NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2017; **00**:1–20
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nla

Two-stage spectral preconditioners for iterative eigensolvers

L. Bergamaschi[1], and A. Martínez[2]

[1] *Department of Civil, Environmental and Architectural Engineering, University of Padua, via Marzolo 9, 35131 Padova, Italy*
[2] *Department of Mathematics, University of Padua, via Trieste 63, 35100 Padova, Italy*

SUMMARY

In this paper we present preconditioning techniques to accelerate the convergence of Krylov solvers at each step of an Inexact Newton's method for the computation of the leftmost eigenpairs of large and sparse symmetric positive definite matrices arising in large scale scientific computations. We propose a two-stage spectral preconditioning strategy: the first stage produces a very rough approximation of a number of the leftmost eigenpairs. The second stage (Inexact Newton) uses these approximations as starting vectors and also to construct the *tuned* preconditioner from an initial inverse approximation of the coefficient matrix, as proposed in [1, *Martínez, Numer. Lin. Alg. Appl., 2016*] in the framework of the Implicitly Restarted Lanczos method. The action of this spectral preconditioner results in clustering a number of the eigenvalues of the preconditioned matrices close to one. We also study the combination of this approach with a BFGS-style updating of the proposed spectral preconditioner as described in [2, *Bergamaschi, Martínez, Opt. Meth. Softw., 2015*]. Extensive numerical testing on a set of representative large SPD matrices gives evidence of the acceleration provided by these spectral preconditioners. Copyright © 2017 John Wiley & Sons, Ltd.

Received ...

Keywords: Eigenvalues, SPD matrix, Newton method, Spectral preconditioner, BFGS update, Incomplete Cholesky preconditioner.

AMS subject classification: 65F05, 65F15, 65H17

1. INTRODUCTION

The computation of the $m \ll n$ leftmost eigenpairs of large and sparse symmetric positive definite (SPD) $n \times n$ matrix A is a common task in many scientific applications. Typical examples are offered by the vibrational analysis of mechanical structures [3], and the electronic structure calculations [4]. Computation of a few eigenpairs is also crucial in the approximation of the generalized inverse of the graph Laplacian [5, 6].

Recently in [2] an efficiently preconditioned Newton method (DACG-Newton) has been developed which has proven to display comparable performances against the well known Jacobi-Davidson (JD) method [7] and outperforms the Implicitly Restarted Lanczos method with optimal tuned preconditioning [1] if a moderate number of eigenpairs is being sought.

The idea of spectral preconditioners to accelerate linear system solvers has been described in several papers such as [8, 9, 10] and, more recently [11], where the authors start with an initial preconditioner P_0 and use an approximation of a few eigenvectors of the preconditioned matrix to update P_0 with a low-rank matrix. In all these papers it is shown that some of the smallest

*Correspondence to: Luca Bergamaschi, Department of Civil, Environmental and Architectural Engineering, University of Padua, via Marzolo 9, 35131, Padova, Italy. E-mail: `luca.bergamaschi@unipd.it`.

eigenvalues of the new preconditioned matrix are incremented by 1 with an obvious reduction of the condition number.

In this paper we propose and develop a new preconditioning strategy for accelerating the second stage of the DACG-Newton method, namely the Newton iteration in the unit sphere [12, 2] also referred to as Newton-Grassmann method. The idea is to use the initial DACG approximation not only as an initial eigenvector guess for the Newton phase, but also to construct a spectral preconditioner for the efficient solution of the correction equation:

$$\begin{aligned} J_k \boldsymbol{u}_k &= -\boldsymbol{r}_k, \qquad \text{where} \\ J_k &= (I - \boldsymbol{u}_k \boldsymbol{u}_k^\top)(A - \theta_k I)(I - \boldsymbol{u}_k \boldsymbol{u}_k^\top), \quad \boldsymbol{r}_k = -(A\boldsymbol{u}_k - \theta_k \boldsymbol{u}_k), \quad \theta_k = \frac{\boldsymbol{u}_k^\top A \boldsymbol{u}_k}{\boldsymbol{u}_k^\top \boldsymbol{u}_k} \end{aligned} \tag{1}$$

to be solved at each step of this projected Newton method. As the preconditioner for equation (1) we propose to use a tuned preconditioner $\mathcal{P}$ ([1]) i.e. a preconditioner satisfying $\mathcal{P}AV_m = V_m$, where V_m is a rectangular matrix containing an approximation of the m leftmost eigenvectors as columns. We include a theoretical study of the spectral properties of $\mathcal{P}J_k$. It is found that, in the computation of a generic eigenpair $(\lambda_j, \boldsymbol{v}_j), j = 1, \ldots, m$, the approximate eigenvectors $\tilde{\boldsymbol{v}}_{j+1}, \ldots, \tilde{\boldsymbol{v}}_m$ provided by DACG are also eigenvectors of $\mathcal{P}J_k$ corresponding to eigenvalues close to one, unless the relative separation between the eigenvalue being sought and the next higher ones is very small.

This spectral preconditioner can be successfully combined with a low-rank update of preconditioners of BFGS type studied in [13, 2] to obtain a very efficient acceleration of the Newton method to compute a small to moderate number of the leftmost eigenpairs of large SPD matrices.

We test the preconditioned DACG-Newton eigensolver onto a number of medium to large-size SPD matrices arising from Finite Element discretization of PDEs arising from groundwater flow models in porous media, geomechanical processes in reservoirs, financial modeling and Laplacian of graphs. The numerical results show the significant improvement provided by this spectral preconditioner on all test problems.

The outline of the paper is as follows: in Section 2 we briefly recall the DACG-Newton method; Section 3 is devoted to the definition of the spectral preconditioner designed to accelerate the convergence of the Newton-Grassmann method and to characterize some of the eigenvalues of the preconditioned matrices. In Section 4 we discuss some implementation issues and present the main algorithms while in Section 5 we briefly recall the BFGS preconditioner together with its theoretical properties. Section 6 reports extensive numerical experiments onto matrices of large size arising from various realistic applications, also including a comparison of the proposed preconditioned method with the Jacobi-Davidson method. Section 7 draws the conclusions.

2. THE DACG-NEWTON METHOD FOR THE LEFTMOST EIGENPAIRS

Consider a symmetric positive definite (SPD) matrix A and denote as

$$\lambda_1 < \lambda_2 < \cdots < \lambda_m < \ldots < \lambda_n$$

its eigenvalues and

$$\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_m, \ldots, \boldsymbol{v}_n$$

the corresponding (normalized) eigenvectors.

Computation of the leftmost eigenpair of A can be recast as the following non linear problem

$$A\boldsymbol{x} - q(\boldsymbol{x})\boldsymbol{x} = 0, \qquad ||x|| = 1, \tag{2}$$

being $q(\boldsymbol{x}) = \boldsymbol{x}^\top A\boldsymbol{x}$ the Rayleigh quotient. This non linear system can be solved by projecting the Jacobian of the above nonlinear function in the space orthogonal to the current iterate $\boldsymbol{u}_k$ giving raise to the following Newton-Grassmann method

Repeat until convergence:

$$\begin{array}{rrcll} & \text{Set } \boldsymbol{u}_0 \text{ as an initial approximation of } \boldsymbol{v}_1, & & k=0 & \\ \text{solve approximately} & J_k \boldsymbol{s}_k & = & -\boldsymbol{r}_k \quad \text{for } \boldsymbol{s}_k \perp \boldsymbol{u}_k & (3) \\ \text{set} & \boldsymbol{u}_{k+1} & = & \dfrac{\boldsymbol{u}_k + \boldsymbol{s}_k}{\|\boldsymbol{u}_k + \boldsymbol{s}_k\|}, \; k = k+1 & (4) \end{array}$$

where

$$J_k = (I - \boldsymbol{u}_k \boldsymbol{u}_k^\top)(A - \theta_k I)(I - \boldsymbol{u}_k \boldsymbol{u}_k^\top) \qquad \text{and} \qquad \boldsymbol{r}_k = A\boldsymbol{u}_k - \theta_k \boldsymbol{u}_k, \qquad \theta_k = \boldsymbol{u}_k^\top A \boldsymbol{u}_k.$$

Note that the above iteration is the basis of the well-known Jacobi-Davidson method which combines the projected Newton's iteration (also called the correction equation) with a Rayleigh-Ritz step.

A crucial issue in the efficiency of the Newton approach for eigenvalue computation is represented by the appropriate choice of the initial guess. In [2] this task is accomplished by performing a few preliminary iterations of the DACG eigenvalue solver [14, 15], in order to start the Newton iteration 'sufficiently' close to the exact eigenvector. The DACG method has proven very robust, and not particularly sensitive to the initial vector, in the computation of a few eigenpairs of large SPD matrices. We note that the choice of DACG as the initial solver is not mandatory. Alternatively, other gradient methods such as e.g. the LOBPCG method [16, 17] could be successfully employed. However, we can not expect important differences in the performances as the two methods (DACG, LOBPCG) behave similarly, as also experimentally showed in [18] for the eigensolution of very large matrices in a parallel environment.

2.1. *PCG solution of the correction equation*

As a Krylov subspace solver for the correction equation we chose the Preconditioned Conjugate gradient (PCG) method since the Jacobian J_k has been shown to be SPD in the subspace orthogonal to $\boldsymbol{u}_k$. Regarding the implementation of PCG, we mainly refer to the work [19], where the author shows that it is possible to solve the linear system in the subspace orthogonal to $\boldsymbol{u}_k$ and hence the projection step needed in the application of J_k can be skipped. Moreover, we adopted the exit strategy for the linear system solution described in the above paper, which allows for stopping the PCG iteration, in addition to the classical exit test based on a tolerance on the relative residual and on the maximum number of iterations (usually set to $20 \div 30$), whenever the l-th PCG iterate $\boldsymbol{x}_l$ satisfies

$$\|\boldsymbol{r}_{k,l}\| = \|A\boldsymbol{x}_l - q(\boldsymbol{x}_l)\boldsymbol{x}_l\| < \varepsilon q(\boldsymbol{x}_l) \tag{5}$$

or when the decrease of $\|\boldsymbol{r}_{k,l}\|$ is slower than the decrease of the residual of the linear system at step l because in this case further iterating does not improve the accuracy of the eigenvector. Note that this dynamic exit strategy implicitly defines an Inexact Newton method since the correction equation is not solved "exactly" i.e. up to machine precision. We have implemented the PCG method as described in Algorithm 5.1 of [19] with the obvious difference in the application of the preconditioner which will be described in Section 3.

2.2. *Computing several eigenpairs*

When seeking an eigenvalue different from λ_1, say λ_j, the Jacobian matrix changes as

$$J_k^{(j)} = \left(I - Q_k^{(j)} Q_k^{(j)\top}\right)(A - \theta_k^{(j)} I)\left(I - Q_k^{(j)} Q_k^{(j)\top}\right) \tag{6}$$

where $Q_k^{(j)} = [\boldsymbol{v}_1 \, \boldsymbol{v}_2 \ldots \boldsymbol{v}_{j-1} \, \boldsymbol{u}_k]$ is the matrix whose first $j-1$ columns are the previously computed eigenvectors. In the above expression as well as in the sequel the index j will denote the eigenvalue level while superscript k will refer to the iteration index during the Newton process.

Every preconditioner of choice $\widehat{P}_k$ used to accelerate the correction equation $J_k^{(j)} \boldsymbol{s}_k = -\boldsymbol{r}_k$ must be projected in the subspace orthogonal to span$\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{j-1}, \boldsymbol{u}_k\}$, namely

$$P_k = \left(I - Q_k^{(j)} Q_k^{(j)^\top}\right) \widehat{P}_k \left(I - Q_k^{(j)} Q_k^{(j)^\top}\right). \tag{7}$$

3. ACCELERATION BY SPECTRAL PRECONDITIONERS

A class of spectral preconditioners is defined in [8] which is based on the knowledge of a few leftmost eigenpairs of the preconditioned matrix P_0A, being P_0 and A both SPD matrices. If we denote as V_m the rectangular matrix whose columns are the eigenvectors of the preconditioned matrix P_0A a spectral preconditioner is defined as

$$P = P_0 + V_m (V_m^\top A V_m)^{-1} V_m^\top.$$

It can be proved that the new preconditioned matrix PA has $\lambda_1 + 1, \ldots, \lambda_m + 1$ as eigenvalues, being $\lambda_1, \ldots, \lambda_m$ the smallest eigenvalues of matrix P_0A.

In the framework of the DACG-Newton method, we take advantage of the knowledge, after the DACG initial iteration, of a number of approximated eigenvectors of A instead of those of P_0A. We propose a new preconditioning strategy for accelerating the second stage of the DACG-Newton method, namely the Newton iteration, by using the initial DACG approximation not only as an initial eigenvector guess for the Newton phase, but also to update a given approximate inverse of the coefficient matrix A in the solution of the correction equation

$$J_k^{(j)} \boldsymbol{s}_k = -\left(A\boldsymbol{u}_k - q(\boldsymbol{u}_k)\boldsymbol{u}_k\right).$$

Denoted as $\widehat{P}_0$ an initial approximate inverse of A, we assume that the DACG method has provided the m leftmost eigenpairs (to a low relative accuracy specified by parameter τ) satisfying

$$A\tilde{\boldsymbol{v}}_j = \lambda_j \tilde{\boldsymbol{v}}_j + \mathbf{res}_j, \qquad \|\mathbf{res}_j\| \le \tau \lambda_j,\ j = 1, \ldots, m, \tag{8}$$

where we have neglected the error in the eigenvalue computation which is $O(\tau^2)$ (see [14]). Then, for a generic eigenvalue λ_j $(j < m)$ we define the following tuned preconditioner, which will be kept constant throughout the Newton iterations, to accurately compute the j-th eigenpair:

$$\widehat{P}_j = \widehat{P}_0 - W\left(W^\top A V_j\right)^{-1} W^\top, \qquad \text{with} \qquad W = \widehat{P}_0 A V_j - V_j \tag{9}$$

where

$$V_j = [\tilde{\boldsymbol{v}}_{j+1} \ldots \tilde{\boldsymbol{v}}_m]. \tag{10}$$

This tuned preconditioner is the inverse representation of the block counterpart of the preconditioner first proposed in [20] in the framework of the inverse iteration and the inexact Rayleigh quotient iteration. In [1] this preconditioner has been proposed to accelerate the inner linear system during the Lanczos' process, moreover some conditions are discussed under which the preconditioner $\widehat{P}_j$ is well defined and also SPD. A direct computation shows that $\widehat{P}_j$ is a tuned preconditioner ([21]) i.e. satisfies:

$$\widehat{P}_j A V_j = V_j, \tag{11}$$

irrespective of the error introduced by the computation of V_j. Relation (11) implies that the preconditioned matrix $\widehat{P}_j A$ has the eigenvalue 1 with at least multiplicity $m - j$.

When $\widehat{P}_j$ is used to accelerate the Newton iteration, it must be projected in the space orthogonal to the previously computed eigenvectors yielding:

$$P_j = \left(I - Q_k^{(j)} Q_k^{(j)^\top}\right) \widehat{P}_j \left(I - Q_k^{(j)} Q_k^{(j)^\top}\right)$$

hence the preconditioned matrix reads:

$$P_j J_k^{(j)} = \left(I - Q_k^{(j)} Q_k^{(j)\top}\right) \widehat{P}_j \left(I - Q_k^{(j)} Q_k^{(j)\top}\right) (A - \theta_k^{(j)} I) \left(I - Q_k^{(j)} Q_k^{(j)\top}\right).$$

The next two Lemmas 3.1, 3.2 and Theorem 3.1 will characterize the eigenvalues of the preconditioned matrix $P_j J_k^{(j)}$. The main theoretical results will be developed under the following:

Assumptions and notation

- **Notation**. The indices j, s are such that $1 \le j < m$ and $j < s < m$. Define the inverse of the relative separation between consecutive eigenvalues as

$$\xi_j = \frac{\lambda_{j+1}}{\lambda_{j+1} - \lambda_j}. \tag{12}$$

  In the sequel with $\widehat{P}_j$ we will denote the non-projected preconditioner, while the final projected preconditioner is referred to as P_j.
- **Assumptions**:

  Iteration index k is such that:

$$\theta_k^{(j)} - \lambda_j < \lambda_{j+1} - \theta_k^{(j)}, \tag{A.1}$$

  which implies that

$$\lambda_s - \theta_k^{(j)} > \lambda_s - \frac{\lambda_{j+1} + \lambda_j}{2} \ge \frac{\lambda_{j+1} - \lambda_j}{2}. \tag{A.2}$$

  We also assume that every residual of the Newton process is smaller than the residual of the DACG method and therefore satisfies:

$$\|\boldsymbol{r}_k^{(j)}\| \le \|\mathbf{res}_j\| \le \tau \lambda_j. \tag{A.3}$$

- From now on we will omit for simplicity super and sub-scripts in matrix $Q \equiv Q_k^{(j)}$ and in the scalar $\theta \equiv \theta_k^{(j)}$.

We begin with Lemma 3.1 which characterizes some of the eigenvalues of $\widehat{P}_j(A - \theta I)$:

Lemma 3.1
Let matrix V_j be defined as in (10), $\widehat{P}_j$ a tuned preconditioner satisfying condition (11), then each column of V_j i.e. $\tilde{\boldsymbol{v}}_s, s = j+1, \dots, m$, is an approximate eigenvector of $\widehat{P}_j(A - \theta I)$ corresponding to the approximate eigenvalue $1 - \dfrac{\theta}{\lambda_s} \approx 1 - \dfrac{\lambda_j}{\lambda_s}$. In particular the following relation holds:

$$\widehat{P}_j (A - \theta I) \tilde{\boldsymbol{v}}_s = \left(1 - \frac{\theta}{\lambda_s}\right) \tilde{\boldsymbol{v}}_s + \boldsymbol{e}_1, \qquad \text{with} \qquad \|\boldsymbol{e}_1\| \le \tau \lambda_{j+1} \|\widehat{P}_j\|.$$

Proof
Let $s \in [j+1, m]$, then from (8) it follows that $\tilde{\boldsymbol{v}}_s = \lambda_s^{-1} A \tilde{\boldsymbol{v}}_s - \lambda_s^{-1} \mathbf{res}_s$. Then

$$\begin{aligned}
\widehat{P}_j (A - \theta I) \tilde{\boldsymbol{v}}_s &= \widehat{P}_j A \tilde{\boldsymbol{v}}_s - \theta \widehat{P}_j \tilde{\boldsymbol{v}}_s \\
&= \tilde{\boldsymbol{v}}_s - \theta \widehat{P}_j \left(\lambda_s^{-1} A \tilde{\boldsymbol{v}}_s - \mathbf{res}_s \lambda_s^{-1}\right) \\
&= \tilde{\boldsymbol{v}}_s - \frac{\theta}{\lambda_s} \tilde{\boldsymbol{v}}_s + \frac{\theta}{\lambda_s} \widehat{P}_j \mathbf{res}_s \\
&= \left(1 - \frac{\theta}{\lambda_s}\right) \tilde{\boldsymbol{v}}_s + \boldsymbol{e}_1.
\end{aligned}$$

with $\|\boldsymbol{e}_1\| \le \tau \dfrac{\theta}{\lambda_s} \lambda_s \|\widehat{P}_j\| = \tau \theta \|\widehat{P}_j\| \le \tau \lambda_{j+1} \|\widehat{P}_j\|$. □

We have just proved that the preconditioned matrix $\widehat{P}_j(A-\theta I)$ has the following $m-j$ approximate eigenvalues

$$1-\frac{\lambda_j}{\lambda_{j+1}}, 1-\frac{\lambda_j}{\lambda_{j+2}}, \dots, 1-\frac{\lambda_j}{\lambda_m}.$$

In the next Lemma we will state and prove some technical facts which will be useful in the proof of Theorem 3.1 which will establish the main result of this Section.

Lemma 3.2
Recalling the definition of ξ_j (12), we have

$$\begin{aligned} \|Q^\top \tilde{\boldsymbol{v}}_s\| &\leq 5\xi_j\tau. & (13)\\ \|(A-\theta I)Q\| &\leq \lambda_{j+1}(1+\tau) & (14)\\ \|Q^\top(A-\theta I)Q\| &< \lambda_{j+1}. & (15)\end{aligned}$$

Proof
Define $\tilde{Q} = [\boldsymbol{v}_1\ \boldsymbol{v}_2\ \dots\ \boldsymbol{v}_{j-1}]$ and $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_{j-1})$. We start by proving (13).

$$Q^\top \tilde{\boldsymbol{v}}_s = \begin{bmatrix} \tilde{Q}^\top \\ \boldsymbol{u}_k^\top \end{bmatrix} \tilde{\boldsymbol{v}}_s = \begin{bmatrix} \tilde{Q}^\top \tilde{\boldsymbol{v}}_s \\ \boldsymbol{u}_k^\top \tilde{\boldsymbol{v}}_s \end{bmatrix}.$$

Now

$$\tilde{Q}^\top \tilde{\boldsymbol{v}}_s = \ (\text{by } (8))\ = \lambda_s^{-1}\tilde{Q}^\top A\tilde{\boldsymbol{v}}_s - \tilde{Q}^\top \mathbf{res}_s\lambda_s^{-1} = \lambda_s^{-1}\Lambda\tilde{Q}^\top\tilde{\boldsymbol{v}}_s - \tilde{Q}^\top\mathbf{res}_s\lambda_s^{-1}$$

from which

$$\|\tilde{Q}^\top\tilde{\boldsymbol{v}}_s\| \leq \frac{\lambda_{j-1}}{\lambda_s}\|\tilde{Q}^\top\tilde{\boldsymbol{v}}_s\| + \|\mathbf{res}_s\|\lambda_s^{-1},$$

and hence, using the fact that the function $f(t) = \dfrac{t}{t-\lambda_{j-1}}$ is decreasing in $(\lambda_{j-1}, +\infty)$,

$$\|\tilde{Q}^\top\tilde{\boldsymbol{v}}_s\| \leq \frac{\lambda_s}{\lambda_s-\lambda_{j-1}}\tau \leq \frac{\lambda_{j+1}}{\lambda_{j+1}-\lambda_{j-1}}\tau \leq \frac{\lambda_{j+1}}{\lambda_{j+1}-\lambda_j}\tau = \xi_j\tau. \tag{16}$$

Analogously

$$\boldsymbol{u}_k^\top\tilde{\boldsymbol{v}}_s = \ (\text{by } (8))\ = \lambda_s^{-1}\boldsymbol{u}_k^\top A\tilde{\boldsymbol{v}}_s - \boldsymbol{u}_k^\top\mathbf{res}_s\lambda_s^{-1} = \lambda_s^{-1}(\theta\boldsymbol{u}_k^\top + \boldsymbol{r}_k^\top)\tilde{\boldsymbol{v}}_s - \boldsymbol{u}_k^\top\mathbf{res}_s\lambda_s^{-1}$$

from which

$$\boldsymbol{u}_k^\top\tilde{\boldsymbol{v}}_s = \frac{\boldsymbol{r}_k^\top\tilde{\boldsymbol{v}}_s - \boldsymbol{u}_k^\top\mathbf{res}_s}{\lambda_s-\theta}.$$

Now using the fact that the function $g(t) = \dfrac{\lambda_j+t}{t-\theta}$ is decreasing in $(\theta, +\infty)$ and applying (A.3) and (A.2) we can write

$$|\boldsymbol{u}_k^\top\tilde{\boldsymbol{v}}_s| \leq \frac{\lambda_j+\lambda_s}{\lambda_s-\theta}\tau \leq \frac{\lambda_j+\lambda_{j+1}}{\lambda_{j+1}-\theta}\tau \leq 2\frac{\lambda_j+\lambda_{j+1}}{\lambda_{j+1}-\lambda_j}\tau \leq 4\frac{\lambda_{j+1}}{\lambda_{j+1}-\lambda_j}\tau = 4\xi_j\tau. \tag{17}$$

Combining (16) and (17) we finally obtain

$$\|Q^\top\tilde{\boldsymbol{v}}_s\| \leq \|\tilde{Q}^\top\tilde{\boldsymbol{v}}_s\| + |\boldsymbol{u}_k^\top\tilde{\boldsymbol{v}}_s| \leq 5\xi_j\tau.$$

To prove (14) we start from the definition of Q and write $(A-\theta I)Q = \begin{bmatrix}\tilde{Q}(\Lambda-\theta I) & \boldsymbol{r}_k\end{bmatrix}$. Then,

$$\|(A-\theta I)Q\| \leq |\lambda_1-\theta| + \|\boldsymbol{r}_k\| < \lambda_{j+1} + \lambda_j\tau < \lambda_{j+1}(1+\tau).$$

Moreover

$$Q^\top(A-\theta I)Q = \begin{bmatrix}\Lambda-\theta I & 0 \\ 0 & 0\end{bmatrix} \tag{18}$$

since $\boldsymbol{u}_k$ is orthogonal to $\boldsymbol{r}_k$ and therefore $Q^\top\boldsymbol{r}_k = 0$.
Hence $\|Q^\top(A-\theta I)Q)\| \leq \|\Lambda-\theta I\| \leq |\lambda_1-\theta| < \lambda_{j+1}$ and also (15) is proved. □

Theorem 3.1
Let matrix V_j be defined as in (10), $\widehat{P}_j$ a tuned preconditioner satisfying condition (11), then each column of V_j i.e. $\tilde{\boldsymbol{v}}_s, s = j+1, \ldots, m$, is an approximate eigenvector of $P_j J_k^{(j)}$ corresponding to the approximate eigenvalue $1 - \dfrac{\theta}{\lambda_s} \approx 1 - \dfrac{\lambda_j}{\lambda_s}$. In particular the following relation holds:

$$P_j J_k^{(j)} \tilde{\boldsymbol{v}}_s = \left(1 - \frac{\theta}{\lambda_s}\right) \tilde{\boldsymbol{v}}_s + \mathbf{err} \tag{19}$$

with $\|\mathbf{err}\| \le \tau\, C$, and $C \equiv C(\tau, \|\widehat{P}_j\|, \lambda_j, \lambda_{j+1})$.

Proof

$$\begin{aligned} P_j J_k^{(j)} \tilde{\boldsymbol{v}}_s &= (I - QQ^\top)\widehat{P}_j(I - QQ^\top)(A - \theta I)(I - QQ^\top)\tilde{\boldsymbol{v}}_s = \\ &= \widehat{P}_j(A - \theta I)\tilde{\boldsymbol{v}}_s - QQ^\top \widehat{P}_j(A - \theta I)\tilde{\boldsymbol{v}}_s + (I - QQ^\top)\widehat{P}_j \boldsymbol{z}, \end{aligned} \tag{20}$$

where we have set

$$\boldsymbol{z} = \underbrace{-QQ^\top(A - \theta I)\tilde{\boldsymbol{v}}_s}_{\boldsymbol{z}_1} - \underbrace{(A - \theta I)QQ^\top \tilde{\boldsymbol{v}}_s}_{\boldsymbol{z}_2} + \underbrace{QQ^\top(A - \theta I)QQ^\top \tilde{\boldsymbol{v}}_s}_{\boldsymbol{z}_3} .$$

Recalling Lemma 3.1 we can rewrite (20) as

$$P_j J_k^{(j)} \tilde{\boldsymbol{v}}_s = \left(1 - \frac{\theta}{\lambda_s}\right) \tilde{\boldsymbol{v}}_s + \boldsymbol{e}_1 - \boldsymbol{e}_2 + \boldsymbol{e}_3, \tag{21}$$

with $\boldsymbol{e}_2 = QQ^\top \widehat{P}_j(A - \theta I)\tilde{\boldsymbol{v}}_s,\ \boldsymbol{e}_3 = (I - QQ^\top)\widehat{P}_j \boldsymbol{z}$.
Now from Lemma 3.1 and Lemma 3.2 it follows that

$$\boldsymbol{e}_2 = QQ^\top \widehat{P}_j(A - \theta I)\tilde{\boldsymbol{v}}_s = QQ^\top((1 - \theta\lambda_s^{-1})\tilde{\boldsymbol{v}}_s + \boldsymbol{e}_1) = Q((1 - \theta\lambda_s^{-1})Q^\top \tilde{\boldsymbol{v}}_s + Q^\top \boldsymbol{e}_1)$$

so that

$$\|\boldsymbol{e}_2\| \le (5\xi_j + \lambda_{j+1}\|\widehat{P}_j\|)\tau.$$

We now bound $\|\boldsymbol{z}_1\|$:

$$\begin{aligned} \|\boldsymbol{z}_1\| &= \|QQ^\top(A - \theta I)\tilde{\boldsymbol{v}}_s\| = \|Q\left[(\Lambda - \theta I)\tilde{Q}^\top\ \boldsymbol{r}_k^\top\right]\tilde{\boldsymbol{v}}_s\| = \\ &= \|Q(\Lambda - \theta I)\tilde{Q}^\top \tilde{\boldsymbol{v}}_s + Q\boldsymbol{r}_k^\top \tilde{\boldsymbol{v}}_s\| \le \\ &= \|(\Lambda - \theta I)\tilde{Q}^\top \tilde{\boldsymbol{v}}_s\| + \|\boldsymbol{r}_k^\top \tilde{\boldsymbol{v}}_s\| \le \\ &\le \quad \text{by (16) and } (A.3)\ \le \tau\lambda_{j+1}\xi_j + \tau\lambda_j = \tau\lambda_{j+1}(\xi_j + 1). \end{aligned} \tag{22}$$

Then, from (14) and (13) we easily have

$$\|\boldsymbol{z}_2\| = \|(A - \theta I)QQ^\top \tilde{\boldsymbol{v}}_s\| \le 5\tau\xi_j\lambda_{j+1}(1 + \tau). \tag{23}$$

Finally the norm of z_3 is bounded as follows by using (15) and again (13):

$$\|\boldsymbol{z}_3\| = \|QQ^\top(A - \theta I)QQ^\top \tilde{\boldsymbol{v}}_s\| \le 5\tau\lambda_{j+1}\xi_j. \tag{24}$$

Combining (22) – (24) we have:

$$\|\boldsymbol{e}_3\| \le \|\widehat{P}_j\|\|\boldsymbol{z}\| \le \|\widehat{P}_j\|\left(\|\boldsymbol{z}_1\| + \|\boldsymbol{z}_2\| + \|\boldsymbol{z}_3\|\right) \le \tau\lambda_{j+1}\|\widehat{P}_j\|(\xi_j(11 + 5\tau) + 1)\,. \tag{25}$$

We can finally rewrite (21) as

$$P_j J_k^{(j)} \tilde{\boldsymbol{v}}_s = \left(1 - \frac{\theta}{\lambda_s}\right) \tilde{\boldsymbol{v}}_s + \mathbf{err} \tag{26}$$

with

$$\|\mathbf{err}\| \leq \|\boldsymbol{e}_1\| + \|\boldsymbol{e}_2\| + \|\boldsymbol{e}_3\| \leq \tau \left[5\xi_j + \lambda_{j+1}\|\widehat{P}_j\| \left((11 + 5\tau)\xi_j + 3\right)\right],$$

and the thesis holds by setting $C = \left[5\xi_j + \lambda_{j+1}\|\widehat{P}_j\|\big((11 + 5\tau)\xi_j + 3\big)\right]$. □

Remark 3.1
We have just proved that $m - j$ eigenvalues of the preconditioned Newton matrix $P_j J_k^{(j)}$ are close to one if the j-th eigenvalue is well separated from the next higher ones. We also note that $\|\mathbf{err}\|$ can be controlled by suitable choice of τ (tolerance for the initial DACG iterations) as C is usually order of units unless ξ_j is very large.

4. ALGORITHMIC ISSUES

In order to yield an efficient implementation of our spectral preconditioner, the following issues should be taken into account:

1. Limited memory implementation. If the number of eigenpairs being sought is large, it is convenient to limit the number of eigenvectors used for the update. To this end we fix the maximum column dimension of matrix V_j, parameter $l_{\max}$.

2. Conversely, it is also true that in assessing an eigenpair whose index is close to m, the size of matrix V_j is necessarily small and only $m - j$ eigenvalues of the preconditioned matrix will be characterized by Theorem 3.1. In particular when $j = m$, V_j is the empty matrix and a consequent null improvement is provided by the spectral preconditioner, since $\widehat{P}_j \equiv \widehat{P}_0$. To make the proposed approach effective also for such eigenpairs a second variant consists in computing an additional number of approximated eigenpairs by the DACG procedure. We then introduce a further parameter, win, which counts these extra eigenpairs.

Taking into account these variants, in the computation of the j-th eigenpair we will use $V_j = [\tilde{\boldsymbol{v}}_{j+1} \dots \; \tilde{\boldsymbol{v}}_{j_{end}}]$ with $j_{end} = \min\{m + \mathrm{win}, l_{\max} + j\}$ to get the final expression for our spectral preconditioner which from now on we will denote as $P_0^{(j)}$:

$$\begin{aligned} P_{chol} &= (LL^\top)^{-1} \\ \widehat{P}_0^{(j)} &= P_{chol} - W\left(W^\top A V_j\right)^{-1} W^\top, \qquad \text{with} \qquad W = P_{chol} A V_j - V_j \\ P_0^{(j)} &= (I - QQ^\top)\widehat{P}_j^{(0)}(I - QQ^\top) \end{aligned} \tag{27}$$

being $L = IC(\text{LFIL}, \tau_{IC}, A)$ an incomplete triangular Cholesky factor of A, with parameters LFIL, maximum fill-in of a row in L, and τ_{IC} the threshold for dropping small elements in the factorization. The DACG-Newton algorithm with spectral preconditioner is then sketched in Algorithm 1.

4.1. *Repeated application of the spectral preconditioning technique*

In principle every eigenvalue solver may take advantage of the spectral preconditioning technique to update a given preconditioner. In our case the idea is to run twice the DACG method: in the first run a very rough approximation of the leftmost $m + \mathrm{win}$ eigenpairs: $\tilde{\boldsymbol{v}}_1^{(0)}, \tilde{\boldsymbol{v}}_2^{(0)}, \dots, \tilde{\boldsymbol{v}}_{m+\mathrm{win}}^{(0)}$ is provided. To this end we define a tolerance $\mu (> \tau)$ and iterate until the test on the residual $\|A\tilde{\boldsymbol{v}}_j^{(0)} - q(\tilde{\boldsymbol{v}}_j^{(0)})\tilde{\boldsymbol{v}}_j^{(0)}\| \leq \mu q(\tilde{\boldsymbol{v}}_j^{(0)})$ is satisfied. Then a second run of DACG to the final DACG

Algorithm 1 DACG-Newton with spectral preconditioner.

```
1. INPUT:
    1. Matrix $A$;
    2. number of sought eigenpairs $m$;
    3. tolerance and maximum number of its for the outer iteration: $\varepsilon$, ITMAX;
    4. tolerance for the initial eigenvector guess $\tau$;
    5. tolerance and maximum number of its for the inner iteration: $\tau_{PCG}$, ITMAX$_{PCG}$;
    6. parameters for the IC preconditioner:, LFIL and $\tau_{IC}$;
    7. maximum allowed size for matrix $V_j$: $l_{\max}$.
    8. number of extra eigenpairs computed by DACG for matrix $V_j$: $\mathrm{win}$.
2. Compute an incomplete Cholesky factorization of $A$: $P_{chol}$ with parameters LFIL and $\tau_{IC}$.
3. $V := [\,]$.
4. FOR $j := 1$ TO $m + \mathrm{win}$
    (a) Choose $\boldsymbol{x}_0$ such that $V^\top \boldsymbol{x}_0 = 0$.
    (b) Compute $\tilde{\boldsymbol{v}}_j$, an approximation to $\boldsymbol{v}_j$ by the DACG procedure with initial vector $\boldsymbol{x}_0$,
        preconditioner $P_{chol}$ and tolerance $\tau$.
    (c) Set $V := [V\ \tilde{\boldsymbol{v}}_j]$
    END FOR
5. $\tilde{Q} := [\,]$.
6. FOR $j := 1$ TO $m$
    (a) $k := 0$, $\boldsymbol{u}_0 = \tilde{\boldsymbol{v}}_j$, $\theta_0 := \boldsymbol{u}_0^\top A \boldsymbol{u}_0$.
    (b) $Q := [\tilde{Q}\ \boldsymbol{u}_0]$.
    (c) Set $j_{end} = \min\{m + \mathrm{win}, l_{\max} + j\}$, $V_j = [\tilde{\boldsymbol{v}}_{j+1} \dots \tilde{\boldsymbol{v}}_{j_{end}}]$
    (d) Compute $\widehat{P}_0^{(j)}$ using (27) and set $P_0^{(j)} = (I - QQ^\top)\widehat{P}_j^{(0)}(I - QQ^\top)$;
    (e) WHILE $\|A\boldsymbol{u}_k - \theta_k \boldsymbol{u}_k\| > \varepsilon\theta_k$ AND $k <$ IMAX DO
        1. Solve $J_k \boldsymbol{s}_k = -\boldsymbol{r}_k$ for $\boldsymbol{s}_k \perp Q$ by the PCG method with preconditioner $P_j^{(0)}$ and
           tolerance $\varepsilon_{PCG}$.
        2. $\boldsymbol{u}_{k+1} := \dfrac{\boldsymbol{u}_k + \boldsymbol{s}_k}{\|\boldsymbol{u}_k + \boldsymbol{s}_k\|}$, $\theta_{k+1} = \boldsymbol{u}_{k+1}^\top A \boldsymbol{u}_{k+1}$.
        3. $k := k + 1$
        4. $Q := [\tilde{Q}\ \boldsymbol{u}_k]$.
    (f) END WHILE
    (g) Assume $\boldsymbol{v}_j = \boldsymbol{u}_k$ and $\lambda_j = \theta_k$. Set $\tilde{Q} := [\tilde{Q}\ \boldsymbol{v}_j]$
    END FOR
```

tolerance is carried on, using $\tilde{\boldsymbol{v}}_1^{(0)}, \tilde{\boldsymbol{v}}_2^{(0)}, \dots, \tilde{\boldsymbol{v}}_m^{(0)}$ as the starting points and also using them for updating the Cholesky preconditioner. Clearly this DACG step will be accelerated by the non-projected spectral preconditioner $\widehat{P}_j^{(0)}$. The output of this second run will be the sequence of vectors

$\tilde{\boldsymbol{v}}_1, \tilde{\boldsymbol{v}}_2, \ldots, \tilde{\boldsymbol{v}}_m$ which will be in their turn the starting points of the subsequent Newton scheme, while the set $\{\tilde{\boldsymbol{v}}_1, \tilde{\boldsymbol{v}}_2, \ldots, \tilde{\boldsymbol{v}}_m, \tilde{\boldsymbol{v}}_{m+1}^{(0)}, \ldots, \tilde{\boldsymbol{v}}_{m+\text{win}}^{(0)}\}$ will be used for the preconditioner updating.

These steps are summarized in Algorithm 2 where we only underline the variations with respect to Algorithm 1.

Algorithm 2 Two-stage DACG-Newton with spectral preconditioner.

```
1. INPUT: (in addition to that of Algorithm 1)
      9. tolerance for the first DACG stage $\mu (\geq \tau)$;
(STEPS 3. and 4. in Algorithm 1 are substituted with the following ones)
  3.a $V^{(0)} := [\,]$.
  3.b FOR $j := 1$ TO $m + \text{win}$
        (a) Choose $\boldsymbol{x}_0$ such that ${V^{(0)}}^\top \boldsymbol{x}_0 = 0$.
        (b) Compute $\tilde{\boldsymbol{v}}_j^{(0)}$ by the DACG procedure with initial vector $\boldsymbol{x}_0$, preconditioner $P_{chol}$ and
            tolerance $\mu$.
        (c) Set $V^{(0)} := [V^{(0)}\ \tilde{\boldsymbol{v}}_j^{(0)}]$
      END FOR
  4.a $V := [\,]$.
  4.b Define the spectral preconditioner
        $$\widehat{P}_0^{(j)} = P_{chol} - W \left( W^\top A V_j^{(0)} \right)^{-1} W^\top, \; W = P_{chol} A V_j^{(0)} - V_j^{(0)}$$
  4.c FOR $j := 1$ TO $m$
        (a) Compute $\tilde{\boldsymbol{v}}_j$ by the DACG procedure with initial vector $\tilde{\boldsymbol{v}}_j^{(0)}$, preconditioner $P_0^{(j)}$ and
            tolerance $\tau$.
        (b) Set $V := [V\ \tilde{\boldsymbol{v}}_j]$
      END FOR
  4.d $V := [V\ \tilde{\boldsymbol{v}}_{m+1}^{(0)} \ldots \tilde{\boldsymbol{v}}_{m+\text{win}}^{(0)}]$.
```

5. BFGS LOW-RANK UPDATE OF GIVEN PRECONDITIONERS

In [2] the sequence of correction equations $J_k^{(j)} \boldsymbol{s}_k = -\boldsymbol{r}_k$ is preconditioned by means of a sequence of low-rank updates of a given approximate inverse of A which takes the following form, in the computation of the smallest eigenvalue:

$$\begin{aligned}
\widehat{P}_0 &= P_{chol} && (28)\\
P_0 &= (I - \boldsymbol{u}_0 \boldsymbol{u}_0^\top) \widehat{P}_0 (I - \boldsymbol{u}_0 \boldsymbol{u}_0^\top); \\
\widehat{P}_{k+1} &= -\frac{\boldsymbol{s}_k \boldsymbol{s}_k^\top}{\boldsymbol{s}_k^\top \boldsymbol{r}_k} + \left(I - \frac{\boldsymbol{s}_k \boldsymbol{r}_k^\top}{\boldsymbol{s}_k^\top \boldsymbol{r}_k} \right) \widehat{P}_k \left(I - \frac{\boldsymbol{r}_k \boldsymbol{s}_k^\top}{\boldsymbol{s}_k^\top \boldsymbol{r}} \right), \quad k = 0, \ldots, \\
P_{k+1} &= (I - \boldsymbol{u}_{k+1} \boldsymbol{u}_{k+1}^\top) \widehat{P}_{k+1} (I - \boldsymbol{u}_{k+1} \boldsymbol{u}_k^\top). && (29)
\end{aligned}$$

In [2] the following theorem is proved which bounds the norm of the preconditioned matrix in terms of the quality of the initial preconditioner and the closeness of the initial guess to the true eigenvector:

Theorem 5.1
For every constant $K > 0$ there exist $\delta_0, \delta > 0$ such that if $\|E_0\| < \delta_0$, and $\|\boldsymbol{e}_0\| < \delta$ then

$$\|E_{k+1}\| \le \|E_k\| + K\sqrt{\|\boldsymbol{e}_k\|}.$$

where $E_k = I - J_k^{1/2} P_k J_k^{1/2}$ and $\boldsymbol{e}_k = \boldsymbol{u}_k - \boldsymbol{v}_1$.

This BFGS preconditioner suffers from two main drawbacks, namely increasing costs of memory for storing $\boldsymbol{s}$ and $\boldsymbol{r}$, and the increasing cost of preconditioner application with the iteration index k. To overcome these difficulties we define $k_{\max}$ the maximum number of rank two corrections allowed. When the nonlinear iteration counter k is larger than $k_{\max}$, the vectors $\boldsymbol{s}_i, \boldsymbol{r}_i,\ i = k - k_{\max}$ are substituted with the last computed $\boldsymbol{s}_k, \boldsymbol{r}_k$. Vectors $\{\boldsymbol{s}_i, \boldsymbol{r}_i\}$ are stored in an array with n rows and $2 \times k_{\max}$ columns. The implementation of the BFGS update is well suited to parallelization provided that the initial preconditioner is available as an approximate inverse of A and an efficient matrix-vector product routine is employed. The bottleneck is represented by the high number of scalar products which may worsen the parallel efficiency when a very large number of processor is employed. Preliminary numerical results are encouraging as documented in [22].

5.1. *Combination of the BFGS low-rank update and the spectral approaches*

The BFGS approach just recalled and the spectral techniques described in the previous sections can be combined giving raise to a spectral-BFGS preconditioner for the Inexact Newton method, which is defined as follows for a given eigenpair j:

$$\widehat{P}_0^{(j)} = P_{chol} - W\left(W^\top A V_j\right)^{-1} W^\top, \qquad \text{with} \qquad W = P_{chol} A V_j - V_j \tag{30}$$

$$\begin{aligned} \widehat{P}_{k+1}^{(j)} &= -\frac{\boldsymbol{s}_k \boldsymbol{s}_k^\top}{\boldsymbol{s}_k^\top \boldsymbol{r}_k} + \left(I - \frac{\boldsymbol{s}_k \boldsymbol{r}_k^\top}{\boldsymbol{s}_k^\top \boldsymbol{r}_k}\right) \widehat{P}_k^{(j)} \left(I - \frac{\boldsymbol{r}_k \boldsymbol{s}_k^\top}{\boldsymbol{s}_k^\top \boldsymbol{r}_k}\right) \\ & \qquad\qquad\qquad\qquad k = 0, \ldots, \\ P_{k+1}^{(j)} &= (I - Q_{k+1}^{(j)} Q_{k+1}^{(j)\top}) \widehat{P}_{k+1}^{(j)} (I - Q_{k+1}^{(j)} Q_{k+1}^{(j)\top}). \end{aligned} \tag{31}$$

6. NUMERICAL RESULTS

In this Section we provide numerical results where the preconditioned DACG-Newton algorithm is tried for different values of the spectral and BFGS update parameters.

We tested the proposed algorithm in the computation of the 20 smallest eigenpairs of a number of small to large matrices arising from various realistic applications. The CPU times (in seconds) refer to running a Fortran 90 code on a 2 x Intel Xeon CPU E5645 at 2.40GHz (six core) and with 4GB RAM for each core.
In all the test cases, unless differently specified, we computed $m = 20$ eigenpairs with a tolerance for the relative eigenresidual equal to $\varepsilon = 10^{-8}$, namely we stop whenever the following exit test is satisfied:

$$\frac{\|A\boldsymbol{u} - q(\boldsymbol{u})\boldsymbol{u}\|}{q(\boldsymbol{u})} \le \varepsilon.$$

The parameters for the inner PCG solver were set to: $\tau_{PCG} = 10^{-2}$, $\text{ITMAX}_{PCG} = 20$.
The list of the selected problems together with their size n, and nonzero number nz is reported in Table I, where (M)FE stands for (Mixed) Finite Elements.
Some of the matrices are publicly available in the University of Florida (UF) Sparse Matrix Collection at `https://www.cise.ufl.edu/research/sparse/matrices`.

Table I. Main characteristics of the matrices used in the tests.

Matrix	where it comes from	UF collection	n	nz	σ
TRINO	3D-FE discretization of flow in porous media	NO	4560	64030	0.44
MONTE-CARLO	2D-MFE stochastic PDE	NO	77120	384320	2.30
FINAN512	financial problem	YES	74752	596992	3.88
MAT268515	3D-FE discretization of flow in porous media	NO	268515	3 926823	2.67
EMILIA-923	3D-FE elasticity problem	YES	923136	41 005206	1.86
WWW-327	Graph Laplacian of the Web network	NO	325729	2 505945	1.17

We also computed the fill-in σ of the initial preconditioner defined as

$$\sigma = \frac{\text{nonzeros of } L}{\text{nonzeros of lower triangular part of } A}.$$

6.1. *Eigenvalue distribution*

We will first experimentally analyze the eigenvalue distribution of one of the spectral preconditioned matrices in eigensolving test case TRINO. We have considered the evaluation of the sixth ($j = 6$) eigenpair and the third ($k = 3$) Newton iteration. We used the following parameters: $m = 50, l_{\max} = 20, \text{win} = 10$ and for the initial preconditioner $\tau_{IC} = 10^{-1}$, $\text{LFIL} = 10$ giving raise to a fill-in ratio $\sigma = 0.44$. We set $\tau = 10^{-1}$ and $\tau = 10^{-3}$ for the initial DACG iteration.

Table II. Check of the bounds provided by Theorem 3.1 for matrix TRINO. On the left the case with $\tau = 0.1$, on the right $\tau = 10^{-3}$.

s	γ_s	$\|\mathbf{err}\|$	$\|\mathbf{err}\|/\gamma_s$	s	γ_s	$\|\mathbf{err}\|$	$\|\mathbf{err}\|/\gamma_s$
7	0.0330	0.11×10^{-1}	0.35×10^{0}	7	0.0331	0.14×10^{-4}	0.42×10^{-3}
8	0.0542	0.11×10^{-1}	0.20×10^{0}	8	0.0561	0.21×10^{-4}	0.38×10^{-3}
9	0.2434	0.95×10^{-2}	0.39×10^{-1}	9	0.2424	0.30×10^{-4}	0.12×10^{-3}
10	0.2816	0.12×10^{-1}	0.42×10^{-1}	10	0.2808	0.33×10^{-4}	0.12×10^{-3}
11	0.3229	0.12×10^{-1}	0.38×10^{-1}	11	0.3233	0.31×10^{-4}	0.97×10^{-4}
12	0.3517	0.78×10^{-2}	0.22×10^{-1}	12	0.3508	0.34×10^{-4}	0.96×10^{-4}
13	0.3742	0.10×10^{-1}	0.28×10^{-1}	13	0.3743	0.36×10^{-4}	0.96×10^{-4}
14	0.3954	0.11×10^{-1}	0.27×10^{-1}	14	0.3946	0.35×10^{-4}	0.88×10^{-4}
15	0.4122	0.10×10^{-1}	0.25×10^{-1}	15	0.4132	0.29×10^{-4}	0.70×10^{-4}
16	0.4394	0.76×10^{-2}	0.17×10^{-1}	16	0.4396	0.28×10^{-4}	0.63×10^{-4}
17	0.4610	0.60×10^{-2}	0.13×10^{-1}	17	0.4602	0.20×10^{-4}	0.44×10^{-4}
18	0.4680	0.65×10^{-2}	0.14×10^{-1}	18	0.4677	0.25×10^{-4}	0.54×10^{-4}
19	0.4789	0.75×10^{-2}	0.16×10^{-1}	19	0.4793	0.31×10^{-4}	0.65×10^{-4}
20	0.5278	0.56×10^{-2}	0.11×10^{-1}	20	0.5272	0.32×10^{-4}	0.62×10^{-4}
21	0.5494	0.11×10^{-1}	0.19×10^{-1}	21	0.5498	0.28×10^{-4}	0.51×10^{-4}
22	0.5775	0.68×10^{-2}	0.12×10^{-1}	22	0.5774	0.31×10^{-4}	0.54×10^{-4}
23	0.5843	0.49×10^{-2}	0.84×10^{-2}	23	0.5829	0.11×10^{-4}	0.19×10^{-4}
24	0.5887	0.47×10^{-2}	0.81×10^{-2}	24	0.5886	0.28×10^{-4}	0.47×10^{-4}
25	0.6037	0.58×10^{-2}	0.96×10^{-2}	25	0.6035	0.23×10^{-4}	0.38×10^{-4}
26	0.6116	0.63×10^{-2}	0.10×10^{-1}	26	0.6124	0.31×10^{-4}	0.51×10^{-4}

In Table III we check the bound provided by Theorem 3.1 regarding eigenvalues of the preconditioned matrix. We report the predicted eigenvalues $\gamma_s = \left(1 - \frac{\theta_k^{(j)}}{\lambda_s}\right)$, together with the norm of the computed residual $\|\mathbf{err}\| = \|P_j J_k^{(j)}\tilde{\boldsymbol{v}}_s - \gamma_s \tilde{\boldsymbol{v}}_s\|$ and also the relative residual $\frac{\|\mathbf{err}\|}{\gamma_s}$. On the left we report the results with DACG tolerance $\tau = 0.1$ on the right with $\tau = 10^{-3}$. From the table we see that the relative error, even with $\tau = 0.1$, is always smaller than 0.1, except for γ_7, γ_8, thus confirming the findings of Theorem 3.1. Moreover the residuals follow the dependence on τ as those in the table on the right are roughly two orders of magnitude smaller than the corresponding residuals on the left.

In view of the small size of this matrix, we also computed all the "exact" eigenvalues of both $P_{chol}J_3^{(6)}$ and $P_0^{(6)}J_3^{(6)}$. In Figure 1 we report the distribution of the computed smallest eigenvalues of these preconditioned matrices. We explicitly computed the condition number of both preconditioned matrices finding $\kappa(P_{chol}J_3^{(6)}) = 2.6 \times 10^3$ and $\kappa(P_0^{(6)}J_3^{(6)}) = 3.6 \times 10^2$ with an important reduction of the condition number provided by the spectral preconditioner.

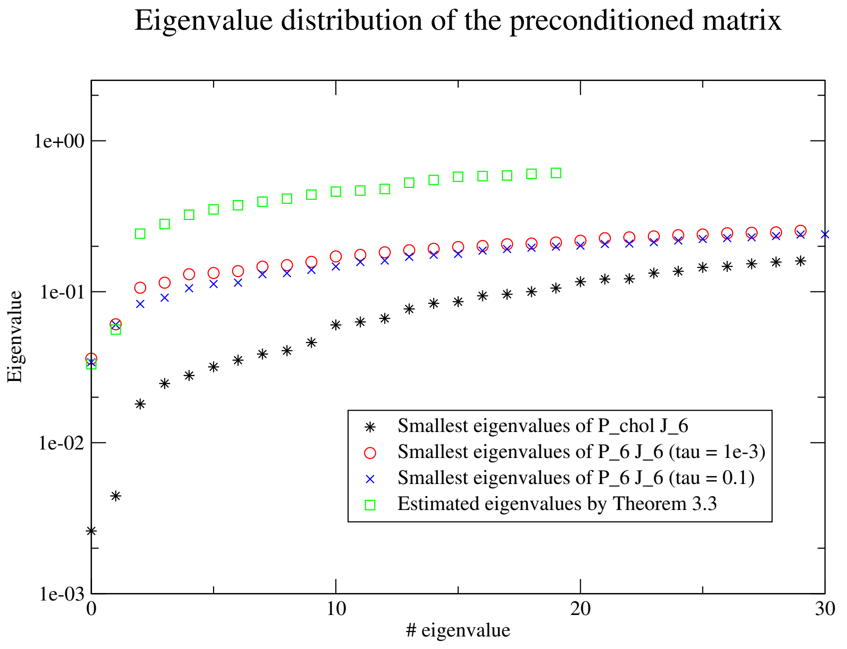


Figure 1. "Exact" smallest eigenvalues of the preconditioned Jacobian at step 3 of Newton iteration to compute eigenpair #6 of matrix TRINO. The smallest eigenvalues of $P_{chol}J_3^{(6)}$ and $P_0^{(6)}J_3^{(6)}$ with two different values of τ are displayed together with the estimates provided by Theorem 3.1.

Careful interpretation of the Figure reveals that:

1. The smallest eigenvalues of $P_0^{(6)}J_3^{(6)}$ (circles) are much larger than those of $P_{chol}J_3^{(6)}$ (stars).
2. The two smallest eigenpairs of the preconditioned $P_0^{(6)}J_3^{(6)}$ are consistent with Theorem 3.1, being close to γ_7 and γ_8, respectively, as shown by the coincidence of squares and circles. The third smallest eigenvalue of $P_0^{(6)}J_3^{(6)}$ is equal to eigenvalue #20 of $P_{chol}J_3^{(6)}$, which accounts for a *bottom-up shift* of the eigenvalues of the preconditioned matrices.
3. The eigenvalue distribution of the spectral-preconditioned matrix $P_0^{(6)}J_3^{(6)}$ is not significantly affected by the value of τ (compare symbols: $\times$ and $\circ$ in Figure 1).

We conclude this analysis reporting in Table III the absolute and relative residual norm for two more test cases: MAT268515 and EMILIA-923 again in preconditioning $J_3^{(6)}$. In this case we used

Table III. Check of the bounds provided by Theorem 3.1 for matrices MAT268515 (left) and EMILIA-923 (right).

s	γ_s	$\|\mathbf{err}\|$	$\|\mathbf{err}\|/\gamma_s$
7	0.1681	6.20×10^{-4}	3.71×10^{-3}
8	0.2361	1.47×10^{-3}	6.32×10^{-3}
9	0.4769	7.80×10^{-4}	1.61×10^{-3}
10	0.4805	1.63×10^{-2}	3.41×10^{-2}
11	0.5292	3.18×10^{-2}	6.01×10^{-2}
12	0.6492	4.39×10^{-2}	6.76×10^{-2}
13	0.5448	1.83×10^{-2}	3.36×10^{-2}
14	0.6020	5.98×10^{-3}	9.89×10^{-3}
15	0.6020	4.92×10^{-3}	7.23×10^{-3}
16	0.6824	1.22×10^{-2}	1.78×10^{-2}

s	γ_s	$\|\mathbf{err}\|$	$\|\mathbf{err}\|/\gamma_s$
7	0.0681	6.88×10^{-4}	1.01×10^{-2}
8	0.0839	4.42×10^{-4}	5.22×10^{-3}
9	0.1003	7.60×10^{-4}	7.60×10^{-3}
10	0.3322	4.77×10^{-3}	1.44×10^{-2}
11	0.3701	1.86×10^{-3}	5.02×10^{-2}
12	0.3750	2.76×10^{-3}	7.39×10^{-3}
13	0.3913	1.87×10^{-2}	4.79×10^{-2}
14	0.4281	6.09×10^{-3}	1.42×10^{-2}
15	0.4851	5.69×10^{-3}	1.12×10^{-2}
16	0.4880	2.11×10^{-3}	4.33×10^{-3}

$l_{\max} = 10$, $\text{win} = 5$ and $\tau = 0.02$ for both tests. Again we found that almost all eigenvalues are clustered around one, and in any case $\|\mathbf{err}\|$ is smaller than, or very close to, the tolerance τ.

6.2. *Results with matrices arising from discretization of PDEs*

In Tables from IV to VI we report the results in terms of outer iterations (OUT) matrix-vector products (MVP) and total CPU time of the DACG-Newton method with the proposed spectral preconditioner. These three test cases rely on matrices all arising from various discretization of PDEs modelling fluid flow or geomechanical processes in porous media. For these matrices we carried out an extensive number of runs for a wide range of the parameters $l_{\max}$, the maximum column number of V_j, win, the additional number of eigenvectors computed by the first DACG stage and $k_{\max}$ the maximum number of BFGS corrections. Whenever $k_{\max}$ is set equal to zero only the spectral preconditioner is used, as described in Algorithms 1 and 2; if the second DACG tolerance is not specified it means that DACG is run only once as in Algorithm 1.

Considering the results of Tables IV and V, we notice that for a fixed $k_{\max}$ value the spectral preconditioner proposed in this paper is very efficient to reduce both the outer-inner iterations in the Newton phase and the overall matrix-vector products and CPU time. Moreover, the two-stage DACG run (see Algorithm 2) provides a further improvement by reducing the cost of the initial assessment of the Newton starting point. In computing the 20 leftmost eigenpairs $\text{win} = 5$ seems the best value, which results in a trade-off between the extra DACG cost and the improved acceleration of the spectral preconditioner. The BFGS acceleration is more effective when not combined with the spectral technique while, for these test cases, it does not provide significant improvement when it is used together with the spectral update.

The results reported in Table VI enhance even more the effects of the spectral preconditioner which produces an improvement in terms of total MVP from 25% (using $k_{\max} = 5$) to more than 50% (if $k_{\max} = 0$ is selected). For this test case the combined BFGS-spectral preconditioners produces the best result in terms of CPU time and number of MVP.

The last two rows of Table VI give experimental evidence that the spectral preconditioner can be efficiently applied to a selected eigenvalue solver (in our case DACG) which is run twice: the first stage is aimed at constructing matrix V_j containing the approximate eigenvectors; the second stage, accelerated by the spectral preconditioner, is run up to the final tolerance.

In order to have a graphical evidence of the improvement provided by the spectral preconditioner, for matrix EMILIA-923, in Figure 2 we plot the convergence profiles of the Newton phase in computing eigenpairs $j = 5$ (circles), $j = 10$ (squares) and $j = 15$ (stars) and four combination of the parameters (red symbols = spectral + BFGS, blue symbols = spectral only, magenta symbols = BFGS only, and black symbols = fixed preconditioner); in Figure 3 for the same eigenpair levels

Table IV. Timings and iterations with the preconditioned DACG-Newton method for the computation of $m = 20$ eigenpairs of matrix MONTE-CARLO.

			DACG				Newton			Total	
							Iterations		CPU		
win	$l_{\max}$	$k_{\max}$	μ	τ	Its.	CPU	OUT	Inner		MVP	CPU
0	0	0		0.02	1104	9.89	159	2837	24.41	4100	36.38
5	10	0		0.02	1326	12.23	68	698	8.79	2092	21.10
5	20	0		0.02	1326	12.47	58	620	7.85	2004	20.41
5	10	0	0.2	0.02	992	9.55	67	719	8.26	1778	17.90
5	20	0	0.2	0.02	991	9.53	62	642	7.88	1695	17.49
0	0	1		0.02	1104	9.87	124	2043	20.12	3271	30.08
5	10	1		0.02	1326	12.19	68	670	8.08	2064	20.35
5	20	1		0.02	1326	12.58	62	600	8.21	1988	20.87
5	10	1	0.2	0.02	992	9.56	65	689	8.32	1746	17.97
5	20	1	0.2	0.02	991	9.60	52	618	7.69	1661	17.34
0	0	5		0.02	1104	9.86	99	1600	16.33	2803	26.27
5	10	5		0.02	1326	12.20	67	670	8.16	2063	20.44
5	20	5		0.02	1326	12.48	61	598	8.15	1985	20.70
5	10	5	0.2	0.02	992	9.52	68	692	8.38	1752	17.98
5	20	5	0.2	0.02	991	9.52	55	615	7.78	1661	17.38

Table V. Timings and iterations with the DACG-Newton method for the computation of $m = 20$ eigenpairs of matrix MAT268515.

			DACG				Newton			Total	
							Iterations		CPU		
win	$l_{\max}$	$k_{\max}$	μ	τ	Its.		OUT	Inner		MVP	CPU
0	0	0		0.02	782	45.57	105	1349	86.89	2236	133.87
5	10	0		0.02	962	57.94	61	587	45.14	1610	104.49
5	20	0		0.02	962	57.21	60	542	43.80	1564	102.45
5	10	0	0.2	0.02	776	50.03	63	652	49.37	1491	100.80
0	0	5		0.02	782	45.36	86	1022	71.08	1890	117.85
5	10	5		0.02	962	57.20	55	557	44.35	1574	102.96
5	20	5		0.02	962	57.54	55	516	42.76	1533	101.73
5	10	5	0.2	0.02	776	51.62	60	604	42.01	1440	95.07

the convergence profiles of the DACG solver with (red symbols) or without (blue symbols) spectral acceleration.

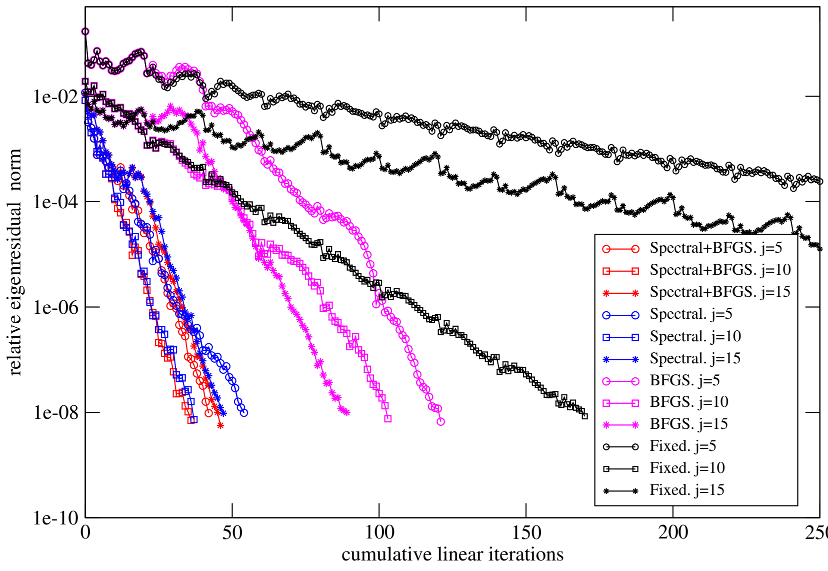
6.3. *A matrix with clustered small eigenvalues*

We now report the results in eigensolving matrix FINAN512. For this test case the 20 smallest eigenvalues are much clustered, thus suggesting that the spectral preconditioner could not accelerate the iterative eigensolvers. We show in Table VII the 20 smallest eigenvalue together with the reciprocal of the indicator ξ_j. It is worth observing that $\xi_j^{-1} = 1 - \dfrac{\lambda_j}{\lambda_{j+1}}$ coincides with the smallest eigenvalue of the preconditioned matrix as predicted by Theorem 3.1. For every index j these values are closer to zero than to one, thus suggesting a high condition number of the preconditioned matrix. The results reported in Table VIII confirms this observation. The spectral preconditioning technique does not provide a significant improvement of the iterative solver performance, irrespective on

Table VI. Timings and iterations with the DACG-Newton method for the computation of $m = 20$ eigenpairs of matrix EMILIA-923.

			DACG				Newton			Total	
							Iterations		CPU		
win	$l_{\max}$	$k_{\max}$	μ	τ	Its.	CPU	OUT	Inner		MVP	CPU
0	0	0		0.02	1404	413.85	226	4011	1306.22	5641	1738.43
5	10	0		0.02	1716	507.94	86	876	319.25	2678	845.65
5	20	0		0.02	1716	510.43	77	817	309.26	2610	838.24
0	0	1		0.02	1404	409.50	140	2341	764.60	3885	1192.38
5	10	1		0.02	1716	508.20	79	819	306.69	2614	833.33
5	20	1		0.02	1716	510.68	70	765	295.03	2551	823.98
0	0	5		0.02	1404	410.16	104	1705	568.14	3213	996.60
5	10	5		0.02	1716	510.20	76	800	306.15	2592	834.67
5	20	5		0.02	1716	508.23	66	745	283.51	2527	809.97
5	10	5	0.1	0.02	1381	420.23	71	838	309.50	2290	747.93
5	20	5	0.1	0.02	1379	422.51	64	750	288.20	2193	729.05
0	0	0		10^{-8}	3990	1256.63	–	–	–	3990	1274.75
5	20	0	0.1	10^{-8}	2250	764.22	–	–	–	2250	782.80

Figure 2. Convergence profile of the Newton phase for problem EMILIA-923.



the $l_{\max}$ and win values. Here the BFGS update strategy is the winner one since it produces an improvement from 160.8 CPU seconds ($k_{\max} = 0$) to 27.07 seconds ($k_{\max} = 5$) with no spectral update.

6.4. *A graph Laplacian matrix with zero smallest eigenvalue*

Matrix WWW-327 is the Laplacian of the Web network within *nd.edu* domain [23]. This network is directed but arc direction has been ignored in order to obtain a symmetric Laplacian. The corresponding Laplacian matrix is characterized by a zero smallest eigenvalue corresponding the eigenvector with all components equal to one. We notice that, due to the erratic sparsity of the graph Laplacian matrix, the computation of an efficient IC decomposition was rather costly as it took

Figure 3. Convergence profile of the DACG method for problem EMILIA-923.

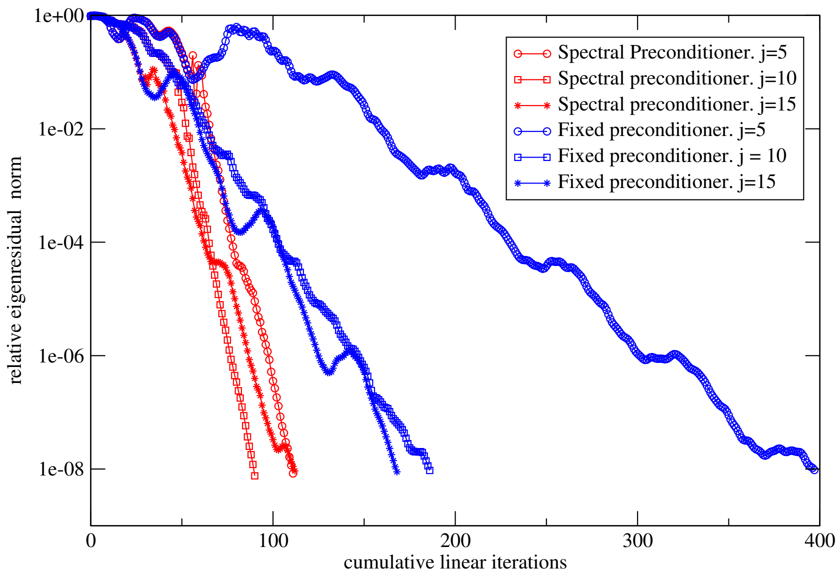


Table VII. Eigenvalues λ_j and relative separation ξ_j^{-1}, for matrix FINAN512.

j	λ_j	ξ_j^{-1}	j	λ_j	ξ_j^{-1}
1	0.94746	2.9×10^{-3}	11	1.05180	5.7×10^{-3}
2	0.95024	6.2×10^{-2}	12	1.05779	1.2×10^{-2}
3	1.01279	6.0×10^{-3}	13	1.07086	3.5×10^{-3}
4	1.01895	9.8×10^{-3}	14	1.07460	1.8×10^{-3}
5	1.02902	2.7×10^{-3}	15	1.07652	1.6×10^{-3}
6	1.03176	1.1×10^{-3}	16	1.07829	1.5×10^{-5}
7	1.03288	6.3×10^{-3}	17	1.07831	6.9×10^{-4}
8	1.03943	3.3×10^{-3}	18	1.07905	1.9×10^{-3}
9	1.04282	2.0×10^{-4}	19	1.08108	1.2×10^{-3}
10	1.04303	8.3×10^{-3}	20	1.08235	5.7×10^{-3}

Table VIII. Timings and iterations with the DACG-Newton method for the computation of $m = 20$ eigenpairs of matrix FINAN512.

			DACG				Newton			Total	
							Iterations		CPU		
win	$l_{\max}$	$k_{\max}$	μ	τ	Its.	CPU	OUT	Inner		MVP	CPU
0	0	0		10^{-3}	1301	12.64	704	13780	147.99	15785	160.80
5	10	0		10^{-3}	1629	16.07	467	9075	98.72	11171	114.97
0	0	1		10^{-3}	1301	12.53	122	2127	22.99	3550	35.70
5	10	1		10^{-3}	1629	16.21	81	1291	15.66	3001	32.05
0	0	5		10^{-3}	1301	12.51	80	1300	14.39	2681	27.07
5	10	5		10^{-3}	1629	16.21	60	971	11.97	2660	28.36
5	10	5	0.2	10^{-3}	1196	12.84	65	1011	12.49	2272	25.51
10	15	5	0.2	10^{-3}	1220	13.77	64	1002	13.02	2286	26.97

a large percentage of the total CPU time (about 205 seconds). Only for this test case we set the maximum number of PCG inner iterations to 30.

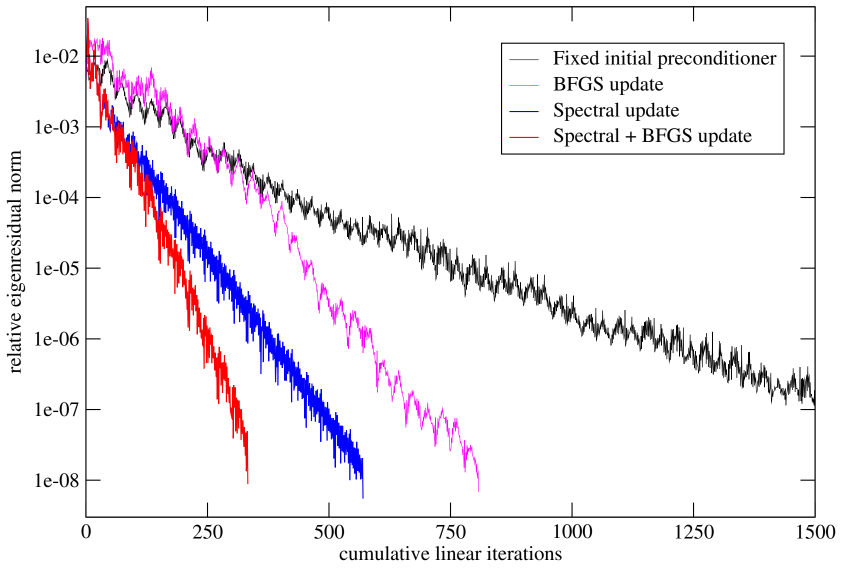
Table IX. Timings and iterations with the DACG-Newton method for the computation of $m = 20$ eigenpairs of matrix WWW-327. Symbol ‡ stands for: no convergence of the Newton step within 200 outer iterations.

			DACG				Newton			Total	
							Iterations		CPU		
win	$l_{\max}$	$k_{\max}$	μ	τ	Its.	CPU	OUT	Inner		MVP	CPU
0	0	0		0.01	20593	749.37	‡	‡	‡	‡	‡
5	15	0	0.1	0.01	13347	536.51	1180	34838	1937.66	49365	2692.25
0	0	5		0.01	20593	749.37	1392	41436	1987.58	63421	2949.82
5	15	5	0.1	0.01	13347	543.57	352	10290	568.02	23989	1318.65
0	0	15		0.01	20593	749.37	812	24111	1431.54	45516	2388.39
3	10	15	0.1	0.01	13306	535.68	353	10349	625.74	24008	1368.16
5	15	15	0.1	0.01	13347	543.57	305	8904	542.96	22556	1295.08
8	20	20	0.1	0.01	14298	584.27	280	8200	494.11	22278	1283.70

With a fixed initial Cholesky preconditioner we report no convergence of the Newton phase within 200 outer iterations. If either the BFGS update alone is used (see the third and the fifth row in the Table) or the spectral preconditioner without BFGS correction (2nd row), convergence of the Newton step reveals very slow. The other runs reported in Table IX show the great improvement in the Newton phase, which is speeded-up by a factor three in terms of CPU time and MVP, when the spectral preconditioner is used in combination with the BFGS low-rank update.

This behavior is clearly evident in Figure 4 where the convergence profile is provided of the DACG-Newton method with four preconditioner strategies to assess eigenpair $j = 7$.

Figure 4. Convergence profile of the DACG-Newton method for eigenpair # 7 and problem WWW-327.



6.5. *Suggestions on the choice of the parameters*

When employing both the two-stage spectral and the BFGS acceleration there are a number of parameters to be assessed. However, the two parameters of the spectral preconditioner are relatively easy to determine: our experiments suggest that, denoting as before with m the number of sought eigenpairs, choosing $l_{\max} \in \left[\frac{m}{2}, m\right]$ and $\text{win} \in [5, 10]$ will lead to optimal performances. Also $k_{\max} = 5$ seems to be an effective choice in most cases as the Newton process is expected to converge within a small number of iterations (and this is also dependent on the initial DACG approximation). Finally the choice of the two DACG tolerances, though clearly problem dependent, can be successfully set as $\mu \approx 10^{-1}$ and $\tau \approx 10^{-2}$.

6.6. *Comparison with Jacobi-Davidson*

We include in this section a comparison of the algorithm presented and analyzed in the previous sections with the well-known Jacobi-Davidson (JD) method. For the details of this method we refer to the original paper [7], as well as to successive works [24, 25, 19] which analyze both theoretically and experimentally a number of variants of this method. In this paper, we followed the implementation suggested in the previously cited work [19], i.e. we made use of the PCG method as the inner solver, with the same initial preconditioner as that used in the DACG-Newton method. Also the exit tests used in the two methods are identical for both the outer iteration and the inner PCG solver.

Table X. Comparison between the spectral preconditioned DACG-Newton and Jacobi-Davidson. †: In solving problem WWW-327 JD was not able to compute the eigenpairs with the requested tolerance (10^{-8}).

problem	DACG-Newton								Jacobi-Davidson		
	win	$l_{\max}$	$k_{\max}$	μ	τ	MVP	OUT	CPU	MVP	OUT	CPU
MONTE-CARLO	5	20	5	0.2	0.02	1561	55	17.24	2156	130	23.79
MAT268515	5	10	5	0.2	0.02	1440	60	95.07	1624	114	99.86
EMILIA-923	5	20	5	0.1	0.02	2193	64	729.05	2346	140	885.08
FINAN512	5	10	5	0.1	10^{-3}	2272	65	25.51	1948	131	31.41
WWW-327 ($\varepsilon = 10^{-8}$)	8	20	20	0.1	0.01	22278	280	1283.70	†	†	†
WWW-327 ($\varepsilon = 10^{-4}$)	8	20	20	0.2	0.05	15068	179	902.82	10593	424	622.55

In the JD implementation two parameters are crucial for its efficiency namely $m_{\min}$ and $m_{\max}$, the smallest and the largest dimension of the subspace where the Rayleigh-Ritz projection takes place. After some attempts, we found that $m_{\min} = 15$ and $m_{\max} = 25$ were on the average the optimal values of such parameters. In all the examples and both solvers we set the maximum number of inner PCG iterations to 20 with the only exception of problem WWW-327 where we set $\text{ITMAX}_{PCG} = 30$.

The results of the comparison are summarized in Table X where we also specify the parameters of the spectral (two-stage) DACG-Newton algorithm. Regarding the first four problems, it is found that the proposed method is superior to the JD method in terms of CPU time and also (on three problems out of four) in terms of number of MVP. For the problem WWW-327 JD was not able to compute the eigenpairs up to the prescribed accuracy. To perform a proper comparison, we then run both the DACG-Newton and the JD solvers with a lower accuracy ($\varepsilon = 10^{-4}$). In this case, JD performs better than the proposed DACG-Newton method.

We conclude this section by observing that also the Jacobi-Davidson method itself could in principle take advantage of the two-stage spectral preconditioner. The JD solver could be used either as the second stage in combination with the DACG method, or on its own. In the latter case, after a first JD stage providing a set of eigenvectors to a low accuracy, the second JD run will use information from the subspace generated by the first-run Rayleigh-Ritz procedure, to improve the initial guess, and the approximate eigenvectors just obtained to form the spectral preconditioner.

7. CONCLUSIONS

We have proposed a class of spectral preconditioners for the two-stage iterative eigensolver DACG-Newton which is shown to cluster the eigenvalues of the preconditioned matrix in the Newton phase with a consequent acceleration of the iterative procedure. We have theoretically proved that a number of the eigenvalues of the preconditioned matrix are related to the relative separation between the eigenvalue being sought and the next higher ones. Numerical experiments onto large size problems arising from various applications show that in any case the spectral preconditioner is effective in accelerating the Newton phase and particularly so when it is combined with a low-rank update of BFGS type. This spectral updating technique is also employed in accelerating the DACG method itself by running it twice: in the first run the eigenpairs are approximated with very low accuracy, the second run is accelerated by the spectral preconditioner. We stress that this technique can be applied to the eigensolver of choice which can be employed in its first run to assess both a good starting point and a rough approximation of the leftmost eigenvectors which constitute the basis for the spectral preconditioner to accelerate the second run of the same eigensolver. In particular, as a future work, we plan to apply the spectral preconditioner for the acceleration of the Jacobi-Davidson method.

Acknowledgments. The work of the first author has been partially supported by the Spanish grant MTM2010-18674. Both authors have been partially supported by the Italian project CPDA155834/15: "Stable and efficient discretizations of the mechanics of faults"

REFERENCES

1. Martínez A. Tuned preconditioners for iterative SPD eigensolvers. *Numer. Lin. Alg. Appl.* 2016; **23**(3):427–443.
2. Bergamaschi L, Martínez A. Efficiently preconditioned inexact Newton methods for large symmetric eigenvalue problems. *Optimization Methods & Software* 2015; **30**:301–322.
3. Bathe KJ. *Finite Element Procedures in Engineering Analysis*. Prentice-Hall: Englewood Cliffs, 1982.
4. Saad Y, Stathopoulos A, Chelikowsky J, Wu K, Öğüt S. Solution of large eigenvalue problems in electronic structure calculations. *BIT* 1996; **36**(3):563–578. International Linear Algebra Year (Toulouse, 1995).
5. Bozzo E, Franceschet M. Approximations of the generalized inverse of the graph laplacian matrix. *Internet Mathematics* 2012; **8**:1–26.
6. Bergamaschi L, Bozzo E. Computing the smallest eigenpairs of the graph Laplacian. *S$\vec{e}$MA Journal* 2016; To appear.
7. Sleijpen GLG, van der Vorst HA. A Jacobi-Davidson method for linear eigenvalue problems. *SIAM J. Matrix Anal.* 1996; **17**(2):401–425.
8. Carpentieri B, Duff IS, Giraud L. A class of spectral two-level preconditioners. *SIAM J. Sci. Comput.* 2003; **25**(2):749–765 (electronic).
9. Duff IS, Giraud L, Langou J, Martin E. Using spectral low rank preconditioners for large electromagnetic calculations. *Int. J. Numer. Methods Engrg.* 2005; **62**:416–434.
10. Giraud L, Gratton S, Martin E. Incremental spectral preconditioners for sequences of linear systems. *Applied Numerical Mathematics* 2007; **57**(11-12):1164 – 1180. Numerical Algorithms, Parallelism and Applications (2).
11. Mas J, Cerdán J, Malla N, Marín J. Application of the Jacobi-Davidson method for spectral low-rank preconditioning in computational electromagnetics problems. *S$\vec{e}$MA Journal* 2015; **67**:39–50.
12. Simoncini V, Eldén L. Inexact Rayleigh quotient-type methods for eigenvalue computations. *BIT* 2002; **42**(1):159–182.
13. Bergamaschi L, Bru R, Martínez A. Low-rank update of preconditioners for the inexact Newton method with SPD jacobian. *Mathematical and Computer Modelling* 2011; **54**(7–8):1863–1873.
14. Bergamaschi L, Gambolati G, Pini G. Asymptotic convergence of conjugate gradient methods for the partial symmetric eigenproblem. *Numer. Lin. Alg. Appl.* 1997; **4**(2):69–84.
15. Bergamaschi L, Putti M. Numerical comparison of iterative eigensolvers for large sparse symmetric matrices. *Comp. Methods App. Mech. Engrg.* 2002; **191**(45):5233–5247.
16. Knyazev AV. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM J. Sci. Comput.* 2001; **23**(2):517–541.
17. Knyazev AV, Argentati ME, Lashuk I, Ovtchinnikov EE. Block locally optimal preconditioned eigenvalue xolvers (BLOPEX) in Hypre and PETSc. *SIAM Journal on Scientific Computing* 2007; **29**(5):2224–2239.
18. Bergamaschi L, Martínez A, Pini G. Parallel Rayleigh Quotient optimization with FSAI-based preconditioning. *J. Applied Mathematics* 2012; **Article ID 872901**:1–14.
19. Notay Y. Combination of Jacobi-Davidson and conjugate gradients for the partial symmetric eigenproblem. *Numer. Linear Algebra Appl.* 2002; **9**(1):21–44.
20. Freitag MA, Spence A. A tuned preconditioner for inexact inverse iteration applied to Hermitian eigenvalue problems. *IMA J. Numer. Anal.* 2008; **28**(3):522–551.
21. Freitag MA, Spence A. Shift-invert Arnoldi's method with preconditioned iterative solves. *SIAM J. Matrix Anal. Appl.* 2009; **31**(3):942–969.
22. Bergamaschi L, Martínez A. Parallel RFSAI-BFGS preconditioners for large symmetric eigenproblems. *J. Applied Mathematics* 2013; **Article ID 767042**:1–10.
23. Albert R, Jeong H, Barabási AL. Diameter of the world-wide web. *Nature* 1999; **401**(6749):130–131.
24. Stathopoulos A. A case for a biorthogonal Jacobi-Davidson method: restarting and correction equation. *SIAM J. Matrix Anal. Appl.* 2002; **24**(1):238–259 (electronic).

25. Fokkema DR, Sleijpen GLG, van der Vorst HA. Jacobi-Davidson style QR and QZ algorithms for the reduction of matrix pencils. *SIAM J. Sci. Comput.* 1998; **20**(1):94–125 (electronic).