iterativos do Subespaço de Krylov para a Resolução de Sistemas de Equações Algébricas em CFD. *VII Congresso de Mecânica Aplicada e Computacional*, Universidade de Évora, Portugal, 14/4/2003.

International summer school

CARLOS BALSA, JOSÉ L. PALMA. Solving Linear Systems in Computational Fluid Dynamics, by Non-Stationary Iterative Methods. *Advanced School - Recent Developments in Large Scale Scientific Computing*, Faculdade de Engenharia da Universidade do Porto, Portugal, 1/7/2001.

Technical reports

CARLOS BALSA, MICHEL DAYDÉ, RONAN GUIVARC'H, JOSÉ L. PALMA, DANIEL RUIZ. Monitoring the Convergence of the Inexact Subspace Inverse Iteration to Compute Partial Spectral Information. *Technical Report TR/TLSE/05/02*, ENSEEIHT-IRIT, Toulouse, 2005.

CARLOS BALSA. Resolução de Sistemas de Equações Lineares, em Mecânica dos Fluidos Computacional, por Métodos Iterativos Não Estacionários. *Tese de Mestrado*, Faculdade de Engenharia da Universidade do Porto, 2000.



FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

BIBLIOTECA



0000084425