Tuned preconditioners for the eigensolution of large SPD matrices arising in engineering problems

Ángeles Martínez*,†

Department of Mathematics, University of Padua, via Trieste 63, 35100 Padova, Italy

SUMMARY

In this paper, we study a class of tuned preconditioners that will be designed to accelerate both the DACG–Newton method and the implicitly restarted Lanczos method for the computation of the leftmost eigenpairs of large and sparse symmetric positive definite matrices arising in large-scale scientific computations. These tuning strategies are based on low-rank modifications of a given initial preconditioner. We present some theoretical properties of the preconditioned matrix. We experimentally show how the aforementioned methods benefit from the acceleration provided by these tuned/deflated preconditioners. Comparisons are carried out with the Jacobi–Davidson method onto matrices arising from various large realistic problems arising from finite element discretization of PDEs modeling either groundwater flow in porous media or geomechanical processes in reservoirs. The numerical results show that the Newton-based methods (which includes also the Jacobi–Davidson method) are to be preferred to the – yet efficiently implemented – implicitly restarted Lanczos method whenever a small to moderate number of eigenpairs is required.

Accepted 28 December 2015

KEY WORDS: eigenvalues; Lanczos' method; SPD matrix; Newton's method; BFGS update; incomplete Cholesky preconditioner

1. INTRODUCTION

Consider a symmetric positive definite (SPD) matrix A, which is also assumed to be large and sparse. We will denote as

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

the eigenvalues of A and

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

the corresponding (normalized) eigenvectors.

The computation of the $p \ll n$ leftmost eigenpairs of A is a common task in many scientific applications. Typical examples are offered by the vibrational analysis of mechanical structures [1], and the electronic structure calculations [2]. Computation of a few eigenpairs is also crucial in the approximation of the generalized inverse of the graph Laplacian [3, 4].

Recently in [5], an efficiently preconditioned Newton method (deflation-accelerated conjugate gradient [DACG]–Newton, DN in short) has been developed, which has proven to display comparable performances against the well-known Jacobi–Davidson (JD) method [6]. However, one can wonder if the JD method can be considered as the optimal benchmark to test an iterative eigensolver. Actually, the preconditioned Arnoldi method is commonly considered the most efficient tool especially for SPD eigenproblems.

*Correspondence to: Ángeles Martínez, Department of Mathematics, University of Padua, via Trieste 63, 35100 Padova, Italy.

†E-mail: angeles.martinez@unipd.it

The advantage offered by the accelerated Newton and JD solvers versus Krylov-like methods is that the former require a number of linear system solutions to a (possibly) very low accuracy while for the latter methods, 'exact' (i.e., very accurate) solution to the inner linear systems is mandatory. In recent papers such as [7, 8], however, inexact variants of the implicitly restarted Arnoldi (IRA) (Lanczos for symmetric matrices, IRL in the sequel) method have been studied, which improve its efficiency by properly relaxing the tolerance of the inner linear systems. Also, a family of ad hoc tuned preconditioners have been developed with the purpose of accelerating the solution of the same linear systems.

The main purpose of this paper is to provide a class of tuned preconditioners that will be designed to accelerate both the DN method and the inexact Lanczos' method. We develop some theoretical properties and perform a set of experimental tests, which account for the effectiveness of such acceleration. Moreover, these two eigensolvers are compared with the JD method on a set of large SPD matrices arising from finite element discretization of PDE modeling either groundwater flow in porous media or geomechanical processes in reservoirs.

The numerical results show the significant improvement provided by the low-rank modification of a given initial preconditioner for both DN and IRL methods on all test problems. Comparisons with the well-established JD method reveal that all in all, JD still proves the most performing algorithm, yet displaying similar efficiency as the DN method.

The outline of the paper is as follows: In Section 2, we recall the BFGS preconditioner in the framework of the DN method. We summarize its theoretical properties and sketch its implementation details. Section 3 is devoted to the definition of a number of tuned preconditioners to accelerate the Lanczos method. Section 4 reports the numerical results and comparisons against the JD method, while Section 5 draws the conclusions.

2. THE DACG–NEWTON METHOD WITH BFGS PRECONDITIONER

The computation of the leftmost eigenpair of an SPD matrix A can be recast as the following nonlinear problem

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

being $q(A, \boldsymbol{x}) = \boldsymbol{x}^\top A\boldsymbol{x}$ the Rayleigh quotient. The corresponding Jacobian matrix can be written as

$$J_C(\boldsymbol{x}) \equiv A - \boldsymbol{x}^\top A\boldsymbol{x}\, I - 2\boldsymbol{x}\boldsymbol{x}^\top A. \tag{2}$$

The $(k+1)$-th iteration of Newton's method applied to Equation (1) can therefore be written as

$$\begin{aligned} J_C(\boldsymbol{u}_k)\boldsymbol{s}_k &= -\left(A\boldsymbol{u}_k - \left(\boldsymbol{u}_k^\top A\boldsymbol{u}_k\right)\boldsymbol{u}_k\right) \\ \boldsymbol{u}_{k+1} &= \frac{\boldsymbol{u}_k + \boldsymbol{s}_k}{\|\boldsymbol{u}_k + \boldsymbol{s}_k\|} \end{aligned}$$

However, it has been observed by various authors (e.g., [9, 10]) that Newton's method applied to Equation (1) suffers stagnation. The most popular and efficient variant consists in projecting the Jacobian J_C in the space orthogonal to the current iterate $\boldsymbol{u}_k$ giving rise to the following Newton–Grassmann method, where $J_k = \left(I - \boldsymbol{u}_k\boldsymbol{u}_k^\top\right)\left(A - \boldsymbol{u}_k^\top A\boldsymbol{u}_k I\right)\left(I - \boldsymbol{u}_k\boldsymbol{u}_k^\top\right)$ and $\boldsymbol{r}_k = \left(A\boldsymbol{u}_k - \left(\boldsymbol{u}_k^\top A\boldsymbol{u}_k\right)\boldsymbol{u}_k\right)$:

Repeat until convergence:

Set $\boldsymbol{u}_0$ as an initial approximation of $\boldsymbol{v}_1$, $\qquad k = 0$

$$\text{solve approximately} \quad J_k\boldsymbol{s}_k = -\boldsymbol{r}_k \quad \text{for } \boldsymbol{s}_k \perp \boldsymbol{u}_k \tag{3}$$

$$\text{set} \quad \boldsymbol{u}_{k+1} = \frac{\boldsymbol{u}_k + \boldsymbol{s}_k}{\|\boldsymbol{u}_k + \boldsymbol{s}_k\|},\ k = k+1 \tag{4}$$

Note that the previous iteration is the basis of the well-known JD method, which combines the projected Newton's iteration (also called the correction equation) with a Rayleigh–Ritz step.
Following the idea described in [5, 11–13], we propose a sequence of preconditioners for the correction Equation (3) using the BFGS rank-two update of a given initial approximation of A^{-1}. To precondition the initial Newton system $J_0 \boldsymbol{s}_0 = -\boldsymbol{r}_0$, we chose to use a projected incomplete Cholesky preconditioner with partial fill-in [14]: $P_0 = \left(I - \boldsymbol{u}_0 \boldsymbol{u}_0^\top\right) \widehat{P}_0 \left(I - \boldsymbol{u}_0 \boldsymbol{u}_0^\top\right)$ with $\widehat{P}_0 = \left(LL^\top\right)^{-1}$ 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. Then a sequence of projected preconditioners for the subsequent linear systems $J_{k+1}\boldsymbol{s}_{k+1} = -\boldsymbol{r}_{k+1}$ may be defined by using an approximated BFGS formula as

$$\widehat{P}_{k+1} = -\frac{\boldsymbol{s}_k \boldsymbol{s}_k^\top}{\boldsymbol{s}_k^\top \boldsymbol{r}_k} + \quad I - \frac{\boldsymbol{s}_k \boldsymbol{r}_k^\top}{\boldsymbol{s}_k^\top \boldsymbol{r}_k} \Bigg) \widehat{P}_k \left(I - \frac{\boldsymbol{r}_k \boldsymbol{s}_k^\top}{\boldsymbol{s}_k^\top \boldsymbol{r}_k} \right). \tag{5}$$

For details of the development of this formula, see [5].

2.1. Theoretical properties of the BFGS preconditioner

Let us define the difference between the approximated and the exact eigenvector $\boldsymbol{e}_k = \boldsymbol{v}_1 - \boldsymbol{u}_k$ and the difference between the preconditioned Jacobian and the identity matrix as

$$E_k = I - J_k^{1/2} P_k J_k^{1/2}.$$

The following theorem will state the so-called *bounded deterioration* [15] of the preconditioner at step $k+1$ with respect to that of step k, namely that the distance of the preconditioned matrix from the identity matrix at step $k+1$ is less or equal than that at step k plus a constant that may be small as desired, depending on the closeness of $\boldsymbol{u}_0$ to the exact solution.

Theorem 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

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

Proof
See [5]. □

2.2. Computing several eigenpairs

When seeking an eigenvalue different from λ_1, say λ_j, the projected Jacobian matrix at iteration k changes, like in the JD algorithm [16], as

$$J_k = \left(I - Q_k Q_k^\top\right)(A - \theta_k I)\left(I - Q_k Q_k^\top\right)$$

where $Q_k = \left[\boldsymbol{v}_1\ \boldsymbol{v}_2 \ldots \boldsymbol{v}_{j-1}\ \boldsymbol{u}_k\right]$ is the matrix whose first $j-1$ columns are the previously computed eigenvectors. Analogously, also the preconditioner must be chosen orthogonal to Q_k as

$$P_k = \left(I - Q_k Q_k^\top\right) \widehat{P}_k \left(I - Q_k Q_k^\top\right). \tag{6}$$

2.3. Choice of the initial Newton vector

An important issue in the efficiency of the Newton approach for eigenvalue computation is represented by the appropriate choice of the initial guess. To properly start the Newton iteration 'sufficiently' close to the exact eigenvector, we propose to perform some preliminary iterations of the DACG [17, 18] eigensolver, which is based on the preconditioned conjugate gradient (PCG) (nonlinear) minimization of the Rayleigh quotient. This method has proven very robust, and not particularly sensitive to the initial vector, in the computation of a few eigenpairs of large SPD matrices.

Algorithm 1 Computation of $\boldsymbol{c} = P_k \boldsymbol{g}_l$ for the BFGS preconditioner.

INPUT: Vector $\boldsymbol{g}_l$, vectors $\boldsymbol{r}_s, \boldsymbol{s}_s$ and scalar products $\alpha_s = \boldsymbol{s}_s^\top \boldsymbol{r}_s,\ s = 1, \ldots, k-1$.

$\boldsymbol{w} = \boldsymbol{g}_l$

FOR $s := k-1$ TO 0
1. $a_s := \boldsymbol{s}_s^\top \boldsymbol{w} / \alpha_s$
2. $\boldsymbol{w} := \boldsymbol{w} - a_s \boldsymbol{r}_s$

END FOR

$\boldsymbol{c} = \widehat{P}_0 \boldsymbol{w}$

FOR $s := 0$ TO $k-1$
1. $b := \boldsymbol{r}_s^\top \boldsymbol{c} / \alpha_s$
2. $\boldsymbol{c} := \boldsymbol{c} - (a_s + b)\boldsymbol{s}_s$

END FOR

$z := Q_k^\top \boldsymbol{c}$

$\boldsymbol{c} := \boldsymbol{c} - Q_k z$

2.4. Implementation of the BFGS preconditioner update

At a certain nonlinear iteration level k of the Krylov subspace method of choice, we need to apply the preconditioner, that is, to compute $\boldsymbol{c} = P_k \boldsymbol{g}_l$, where $\boldsymbol{g}_l$ is the residual of the linear system at iteration l. Let us suppose we compute an initial preconditioner P_0. Then, at the initial nonlinear iteration $k = 0$, we simply have $\boldsymbol{c} = P_0 z_l$. At step k, the preconditioner $\widehat{P}_k$ is defined recursively by (5), while P_k using (6) can be written as

$$
\begin{aligned}
P_k &= \left(I - Q_k Q_k^\top\right) \widehat{P}_k \left(I - Q_k Q_k^\top\right) = \\
&= \left(I - Q_k Q_k^\top\right) \left\{ \left(I - \frac{\boldsymbol{s}_{k-1} \boldsymbol{r}_{k-1}^\top}{\boldsymbol{s}_{k-1}^\top \boldsymbol{r}_{k-1}} \right) \widehat{P}_{k-1} \quad I - \frac{\boldsymbol{r}_{k-1} \boldsymbol{s}_{k-1}^\top}{\boldsymbol{s}_{k-1}^\top \boldsymbol{r}_{k-1}} \right) - \frac{\boldsymbol{s}_{k-1} \boldsymbol{s}_{k-1}^\top}{\boldsymbol{s}_{k-1}^\top \boldsymbol{r}_{k-1}} \Bigg\} \left(I - Q_k Q_k^\top\right).
\end{aligned}
\tag{7}
$$

To compute vector $\boldsymbol{c}$, first we observe that $\boldsymbol{g}_l$ is orthogonal to Q_k, so there is no need to apply matrix $I - Q_k Q_k^\top$ on the right of (7). Application of preconditioner $\widehat{P}_k$ to the vector $\boldsymbol{g}_l$ can be performed at the price of $2k$ dot products and $2k$ daxpys as sketched in Algorithm 1. The scalar products $\alpha_s = \boldsymbol{s}_s^\top \boldsymbol{r}_s$, which appear at the denominator of $\widehat{P}_k$, can be computed once and for all before starting the solution of the k-th linear system. Last, the obtained vector $\boldsymbol{c}$ must be orthogonalized against the columns of Q_k by a classical Gram–Schmidt procedure.

2.5. PCG solution of the correction equation

As a Krylov subspace solver for the correction equation, we chose the PCG method because 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 previous 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, whenever the current solution $\boldsymbol{x}_l$ satisfies

$$
\|\boldsymbol{r}_{k,l}\| = \left\| A\boldsymbol{x}_l - \frac{\boldsymbol{x}_l^\top A \boldsymbol{x}_l}{\boldsymbol{x}_l^\top \boldsymbol{x}_l} \boldsymbol{x}_l \right\| < \tau \left(\boldsymbol{x}_l^\top A \boldsymbol{x}_l \right) \tag{8}
$$

or when the decrease of $\|\boldsymbol{r}_{k,l}\|$ is slower than the decrease of $\|\boldsymbol{g}_l\|$, 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 because the correction equation is not solved 'exactly', that is, 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 is described here in Algorithm 1.

2.6. Limited memory variant of the DACG–Newton method

The BFGS preconditioner defined in Algorithm 1 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 a matrix V with n rows and $2 \times k_{\max}$ columns. The implementation of our DN method for computing the leftmost eigenpairs of large SPD matrices is described in Algorithm 2.

Algorithm 2 DACG-Newton Algorithm.

- INPUT:
  1. Matrix A;
  2. number of sought eigenpairs n_{eig};
  3. tolerance and maximum number of its for the outer iteration: τ, ITMAX;
  4. tolerance for the initial eigenvector guess τ_{DACG};
  5. tolerance and maximum number of its for the inner iteration: τ_{PCG}, MAX_{PCG}
  6. parameters for the IC preconditioner:, LFIL and τ_{IC};
  7. maximum allowed rank-two update in the BFGS preconditioner: $k_{\max}$.
- $\tilde{Q} := [\,]$.
- Compute an incomplete Cholesky factorization of A: $\widehat{P}_0$ with parameters LFIL and τ_{IC}.
- FOR $j := 1$ TO n_{eig}
  1. Choose $\boldsymbol{x}_0$ such that $\tilde{Q}^\top \boldsymbol{x}_0 = 0$.
  2. Compute $\boldsymbol{u}_0$, an approximation to $\boldsymbol{v}_j$ by the DACG procedure with initial vector $\boldsymbol{x}_0$, preconditioner $\widehat{P}_0$ and tolerance τ_{DACG}.
  3. $k := 0$, $\theta_k := \boldsymbol{u}_k^\top A \boldsymbol{u}_k$.
  4. WHILE $\|A\boldsymbol{u}_k - \theta_k \boldsymbol{u}_k\| > \tau\theta_k$ AND $k <$ IMAX DO
     1. $Q := [\tilde{Q}\ \boldsymbol{u}_k]$.
     2. Solve $J_k \boldsymbol{s}_k = -\boldsymbol{r}_k$ for $\boldsymbol{s}_k \perp Q$ by the PCG method with preconditioner P_k and tolerance τ_{PCG}.
     3. $\boldsymbol{u}_{k+1} := \dfrac{\boldsymbol{u}_k + \boldsymbol{s}_k}{\|\boldsymbol{u}_k + \boldsymbol{s}_k\|},\ \theta_{k+1} = \boldsymbol{u}_{k+1}^\top A \boldsymbol{u}_{k+1}$.
     4. $k_1 = k$ MOD $k_{\max}$; $V(*, 2k_1 + 1) := \boldsymbol{s}_k,\ V(*, 2k_1 + 2) := \boldsymbol{r}_k$,
     5. $k := k + 1$
  5. END WHILE
  6. Assume $\boldsymbol{v}_j = \boldsymbol{u}_k$ and $\lambda_j = \theta_k$. Set $\tilde{Q} := [\tilde{Q}\ \boldsymbol{v}_j]$
- END FOR

The previously described implementation is well suited to parallelization provided that an efficient matrix–vector product (MVP) routine is available. 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 [20].

3.1. Description of implicitly restarted Lanczos method

The best known method, the IRA method, is implemented within the ARPACK package [21] and is also available in the most popular scientific computing packages. For SPD matrices, the IRA method simplifies to the IRL method, which reduces the computational cost, by taking advantage of the symmetry of the problem.

The idea of the Lanczos method is to project the coefficient matrix A onto a subspace generated by an arbitrary initial vector $\boldsymbol{v}_1$ and the matrix A itself, known as Krylov subspace. In particular, a Krylov subspace of dimension m is generated by the following set of independent vectors:

$$\boldsymbol{v}_1, A\boldsymbol{v}_1, \cdots, A^{m-1}\boldsymbol{v}_1.$$

Actually, it is convenient to work with an orthogonal counterpart of this basis and to organize its vectors as columns of a matrix Q_m. After the Gram–Schmidt process, a triangular matrix T_m satisfying $T_m = Q_m^\top A Q_m$ is computed. It is well known that the largest eigenvalues of T_m, $\lambda_n^{(m)}, \lambda_{n-1}^{(m)}, \cdots$ converge, as the size of the Krylov subspace m increases, to the largest eigenvalues of A: $\lambda_n, \lambda_{n-1}, \cdots$, while the corresponding eigenvectors of A can be computed from the homologous eigenvectors of T_m by $\boldsymbol{u}_i = Q_m \boldsymbol{u}_i^{(m)}$.

A subspace of dimension NCV $\in \left[n_{eig}, 3n_{eig}\right]$ is usually sufficient to assess a small number n_{eig} of the rightmost eigenpairs to a satisfactory accuracy. The ratio between the number of MVPs and n_{eig} is also known to decrease when n_{eig} is increasing. This eigenvalue solver exits whenever the following test is satisfied:

$$\sum_{k=1}^{n_{eig}} \frac{1}{n_{eig}} \frac{\|A\boldsymbol{u}_k - \lambda_k \boldsymbol{u}_k\|}{\lambda_k} \leqslant \tau,$$

with τ a fixed tolerance.

Convergence to the smallest eigenvalues is much slower. Hence, to compute the leftmost part of the spectrum, it is more usual to apply the Lanczos process to the inverse of the coefficient matrix A^{-1}. Because A is expected to be large and sparse, its explicit inversion is not convenient from both CPU time and storage point of view. This implies that at every stage of the Arnoldi process, a linear system involving matrix A, namely, $A\boldsymbol{q}_k = \boldsymbol{y}$, has to be solved by an iterative method properly preconditioned.

We also adopted the implicit restart strategy as described in [21], a technique to combine the implicitly shifted QR scheme with a k-step Lanczos factorization and obtain a truncated form of the implicitly shifted QR iteration. Implicit restart provides a means to extract interesting information from large Krylov subspaces while avoiding the storage and numerical difficulties associated with the standard approach. It does this by continually compressing the interesting information into a fixed-size k-dimensional subspace. This is accomplished through the implicitly shifted QR mechanism. A Lanczos factorization of length $m = k + p$, $AQ_m = Q_m T_m + r_m e_m$, is compressed to a factorization of length k that retains the eigeninformation of interest.

3.2. Relaxed tolerances for the inner linear systems

In order to speed-up the inner linear system solution, a number of relaxation strategies have been proposed by several authors. This approach has been first analyzed in [22] and [23]. We adopt the strategy described in Equation (3.15) in [7] where the authors suggest to stop the PCG iteration using a variable tolerance, which is set to be proportional to the 'distance' between the converged and the spurious eigenvalues. In this way, a sequence of increasing tolerances is defined, after the first restart, which implies a decreasing number of iterations during the IRA process.

3.3. Preconditioners for the CG solution of the linear system

At step k of the IRL procedure, a linear system involving matrix A should be iteratively solved, which provides the k-th vector of the Lanczos basis:

$$A\boldsymbol{q}_k = y.$$

Because A is SPD, we employ the PCG method for its solution. This inner iterative procedure is stopped when the relative residual norm is smaller than τ_{PCG}, which is to be chosen some orders of magnitude smaller than the tolerance prescribed for the eigenproblem.

The tuned preconditioners $\mathcal{P}_k$, which we will describe in the sequel, are a low-rank modification of a given preconditioner for A, in our case the inverse of the incomplete Cholesky factorization with variable fill-in. If we call this initial preconditioner P, $\mathcal{P}_k$ is defined as

$$\mathcal{P}_k = P + B_k, \qquad \text{with } B_k \text{ a low-rank matrix.}$$

The tuned preconditioners $\mathcal{P}_k$ must satisfy the condition $\mathcal{P}_k A Q_k = Q_k$, which gives as a consequence that the preconditioned matrix $\mathcal{P}_k A$ has the eigenvalue 1 with (at least) multiplicity k.

3.3.1. Freitag–Spence (FS) preconditioner. The first tuned preconditioner is a block generalization of the preconditioner proposed in [24], in the framework of the Rayleigh quotient iteration. To solve the k-th linear system of the IRL process, we propose a block tuned preconditioner of the form

$$\mathcal{P}_k^{(FS)} = P - Z\left(Z^\top A Q_k\right)^{-1} Z^\top,$$

where $Z = PAQ_k - Q_k$.

This preconditioner satisfies $\mathcal{P}_k^{(FS)} A Q_k = Q_k$; it is well defined only if $H_k = (PAQ_k - Q_k)^\top A Q_k$ is nonsingular. Singularity may occur if $(PA - I)Q_k$ is column rank deficient. In this case, $\lambda = 1$ would be an eigenvalue of PA corresponding to a linear combination of the columns of Q_k. Our code explicitly checks if $(PAQ_k - Q_k)^\top A Q_k$ is nonsingular. If this check fails, we use the previous low-rank matrix Q_{k-1} instead of Q_k.

Also, the property of the tuned preconditioner to be SPD depends on matrix H_k, namely, on the norm of its inverse, as well as on the goodness of the initial preconditioner. In fact, setting $N = ZH_k^{-1}Z^\top$, we have

$$\begin{aligned} N &= (PAQ_k - Q_k)\left(Q_k^\top APAQ_k - Q_k^\top AQ_k\right)^{-1}(PAQ_k - Q_k)^\top = \\ &= (PA - I)Q_k H_k^{-1} Q_k^\top (PA - I)^T. \end{aligned}$$

From which, after denoting by h_k the norm of the inverse of H_k,

$$\|N\| \leqslant \|PA - I\|^2 h_k.$$

From the standard perturbation theory (e.g., Theorem 7.2.2 in [25]), we have that

$$\lambda_{\min}\left(\mathcal{P}_k^{(FS)}\right) \geqslant \lambda_{\min}(P) - \|N\| \geqslant \lambda_{\min}(P) - h_k \|PA - I\|^2.$$

If h_k is small and the initial preconditioner P is a good preconditioner for A (and hence $\|I - PA\|$ is small), it is likely that $\mathcal{P}_k^{(FS)}$ remains SPD. We finally remark that in every Lanczos iteration and every test problem considered, we found that $\mathcal{P}_k^{(FS)}$ was actually SPD.

In addition to the tuning property just defined, it has been proved in [7] that the tuned preconditioners have the desirable property to cluster around one of the eigenvalues of the preconditioned matrix.

3.3.2. BFGS-like preconditioner. A BFGS-like preconditioner, similar to that described in Section 2 for the DN methods, can be written as follows (e.g., [26]):

$$\mathcal{P}_k^{(BFGS)} = W + (I - WA)P(I - AW), \qquad \text{where} \qquad W = Q_k \left(Q_k^\top A Q_k\right)^{-1} Q_k^\top.$$

Also, $\mathcal{P}_k^{(BFGS)}$ is a tuned preconditioner. In fact,

$$WAQ_k = Q_k \left(Q_k^\top A Q_k\right)^{-1} Q_k^\top A Q_k = Q_k.$$

Hence, $(I - AW)AQ_k = AQ_k - AQ_k = 0$; it finally follows that

$$\mathcal{P}_k^{(BFGS)} AQ_k = WAQ_k + (I - WA)P(I - AW)AQ_k = Q_k.$$

The sequence $\mathcal{P}_k^{(BFGS)}$ provides all SPD matrices if the initial preconditioner P is so. In fact, for every nonzero vector $\boldsymbol{w} \in \mathcal{R}^n$,

$$\begin{aligned} \boldsymbol{w}^\top \mathcal{P}_k^{(BFGS)} \boldsymbol{w} &= \boldsymbol{w}^\top \left(W + (I - WA)P(I - AW)\right) \boldsymbol{w} \\ &= \boldsymbol{w}^\top W \boldsymbol{w} + \boldsymbol{w}^\top (I - WA)P(I - AW)\boldsymbol{w} \\ &= \boldsymbol{w}^\top W \boldsymbol{w} + z^\top P z = \gamma_W + \gamma_P. \end{aligned}$$

Scalars γ_W and γ_P are both nonnegative. γ_W is zero if and only if $Q_k^\top \boldsymbol{w} = 0$, in which case $z = (I - AW)\boldsymbol{w} = \boldsymbol{w}$; hence, $\gamma_P = z^\top P z = \boldsymbol{w}^\top P \boldsymbol{w} > 0$ because P is SPD.

3.3.3. A simplified preconditioner. A variant of $\mathcal{P}_k^{(BFGS)}$, which we will denote as sBFGS preconditioner, is obtained by setting

$$\mathcal{P}_k^{(sBFGS)} = W + P = P + Q_k \left(Q_k^\top A Q_k\right)^{-1} Q_k^\top.$$

Application of this preconditioner requires less computational effort with respect to the previously defined preconditioners, especially if compared with the complete BFGS preconditioner; in addition, it is always SPD for every choice of matrix Q_k. This preconditioner does not satisfy the property $P_k A Q_k = Q_k$; hence, it cannot be defined a tuned preconditioner; however, it can be proved that the eigenvalue distribution of the preconditioned matrix is more favorable than that of PA as stated by Theorem 2. First, note that the preconditioned matrix $P_k^{(sBFGS)} A$ is similar to $A^{1/2} P_k^{(sBFGS)} A^{1/2} \equiv A_k$. Therefore, we investigate the spectral distribution of A_k through its Rayleigh quotient. Let us denote the initial preconditioned matrix as $A_0 = A^{1/2} P A^{1/2}$.

Theorem 2
For every unit norm vector $\boldsymbol{w}$, there exists a nonnegative real number $\alpha(\boldsymbol{w})$ such that

$$q(A_k, \boldsymbol{w}) = q(A_0, \boldsymbol{w}) + \alpha(\boldsymbol{w}).$$

Proof

$$\begin{aligned} q(A_k, \boldsymbol{w}) &= \boldsymbol{w}^\top A_k \boldsymbol{w} = \boldsymbol{w}^\top A^{1/2} P_k A^{1/2} \boldsymbol{w} \\ &= q(A_0, \boldsymbol{w}) + \boldsymbol{w}^\top A^{1/2} Q_k \left(Q_k^\top A Q_k\right)^{-1} Q_k^\top A^{1/2} \boldsymbol{w} \\ &= q(A_0, \boldsymbol{w}) + \boldsymbol{w}^\top Z \left(Z^\top Z\right)^{-1} Z^\top \boldsymbol{w}. \end{aligned} \tag{9}$$

Now, let us decompose vector $\boldsymbol{w}$ as a sum of one component belonging to the subspace $\mathcal{Z}$ spanned by the columns of Z and the other belonging to $\mathcal{Z}^\perp$ as $\boldsymbol{w} = \boldsymbol{w}_Z + \boldsymbol{w}_\perp = Z\boldsymbol{t} + Z_\perp \boldsymbol{t}_\perp$. Then we can write

$$q(A_k, \boldsymbol{w}) = q(A_0, \boldsymbol{w}) + \boldsymbol{t}^\top Z^\top Z \boldsymbol{t} = q(A_0, \boldsymbol{w}) + \|\boldsymbol{w}_Z\|^2 = q(A_0, \boldsymbol{w}) + \alpha(\boldsymbol{w}). \tag{10}$$

The statement of the theorem implies that A_k may have a more favorable eigenvalue distribution than A_0, because the eigenvalues of A_k are expected to be further away from zero, and particularly so for the smallest eigenvalues of the preconditioned matrices. Moreover, it can be easily shown that if $\boldsymbol{w} \in \mathcal{Z}$ is an eigenvector of A_0 with eigenvalue λ, then it is also an eigenvector of A_k with eigenvalue $\lambda + 1$.

□

3.4. Limited memory variants

As in the BFGS preconditioner for the Newton's method, we want to keep fixed the maximum number (parameter $l_{\max}$) of the low-rank updates defined by the previous preconditioners. Instead of using all the columns of matrix Q_k generated by the IRL process, we use an $n \times l_{\max}$ matrix $\tilde{Q}$ defined as

$$\tilde{Q} = \left[\boldsymbol{q}_j, \boldsymbol{q}_{j+1}, \cdots, \boldsymbol{q}_k\right], \qquad j = \max\{1, k - l_{\max} + 1\}.$$

4. NUMERICAL RESULTS

In this section, we provide numerical results where the performances of DN algorithm for various $k_{\max}$ values and the tuned IRL method with different tuning strategies are tested. We also compared both tuned IRL and DN methods with the JD, as described in [16] and further analyzed and developed in [19, 27, 28]. In the present paper, we implemented JD following the suggestion in [19]; that is, we made use of the PCG method as the inner solver, with the $\widehat{P}_0$ as the preconditioner, which is kept fixed throughout the Newton iteration. Also, the exit tests used in the two methods (DN and JD) are identical for both the outer iteration and the inner PCG solver.

We tested the described algorithms in the computation of the 20 smallest eigenpairs of a number of small-sized to large-sized matrices arising from various realistic applications.

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

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

In most of the runs, unless differently specified, we selected the values of the parameters as reported in Table II for the three solvers. As every eigenvalue solver needs to solve a number of linear systems at each outer iteration, also a number of additional parameters should be fixed depending on the eigensolver as reported in Table III. We remark that the values of the tolerances may not be fixed during the iteration process. For instance, in the IRL method, the inner tolerance is relaxed toward

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

Matrix	Where it comes from	n	nz
`trino`	3D-FE discretization of flow in porous media	4560	64,030
`monte-carlo`	2D-(M)FE stochastic PDE	77,120	384,320
`mat268515`	3D-FE discretization of flow in porous media	268,515	3,926,823
`emilia-923`	3D-FE elasticity problem	923,136	41,005,206

Table II. Default values of parameters.

Number of eigenpairs	$n_{eig} = 20$	
Outer iteration	$\tau = 10^{-8}$,	IMAX $= 100$
Initial preconditioner	LFIL $= 20$,	$\tau_{IC} = 10^{-3}$.

Table III. Default values of PCG parameters.

Solver	τ_{PCG}	MAX_{PCG}
IRL	10^{-10}	200
Jacobi–Davidson	10^{-2}	20, 30
DACG–Newton	10^{-2}	20

IRL, implicitly restarted Lanczos; PCG, preconditioned conjugate gradient.

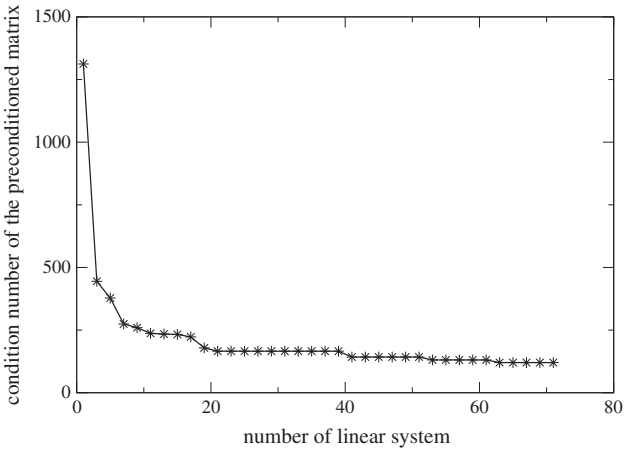

Figure 1. Condition number of the preconditioned matrix with sBFGS tuning strategy for the `trino` problem.

the end of the procedure, as explained in Section 3.4. Also, in the JD and DN methods, the CG iteration may stop before the relative residual is below τ_{PCG} whenever the exit test (8) is satisfied.

The CPU times are expressed in seconds and were obtained by running a Fortran 90 code on a 2× Intel Xeon CPU E5645 at 2.40 GHz (six core) and with 4 GB RAM for each core.

4.1. Spectral distribution of the preconditioned matrix with sBFGS tuning strategy

We will first experimentally analyze the spectral distribution of the preconditioned matrix with the tuning strategy proposed in Section 3.3.3. To this end, we explicitly computed all the eigenvalues of the preconditioned matrix $P_k^{(sBFGS)}A$ at each IRL step for the smallest matrix `trino` starting from a very sparse initial IC preconditioner with parameters LFIL $= 10$, $\tau_{IC} = 10^{-1}$. This choice produced an initial preconditioner with relative fill-in $\sigma = 0.21$. In Figure 1, we report the condition number, that is, the ratio between the largest and the smallest eigenvalue, of $P_k^{(sBFGS)}A$ as a function of the linear system counter. It is found that the condition number is a nonincreasing function of the IRL step number. It decreases from a value of 1312 (initial linear system) to the value of 120.4 for the last linear systems.

4.2. Results with matrix `monte-carlo`

We report in Tables IV and V the results in terms of outer iterations (OUT), matrix-vector products (MVP) and total CPU time of the three methods: JD, tuned IRL, and the DN methods with BFGS preconditioner, respectively. The initial preconditioner's fill-in is $\sigma = 2.30$.

For this test problem, we notice that the JD performance is mildly dependent on the values of the minimum ($m_{\min}$) and maximum ($m_{\max}$) dimension of the subspace generated by the Davidson algorithm. The other parameters were kept fixed as set in Tables II and III.

Regarding the tuned IRL method, Table IV accounts for the important improvement provided by tuning with respect to keeping fixed the initial IC preconditioner. This is so irrespectively of the

Table IV. Timings and iterations for the computation of $n_{eig} = 20$ eigenpairs of matrix `monte-carlo`.

						Prec type	$l_{\max}$	MVP	CPU
MAX_{PCG}	$m_{\min}$	$m_{\max}$	OUT	MVP	CPU	IC	–	7195	50.67
20	10	20	130	2256	24.40	FS	1	6953	51.45
20	15	20	131	2158	24.75	FS	10	4740	40.27
20	20	30	132	2110	25.50	FS	20	4144	40.04
20	25	35	132	2095	26.15	FS	30	3927	41.73
20	25	40	131	2116	28.00				
30	10	20	121	2328	24.90	BFGS	1	6890	52.19
30	15	20	122	2297	25.75	BFGS	10	4633	45.68
30	20	30	121	2207	26.09	BFGS	20	3923	46.44
30	25	35	122	2236	27.50	BFGS	30	3603	49.40
30	25	40	119	2255	27.39				
						sBFGS	1	7024	51.64
						sBFGS	5	6328	46.52
						sBFGS	10	4834	40.77
						sBFGS	20	4158	39.41
						sBFGS	30	3932	40.65

On the left, we report the Jacobi–Davidson results; on the right table, we report the implicitly restarted Lanczos results where NCV = 40 and OUT = 80 for every run.

Table V. Timings and iterations with the DACG–Newton method for the computation of $n_{eig} = 20$ eigenpairs of matrix `monte-carlo`.

				CPU		
τ_{DACG}	$k_{\max}$	OUT	MVP	DACG	Newton	Total
0.1	0	195	4395	8.23	32.48	40.71
0.1	1	141	3339	8.08	23.73	31.80
0.1	5	112	2702	8.06	18.42	26.49
0.1	10	112	2701	8.09	18.39	26.47
0.01	0	149	3991	11.81	25.37	37.18
0.01	1	118	3332	11.66	21.11	32.77
0.01	5	96	2880	11.68	16.10	27.78
0.01	10	96	2880	11.61	16.00	27.61

tuning strategy. The optimal $l_{\max}$ parameter is between 15 and 20, with the FS and sBFGS tuning variants slightly more performing than the full BFGS one, because of the larger cost of application of the BFGS tuning preconditioner. As for the DN method with BFGS preconditioner, the number of low-rank updated $k_{\max}$ greatly influences its performance. With $k_{\max} = 10$, the DN method displays a comparable performance as JD both in terms of total MVP and overall CPU time. The IRL method with both tuning and relaxed-tolerance strategy again needs more MVP and CPU time than the other two solvers.

4.3. Matrix mat268515

For matrix `mat268515`, we consider two different initial preconditioners:

(1) **CASE #1**: LFIL $= 20$, $\tau_{IC} = 10^{-3} \implies \sigma = 2.67$.
(2) **CASE #2**: LFIL $= 20$, $\tau_{IC} = 10^{-2} \implies \sigma = 0.87$.

4.3.1. CASE #1. Numerical results. To clarify the importance of the tuning and relaxation strategies in the IRL method, we first report in Figure 2 a comparison of the number of inner iterations using

either a fixed preconditioner or the tuning preconditioner with $l_{\max} = 20$ in the computation of 20 eigenpairs with NCV $= 40$. Notice that the combination of tuning (from the second IRL step) and relaxation (which starts from the first restart, i.e., after 40 steps) reduces the linear iterations from the initial 70–80 to the final 25.

The results corresponding to CASE #1 are summarized in Tables VI and VII, which account for the number of outer iterations, MVP, and CPU times for JD, tuned IRL, and DN methods with BFGS preconditioner, respectively.

Once again, JD and DN outperform the tuned IRL method for each tuning strategy and $l_{\max}$ value. With this relatively dense preconditioner, however, the influence of the BFGS acceleration in the DN method is modest.

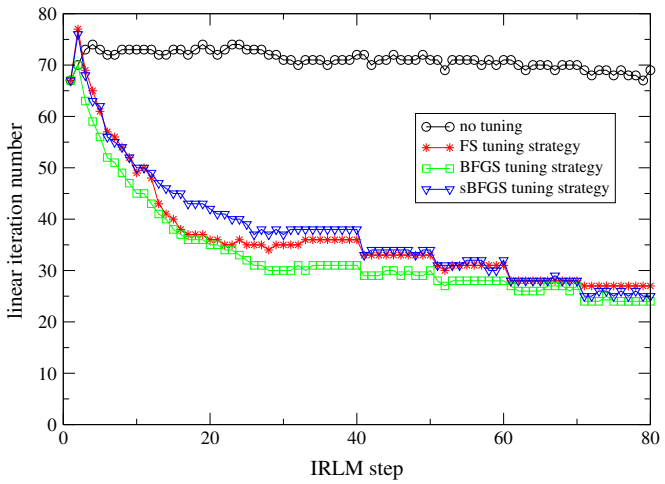

Figure 2. Number of iterations for the preconditioned conjugate gradient solution of the inner systems with implicitly restarted Lanczos in the solution of the `mat268515` test problem. Comparison against the fixed preconditioner of CASE #1 and the three tuned preconditioners with $l_{\max} = 20$.

Table VI. Timings and iterations for the computation of $n_{eig} = 20$ eigenpairs of matrix `mat268518`.

						Prec type	$l_{\max}$	MVP	CPU
MAX_{PCG}	$m_{\min}$	$m_{\max}$	OUT	MVP	CPU	IC	–	5685	265.36
20	10	20	114	1624	99.86	FS	1	5630	247.36
20	15	20	111	1577	109.49	FS	10	3485	184.99
20	20	30	114	1596	110.29	FS	15	3112	178.68
20	25	35	114	1578	110.12	FS	20	3034	180.57
20	25	40	116	1568	116.65	FS	30	2912	184.86
30	10	20	106	1738	116.01				
30	15	20	108	1604	103.12	BFGS	1	5325	258.82
30	20	30	102	1605	105.54	BFGS	10	3295	191.15
30	25	35	108	1585	106.51	BFGS	15	2936	183.47
30	25	40	105	1627	108.39	BFGS	20	2828	185.20
						BFGS	30	2653	189.26
						sBFGS	1	5435	260.29
						sBFGS	10	3471	169.91
						sBFGS	15	3193	178.86
						sBFGS	20	3096	178.62
						sBFGS	30	3009	183.17

On the left, we report the Jacobi–Davidson results, and on the right table, we report the implicitly restarted Lanczos results where NCV $= 40$ and OUT $= 80$ for every run. The initial preconditioner is set as in CASE #1.

Table VII. Timings and iterations with the DACG–Newton method for the computation of $n_{eig} = 20$ eigenpairs of matrix `mat268515`.

				CPU		
τ_{DACG}	$k_{\max}$	OUT	MVP	DACG	Newton	Total
0.02	0	105	2236	44.56	74.26	118.82
0.02	1	100	2051	44.30	67.51	111.81
0.02	5	96	2880	44.93	61.22	106.15
0.02	10	96	2880	44.53	60.84	105.37

The initial preconditioner is set as in CASE #1.

Table VIII. Comparison of all the methods considered for the computation of $n_{eig} = 20$ eigenpairs of matrix `mat268515` using the sparse initial preconditioner of CASE # 2.

Method	Parameters			OUT	MVP	CPU
DACG–Newton	τ_{DACG}	$k_{\max}$				
	0.02	0		256	6425	224.31
	0.02	1		151	4196	155.99
	0.02	5		119	3538	135.17
	0.02	10		119	3538	135.70
Jacobi–Davidson	$m_{\min}$	$m_{\max}$	MAX_{PCG}			
	10	20	20	143	2237	99.04
	15	20	20	141	2314	106.31
	25	30	20	139	2300	110.97
	25	35	20	139	2308	113.40
	10	20	30	134	2670	117.46
	15	20	30	135	2698	119.30
	20	30	30	132	2660	120.47
	25	35	30	133	2688	124.31
IRL	Prec type	$l_{\max}$	NCV			
	IC	–	30	75	9532	255.11
	FS	5	30	75	6774	210.10
	FS	10	30	75	5684	182.81
	FS	20	30	75	5018	190.42
	FS	25	30	75	4832	190.72
	BFGS	5	30	75	6569	217.15
	BFGS	10	30	75	5452	191.24
	BFGS	20	30	75	4663	194.36
	BFGS	25	30	75	4511	202.81
	sBFGS	5	30	75	6708	192.04
	sBFGS	10	30	75	5579	186.21
	sBFGS	20	30	75	4806	168.91
	sBFGS	25	30	75	4696	175.16

4.3.2. CASE #2. Numerical results. In Table VIII, we report a number of selected runs of the three methods. It is worth noticing that the BFGS acceleration of DN is much more evident here than with the denser initial preconditioner. The same consideration applies with the tuning strategy within IRL. In this test case, the new approximated tuning strategy sBFGS reveals the most efficient one.

4.4. Matrix `emilia-923`

We now report the results obtained in eigensolving the largest test matrix, which arises from the regional geomechanical model of a deep hydrocarbon reservoir. This matrix is obtained by discretizing the structural problem with tetrahedral finite elements. Because of the complex geometry of the geological formation, it was not possible to obtain a computational grid characterized by regularly shaped elements. This matrix is publicly available in the University of Florida Sparse Matrix Collection at `http://www.cise.ufl.edu/research/sparse/matrices`.

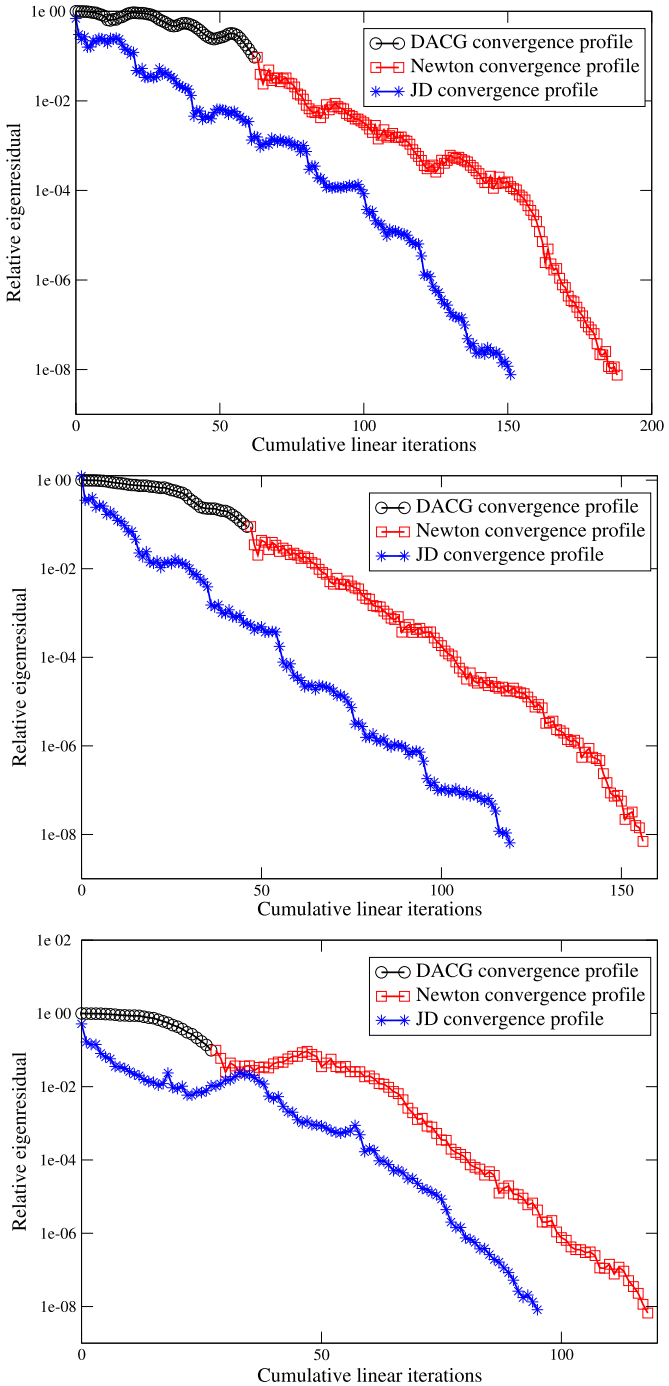

Figure 3. Convergence history (relative residual norm for the eigenvector versus cumulative linear iteration number) for problem `emilia-923` to assess eigenpairs $(\lambda_j, \boldsymbol{v}_j)$, $j = 4, 10$, and 15, from top to bottom, respectively.

For this realistic test case, the performances of JD and DN with BFGS updates well compare in terms of total CPU time. This is also accounted for by Figure 3 where, for a selected number of eigenpairs, we plot the relative residual norm of the eigenvector, namely,

$$\frac{\|A\boldsymbol{x} - q(\boldsymbol{x})\boldsymbol{x}\|}{q(\boldsymbol{x})},$$

at each **linear** iteration, irrespective of the number of linear systems solved.

Table IX. Comparison of all the methods considered for the computation of $n_{eig} = 20$ eigenpairs of matrix `emilia-923`.

				OUT	MVP	CPU
DACG–Newton	τ_{DACG}	$k_{\max}$	MAX_{PCG}			
	0.1	10	20	115	3007	990.94
Jacobi–Davidson	$m_{\min}$	$m_{\max}$	MAX_{PCG}			
	15	25	20	140	2346	885.08
IRL	Prec type	$l_{\max}$	NCV			
($\tau_{PCG} = 10^{-13}$)	IC	–	50	81	8135	2123.45
	FS	30	50	81	5623	1755.48
	BFGS	30	50	81	5157	1991.58
	sBFGS	30	50	81	5662	1744.69

The fill-in of the initial preconditioner is $\sigma = 1.86$.

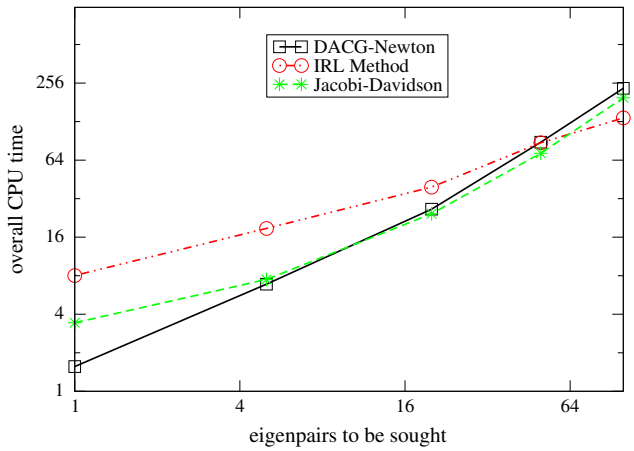

Figure 4. CPU times in seconds versus number of eigenpairs for DACG–Newton, Jacobi–Davidson, and implicitly restarted Lanczos (IRL) methods. Matrix `monte-carlo`.

From the figure, we appreciate the similar convergence profile of the two methods. Apart from the first eigenpair (plot not displayed), for which JD takes much time to enlarge the search subspace and hence is slower than DN, generally JD takes a few iterations less than DN. We remark, however, that the initial DACG iterations are less costly than both the subsequent Newton and JD ones.

It is finally worth mentioning that the 'best' IRL takes roughly twice the CPU time needed by JD and DN (Table IX). This is partly due to the ill-conditioning of matrix `emilia-923`, which requires to solve the inner linear system at a very high accuracy (we had to set $\tau_{PCG} = 10^{-13}$ in order to have the desired accuracy on the eigenvectors).

4.5. Computing a larger number of eigenpairs

The results presented so far show that both JD and DN with low-rank preconditioner updates outperform the IRL method with tuning and relaxation in the computation of 20 eigenpairs. However, if the number of eigenpairs to be sought is larger, the performance of the Newton-based methods namely, JD and DN, is expected to worsen because of excessive reorthogonalization times with respect to the previously computed eigenvectors.

To support this statement, we provide in Figure 4 a picture reporting the CPU times in computing $N_{eig} = 1, 5, 20, 50$, and 100 eigenpairs of matrix `monte-carlo` for the three methods considered with optimal choice of parameters. As expected, when searching for a low number of eigenpairs, DN is the optimal solver, while JD is the most efficient algorithm in computing a moderate number ($10 \div 80$) eigenpairs. The IRL method is the most performing algorithm starting from 70–80 eigenpairs to be sought. Qualitatively comparable pictures could be provided for each test problem.

5. CONCLUSIONS

We have performed an experimental analysis of two eigensolvers (DN method and IRL method) both equipped by a number of deflated/tuned preconditioners, which accelerate the PCG solver in the solution of the inner linear systems. The use of the described deflated/tuned preconditioners (which are based on low-rank updates of a given preconditioner) greatly improves the performances of both the aforementioned methods. A new approximated tuning strategy is proposed for the IRL method, which is proved to shift the eigenvalues of the preconditioned matrix far from zero and well compares with the tuning preconditioners described in previous papers. Numerical results in the eigensolution of large SPD matrices arising from discretization of PDEs modeling realistic flow and structural problems reveal that DN with optimal BFGS preconditioner displays comparable performances as the JD method. The IRL method, implemented using an inexact variant (i.e., less accurate solution of the linear systems after the first restart) and optimal tuning preconditioners, is shown to be the slowest method on all test problems if a moderate number of eigenpairs is being sought.

ACKNOWLEDGEMENTS

We are indebted to the two anonymous referees whose constructive criticism helped improve the presentation of the paper.

REFERENCES

1. Bathe KJ. *Finite Element Procedures in Engineering Analysis*. Prentice-Hall: Englewood Cliffs, 1982.
2. Saad Y, Stathopoulos A, Chelikowsky J, Wu K, Öğüt S. Solution of large eigenvalue problems in electronic structure calculations. *BIT, Numerical Mathematics* 1996; **36**:563–578. International Linear Algebra Year (Toulouse, 1995).
3. Bergamaschi L, Bozzo E. Computing the smallest eigenpairs of the graph Laplacian . arXiv:1311.1695, 2015. Submitted.
4. Bozzo E, Franceschet M. Approximations of the generalized inverse of the graph Laplacian matrix. *Internet Mathematics* 2012; **8**:1–26.
5. Bergamaschi L, Martínez A. Efficiently preconditioned inexact Newton methods for large symmetric eigenvalue problems. *Optimization Methods & Software* 2015; **30**:301–322.
6. Sleijpen GLG, van der Vorst HA. A Jacobi–Davidson iteration method for linear eigenvalue problems. *SIAM Review* 2000; **42**:267–293 (electronic).
7. Freitag MA, Spence A. Shift-invert Arnoldi's method with preconditioned iterative solves. *SIAM Journal on Matrix Analysis and Applications* 2009; **31**:942–969.
8. Xue F, Elman H. Fast inexact implicitly restarted Arnoldi method for generalized eigenvalue problems with spectral transformation. *SIAM Journal on Matrix Analysis and Applications* 2012; **33**:433–459.
9. Simoncini V, Eldén L. Inexact Rayleigh quotient-type methods for eigenvalue computations. *BIT, Numerical Mathematics* 2002; **42**:159–182.
10. Wu K, Saad Y, Stathopoulos A. Inexact Newton preconditioning techniques for large symmetric eigenvalue problems. *Electronic Transactions on Numerical Analysis* 1998; **7**:202–214.
11. 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**:1863–1873.
12. Bergamaschi L, Bru R, Martínez A, Mas J, Putti M. Low-rank update of preconditioners for the nonlinear Richard's equation. *Mathematical and Computer Modelling* 2013; **57**:1933–1941.
13. Bergamaschi L, Bru R, Martínez A, Putti M. Quasi-Newton preconditioners for the inexact Newton method. *Electronic Transactions on Numerical Analysis* 2006; **23**:76–87.
14. Saad Y. ILUT: a dual threshold incomplete LU factorization. *Numerical Linear Algebra with Applications* 1994; **1**:387–402.
15. Kelley CT. *Iterative Methods for Optimization*, Frontiers in Applied Mathematics, vol. 18. Society for Industrial and Applied Mathematics (SIAM): Philadelphia, PA, 1999.
16. Sleijpen GLG, van der Vorst HA. A Jacobi–Davidson method for linear eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications* 1996; **17**:401–425.
17. Bergamaschi L, Gambolati G, Pini G. Asymptotic convergence of conjugate gradient methods for the partial symmetric eigenproblem. *Numerical Linear Algebra with Applications* 1997; **4**:69–84.
18. Bergamaschi L, Putti M. Numerical comparison of iterative eigensolvers for large sparse symmetric matrices. *Computer Methods in Applied Mechanics and Engineering* 2002; **191**:5233–5247.
19. Notay Y. Combination of Jacobi–Davidson and conjugate gradients for the partial symmetric eigenproblem. *Numerical Linear Algebra with Applications* 2002; **9**:21–44.

20. Bergamaschi L, Martínez A. Parallel RFSAI-BFGS preconditioners for large symmetric eigenproblems. *Journal of Applied Mathematics* 2013:Article ID 767042, 10 pages.
21. Lehoucq RB, Sorensen DC, Yang C. *ARPACK Users Guide. Solution of Large Scale Eigenvalue Problem with Implicit Restarted Arnoldi Methods*. SIAM: Philadelphia, 1998.
22. Bouras A, Fraysse V. A relaxation strategy for the Arnoldi method in eigenproblems. *Technical Report TR/PA/00/16, CERFACS,* Toulouse, France, 2000.
23. Simoncini V. Variable accuracy of matrix-vector products in projection methods for eigencomputation. *SIAM Journal on Numerical Analysis* 2005; **43**:1155–1174.
24. Freitag MA, Spence A. A tuned preconditioner for inexact inverse iteration applied to Hermitian eigenvalue problems. *IMA Journal of Numerical Analysis* 2008; **28**:522–551.
25. Golub GH, van Loan CF. *Matrix Computation*. Johns Hopkins University Press: Baltimore, 1991.
26. Szyld DB, Xue F. Efficient preconditioned inner solves for inexact Rayleigh quotient iteration and their connections to the single-vector Jacobi–Davidson method. *SIAM Journal on Matrix Analysis and Applications* 2011; **32**: 993–1018.
27. Fokkema DR, Sleijpen GLG, van der Vorst HA. Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils. *SIAM Journal on Scientific Computing* 1998; **20**:94–125 (electronic).
28. Stathopoulos A. A case for a biorthogonal Jacobi–Davidson method: restarting and correction equation. *SIAM Journal on Matrix Analysis and Applications* 2002; **24**:238–259 (electronic).