Article

Low-rank Updates of Preconditioners for sequences of symmetric linear systems

Luca Bergamaschi (luca.bergamaschi@unipd.it)

Department of Civil Environmental and Architectural Engineering

University of Padua

1. Introduction

The aim of this survey is to review some recent developements in devising efficient preconditioners for sequences of linear systems $A\mathbf{x} = \mathbf{b}$. Such a problem arise in many scientific applications, such as discretization of transient PDEs, solution of eigenvalue problems, (Inexact) Newton method applied to nonlinear systems, rational Krylov methods for computing a function of a matrix. Full purpose preconditioners such as the Incomplete Cholesky (IC) factorization or approximate inverses are aimed at clustering eigenvalues of the preconditioned matrices around one. In this paper we will analyze a number of techniques of updating a given IC preconditioner (which we denote as P_0 in the sequel) by a low-rank matrix with the aim of further improving this clustering. The most popular low-rank strategies are aimed at removing the smallest eigenvalues (deflation) or at shifting them towards the middle of the spectrum. The low-rank correction is based on a (small) number of linearly independent vectors whose choice is crucial for the effectiveness of the approach. In many cases these vectors are approximations of eigenvectors corresponding to the smallest eigenvalues of the preconditioned matrix $P_0 A$. We will also review some techniques to efficiently approximate these vectors when incorporated within a sequence of linear systems all possibly having constant (or slightly changing) coefficient matrices. Numerical results concerning sequences arising from discretization of linear/nonlinear PDEs and iterative solution of eigenvalue problems show that the performance of a given iterative solver can be very much enhanced by the use of low-rank updates.

1.1. Synopsis

In Section 1 we will discuss various low-rank updates, namely, the deflation technique, the *tuning* strategy, developed based on a revisited *secant condition*. and the *spectral preconditioners*. In Section 2 we will concerned with the computational complexity of the low-rank updates. Section 3 discusses the choice of the vectors which form the low-rank matrices while Section 4 will address the issue to cheaply assessing some of the leftmost eigenpairs of the coefficient matrix. Section 5 is devoted to reporting some numerical experiments in the framework of discretization of transient PDES and solution eigenvalue problems. The conclusions are drawn in Section 6.

2. Low-rank updates

2.1. Deflation

The idea to use deflation to accelerate the Conjugate Gradient method goes back to the middle of the 80's. The most cited article on this subject is a paper by Nicolaides [1] who developed a deflated CG based on a three-term recurrence. Other similar formulations are present in [2] and [3]. Some years later in [4] a deflated GMRES method has been proposed by using an augmented basis. Rigorous bounds on the eigenvalues of the deflated-preconditioned matrices are given in [5]. We finally mention recent interesting survey on deflation methods in [6].

We will follow here the formulation of the Deflated-CG algorithm developed in [7]. Given a full column rank $n \times p$ matrix $W = \begin{bmatrix} \mathbf{w}_1 & \dots & \mathbf{w}_p \end{bmatrix}$ the A-orthogonal projector H is defined as

$$H = I - W(W^TAW)^{-1}W^TA \tag{1}$$

which maintains the CG residuals orthogonal to the subspace spanned by $\{\mathbf{w}_1, \dots, \mathbf{w}_p\}$, provided that also $\mathbf{r}_0$ satisfies $W^T\mathbf{r}_0 = 0$. The Deflated-PCG algorithm is described below.

Algorithm 2.1 Deflated PCG

```
1:  Choose an $n \times p$ full column rank matrix $W$. Compute $\Pi = W^TAW$.
2:  Choose $\tilde{\mathbf{x}}_0$; Set $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$ and $\mathbf{x}_0 = \tilde{\mathbf{x}}_0 + W\Pi^{-1}W^T\mathbf{r}_0$.
3:  Compute $\mathbf{z}_0 = P_0\mathbf{r}_0$;
4:  Set $k = 0$, $\mathbf{p}_0 = \mathbf{z}_0 - W\Pi^{-1}W^TA\mathbf{z}_0$
5:  $\rho_0 = \mathbf{r}_0^T\mathbf{z}_0$
6:  REPEAT UNTIL CONVERGENCE
7:      $\alpha_k = \dfrac{\rho_k}{\mathbf{p}_k^TA\mathbf{p}_k}$
8:      $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k$
9:      $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_kA\mathbf{p}_k$
10:     $k = k + 1$
11:     $\mathbf{z}_k = P_0\mathbf{r}_k$
12:     $\rho_k = \mathbf{r}_k^T\mathbf{z}_k$
13:     $\beta_k = \dfrac{\rho_k}{\rho_{k-1}}$
14:     $\mathbf{p}_{k+1} = \beta_k\mathbf{p}_k + \mathbf{z}_k - W\Pi^{-1}W^TA\mathbf{z}_k$
15: END REPEAT.
```

Note that this algorithm is mathematically equivalent to the PCG method with preconditioner $P = HP_0H^T$ since

$$\mathbf{r}_k^T\mathbf{z}_k = \mathbf{r}_k^TP_0\mathbf{r}_k = \mathbf{r}_k^THP_0H^T\mathbf{r}_k,$$

being $W^T\mathbf{r}_k = 0$ and therefore $H^T\mathbf{r}_k = \mathbf{r}_k$. P is only symmetric positive semidefinite, however PCG could not breakdown as stated in the following result [7, Theorem 4.2].

Theorem 1. *Let A be an SPD matrix and W an $n \times p$ matrix with full column rank. The Deflated-PCG algorithm applied to the linear system $A\mathbf{x} = \mathbf{b}$ will not break down at any step. The approximate solution $\mathbf{x}^{(k)}$ is the unique minimizer of $\|\mathbf{x}^{(k)} - \mathbf{x}^{(*)}\|_A$ over the affine space $\mathbf{x}_0 + span(W, \mathcal{K}_k(\mathbf{r}_0))$.*

The action of this deflated preconditioner is to put some of the eigenvalues of the preconditioner matrix to zero in fact:

$$PAW = HP_0H^TAW = 0,$$

i.e. all p columns of W are eigenvectors of PA corresponding to the zero eigenvalue. Deflated CG guarantees that (at least in infinite precision arithmetic) all residuals satisfy $W^T\mathbf{r}_k = 0$. If the columns of W are (approximate) eigenvectors of P_0A corresponding to the smallest eigenvalues then these eigenvalues are removed from the spectrum of PA with obvious decrease of the condition number.

2.2. Tuning

The adjective *tuned* associated with a preconditioner has been introduced in [8] in the framework of iterative eigensolvers. In our context we redefine it in the following way:

Definition 1. *Given a preconditioner P_0 and a vector $\mathbf{w}$, a tuned preconditioner for matrix A is a matrix $P \equiv M^{-1}$ obtained by correcting P_0 by a rank one or rank two matrix depending on A and $\mathbf{w}$ and satisfying*

$$PA\mathbf{w} = \mathbf{w} \qquad (M\mathbf{w} = A\mathbf{w}). \tag{2}$$

In this way matrix M agrees with A in the direction $\mathbf{w}$ and also the preconditioned matrix has the eigenvalue 1 corresponding to the eigenvector $\mathbf{w}$. This definition can be easily extended to multiple vectors:

Definition 2. *Given a preconditioner P_0 and an $n \times p$ matrix W with full column rank, a block tuned preconditioner for matrix A is a matrix P obtained by correcting P_0 by a low rank matrix depending on A and W and satisfying*

$$PAW = W. \tag{3}$$

In [9] the following single-vector tuned preconditioned for SPD linear systems was proposed (here $M_0 = LL^T \approx A$ and $P = M^{-1}$ is computed by the Sherman-Morrison formula).

$$\mathbf{u} = (A - M_0)\mathbf{w} \qquad \mathbf{z} = P_0\mathbf{u} = P_0A\mathbf{w} - \mathbf{w} \tag{4}$$

$$M = M_0 + \frac{\mathbf{u}\mathbf{u}^T}{\mathbf{u}^T\mathbf{w}} \qquad P = P_0 - \frac{\mathbf{z}\mathbf{z}^T}{\mathbf{z}^TA\mathbf{w}} \tag{5}$$

It is easily checked that $PA\mathbf{w} = P_0A\mathbf{w} - \mathbf{z} = \mathbf{w}$.

2.2.1. Digression

This tuned preconditioner, and many others, has however an older derivation. Let us now consider the solution of a nonlinear system of equations

$$\mathbf{F}(\mathbf{x}) = 0, \qquad \mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$$

by the Newton method

$$\begin{aligned} F'(\mathbf{x}_k)\mathbf{s}_k &= -\mathbf{F}(\mathbf{x}_k) \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + \mathbf{s}_k \end{aligned}$$

Quasi-Newton methods construct a sequence of approximations of the Jacobians $B_k \approx F'(\mathbf{x}_k)$. Each B_k is defined by a low-rank update of the previous matrix in the sequence B_{k-1}. Such sequences are also used to correct a given initial preconditioner to accelerate the iterative solution of systems like (6) [10–14]

Among those Quasi-Newton methods we recall three of the most commonly used (note that $\mathbf{y}_k = \mathbf{F}(\mathbf{x}_{k+1}) - \mathbf{F}(\mathbf{x}_k)$):

Broyden's method: $\quad B_{k+1} = B_k + \dfrac{(\mathbf{y}_k - B_k\mathbf{s}_k)\mathbf{s}_k^T}{\mathbf{s}_k^T\mathbf{s}_k}$

SR1 (Symmetric Rank-1): $\quad B_{k+1} = B_k + \dfrac{(\mathbf{y}_k - B_k\mathbf{s}_k)(\mathbf{y}_k - B_k\mathbf{s}_k)^T}{(\mathbf{y}_k - B_k\mathbf{s}_k)^T\mathbf{s}_k}$

BFGS update: $\quad B_{k+1} = B_k + \dfrac{\mathbf{y}_k\mathbf{y}_k^T}{\mathbf{y}_k^T\mathbf{s}_k} - \dfrac{B_k\mathbf{s}_k\mathbf{s}_k^TB_k}{\mathbf{s}_k^TB_k\mathbf{s}_k}$

All these updates satisfy the so called *secant condition* which reads

$$B_{k+1}\mathbf{s}_k = \mathbf{y}_k. \tag{6}$$

The first update is nonsymmetric, the SR1 and BFGS formulae define a symmetric sequence provided B_0 is so. Consider now the problem of updating a given preconditioner $P_0 = M_0^{-1}$. We can use all the preceding equations after employing the following substitutions:

$$\mathbf{s}_k \longrightarrow \mathbf{w}, \qquad \mathbf{y}_k \longrightarrow A\mathbf{w}, \qquad B_k \longrightarrow M_0, \qquad B_{k+1} \longrightarrow M,$$

which transform the secant condition into

$$M\mathbf{w} = A\mathbf{w}, \tag{7}$$

that is the *tuning property*. We can then develop three different tuning strategies. For each of them we write the "direct" formula (with M, M_0) the inverse formulation (with P, P_0), by the Shermann-Morrison inversion formula, and develop the corresponding inverse block formula.

Table 1. Direct, inverse and block versions of the Broyden (nonsymmetric) update.

direct	$M = M_0 + \dfrac{(A - M_0)\mathbf{w}\mathbf{w}^T}{\mathbf{w}^T\mathbf{w}}$	$M\mathbf{w} = A\mathbf{w}$
inverse	$P = P_0 - \dfrac{(P_0A\mathbf{w} - \mathbf{w})\mathbf{w}^TP_0}{\mathbf{w}^TP_0A\mathbf{w}}$	$PA\mathbf{w} = \mathbf{w}$
block	$P = P_0 - (P_0AW - W)\left(W^TP_0AW\right)^{-1}W^TP_0$	$PAW = W$

Table 2. Direct, inverse and block versions of the SR1 update.

direct	$M = M_0 + \dfrac{\mathbf{u}\mathbf{u}^T}{\mathbf{w}^T\mathbf{u}},$	$\mathbf{u} = (A - M_0)\mathbf{w}$
inverse	$P = P_0 - \dfrac{\mathbf{z}\mathbf{z}^T}{\mathbf{z}^TA\mathbf{w}},$	$\mathbf{z} = P_0A\mathbf{w} - \mathbf{w}$
block	$P = P_0 - Z\left(Z^TAW\right)^{-1}Z^T,$	$Z = P_0AW - W$

Table 3. Direct, inverse and block versions of the BFGS update.

direct	$M = M_0 + \dfrac{(A\mathbf{w})^TA\mathbf{w}}{\mathbf{w}^TA\mathbf{w}} - \dfrac{M_0\mathbf{w}\mathbf{w}^TM_0}{\mathbf{w}^TM_0\mathbf{w}}$
inverse	$P = \dfrac{\mathbf{w}\mathbf{w}^T}{\mathbf{w}^TA\mathbf{w}} + \left(I - \dfrac{\mathbf{w}\mathbf{w}^TA}{\mathbf{w}^TA\mathbf{w}}\right)P_0\left(I - \dfrac{A\mathbf{w}\mathbf{w}^T}{\mathbf{w}^TA\mathbf{w}}\right)$
block	$P = W\Pi^{-1}W^T + HP_0H^T \qquad \Pi = W^TAW$ (1) $H = I - W\Pi^{-1}W^TA$

The Broyden tuning strategy must be used for nonsymmetric problems, the BFGS formula is well suited to accelerate the PCG method due to the following result:

Theorem 2. *The preconditioner P yielded by the BFGS update formula is SPD provided P_0 is so.*

Proof. For every nonzero $\mathbf{x} \in \mathbb{R}^n$ we set $\mathbf{z} = H^T\mathbf{x}$ and $\mathbf{u} = W^T\mathbf{x}$. Then we have

$$\mathbf{x}^TP\mathbf{x} = (W^T\mathbf{x})^T\Pi^{-1}(W^T\mathbf{x}) + \mathbf{x}^THP_0H^T\mathbf{x} = \mathbf{u}^T\Pi^{-1}\mathbf{u} + \mathbf{z}^TP_0\mathbf{z} \geq 0. \tag{8}$$

the last inequality holding since both Π^{-1} and P_0 are SPD matrices. The inequality is strict since if $\mathbf{u} = 0$ then $W^T\mathbf{x} = 0$ and hence $\mathbf{z} = (I - AW\Pi^{-1}W^T)\mathbf{x} = \mathbf{x} \neq 0$. □

Finally, the SR1 update provides clearly symmetric matrices, upon symmetry of P_0. If in addition P_0 is SPD the new update P is also likely to be SPD as well, depending on the quality of the initial preconditioner P_0, as stated in the following result [15, Sec. 3.2].

$$\lambda_{\min}(P) \geq \lambda_{\min}(P_0) - \|P_0A - I\|^2\|(W^TAW)^{-1}\|, \tag{9}$$

which has however a modest practical use as $\|(W^TAW)^{-1}\|$ may be large.

Whenever the columns of W approximate the smallest eigenvalues of P_0A something more can be said. Assume that $P_0AW = W\Theta$, with $\Theta = \operatorname{diag}(\mu_1, \dots, \mu_p)$ the diagonal matrix with the eigenvalues of the initially preconditioned matrix. Assume further that $\mu_j < 1, j = 1, \dots, p$. Since P_0A is not symmetric but $P_0^{1/2}AP_0^{1/2}$ is so it follows that W must satisfy $W^TP_0^{-1}W = I$ and hence $W^TAW = \Theta$ (see the discussion on the forthcoming Section 5.1). In such a case we have that matrix

$$-Z^TAW = (W - P_0AW)^T AW = (I - \Theta)W^TAW = (I - \Theta)\Theta$$

is obviously SPD and therefore so is P which is the sum of an SPD matrix (P_0) and a symmetric positive semidefinite matrix ($-Z(Z^TAW)^{-1}Z^T$).

The Broyden tuned preconditioner has been used in [16] to accelerate the GMRES for the inner linear systems within the Arnoldi method for eigenvalue computation. The SR1 preconditioner appeared in the cited works [9,15] and also in [17]. This low-rank update has also been employed to accelerate the PCG method in the solution of linear systems involving SPD Jacobian matrices. In [18] some conditions are proved under which the SR1 update maintains the symmetric positive definiteness of a given initial preconditioner. The BFGS preconditioner has been used (under the name of *balancing preconditioner*) in [19], in [20] for eigenvalue computation and also in [21] to update the Constraint Preconditioner for sequences of KKT linear systems. It has been successfully employed (under the name of limited memory preconditioner) to accelerate sequences of symmetric indefinite linear systems in [22].

We finally mention the work in [23] where the author use this correction to update an Inexact Constraint Preconditioner, by damping the largest eigenvalues of the preconditioned matrices.

2.3. Spectral preconditioners

A further strategy to improve the preconditioner's efficiency by a low-rank update can be found in [17,24–26]. A spectral preconditioner for SPD linear systems is defined as

$$P = P_0 + W(W^TAW)^{-1}W^T,$$

with the leftmost eigenvectors of P_0A as the columns of W. Denoted as $\mu_1, \mu_2, \dots, \mu_p$ the smallest eigenvalues of P_0A it is easily checked that matrix PA has the same eigenvectors $\mathbf{w}_1, \dots, \mathbf{w}_p$ as P_0A with $\mu_1 + 1, \dots, \mu_p + 1$ as eigenvalues.

3. Implementation and computational complexity

We sketch below the most time-consuming computational tasks when using a low-rank update of a given preconditioner. We first evaluate the preliminary cost, which has to be done before starting of the iterative process:

1. All low-rank updates: Computation of $A_W = AW \quad (O(np)$ operations).
2. Broyden and SR1 only: Computation of $Z = P_0A_W - W$ (p applications of the initial preconditioner).
3. All low-rank updates: Computation of $\Pi = W^TA_W \quad (O(p^2n)$ operations).

The cost of this tasks is likely to be amortized during the PCG iterations of the various linear systems to be solved.

Regarding the most important cost per iteration, the deflated, Broyden, SR1 and spectral preconditioners behave the same, being based on the following main tasks, in addition to the application of the initial preconditioner P_0 to a given vector.

1. Multiplication of an $n \times p$ matrix times a vector ($O(np)$ operations)

2. Solution of a small linear system with matrix Π. ($O(p^3)$ operations).
3. Multiplication of a $p \times n$ matrix times a vector ($O(np)$ operations)

The BFGS preconditioner, as it corrects P_0 by a rank-$(2p)$ matrix, roughly doubles tasks 1–3 and hence the total updating cost at each iteration.

4. Choice of the vectors $\{\mathbf{w}_j\}$.

In principle every set of linearly independent vectors can be selected as columns of W. However, the optimal choice is represented by the eigenvectors of the **preconditioned matrix** P_0A corresponding to the smallest eigenvalues. Approximation of these vectors as a by-product of the Conjugate Gradient method has been considered in various works. We will turn on this crucial issue later on.

We start by presenting some numerical results obtained computing the "true" 10 leftmost eigenvectors of P_0A using the MatLAB `eigs` command. To measure the sensitivity of the method to eigenvector accuracy we also use perturbed eigenvectors i.e. satisfying $\|P_0A\mathbf{w}_j - \mu_j\mathbf{w}_j\| \approx \delta$. The coefficient matrix has been chosen as the FD discretization of the Laplacian on an L-shaped domain obtained using the MatLAB command

```
A = delsq(numgrid('L',500));
```

which returns a sparse matrix of order $n = 186003$. The linear system $A\mathbf{x} = \mathbf{b}$, with $\mathbf{b}$ a random uniformly distributed vector, has been solved by the PCG method with various low-rank update techniques with $P_0 = IC(0)$.

The results in Table 4 show that a very low accuracy on the eigenvectors (absolute residual of order of $\delta = 0.01$) is sufficient to provide good preconditioning results. Also notice that the tuned preconditioner seems to be more sensitive to the parameter δ.

Table 4. Influence of accuracy of the eigenvectors of P_0A on the efficiency of the low-rank preconditioners. PCG iterations with various updating strategies are reported.

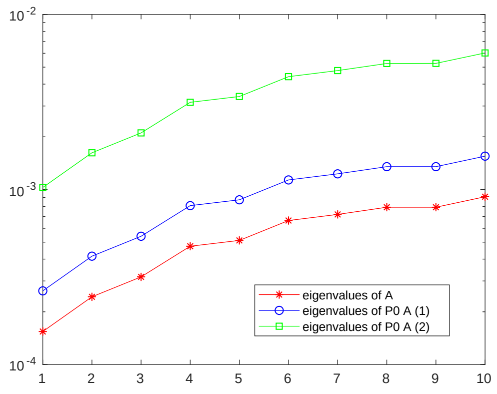
	no update	tuned	deflated	spectral
exact	466	254	254	254
$\delta = 0.01$	466	261	259	290
$\delta = 0.05$	466	378	260	286

It may also happen that some of the leftmost eigenpairs of the **coefficient matrix** are instead available. What if one uses these vectors as columns of W? The following Table 5 gives a surprising answer: if they are computed to a good accuracy they provide a notable acceleration of convergence. Again we use either exact eigenvectors or vectors satisfying $\|A\mathbf{w}_j - \lambda_j\mathbf{w}_j\| \approx \delta$.

Table 5. Influence of accuracy of the eigenvectors of A on the efficiency of the low-rank preconditioners. PCG iterations with various updating strategies are reported.

	no update	tuned	deflated	spectral
exact	466	254	254	254
$\delta = 10^{-3}$	466	296	296	297
$\delta = 0.01$	466	362	361	369

The conclusion is: yes we can use the leftmost eigenvectors of A instead of those of P_0A without dramatically loosing efficiency. The reason for this behavior is connected to the action of preconditioners such as Incomplete Cholesky or approximate inverses. They usually leave almost unchanged the eigenvectors corresponding to the smallest eigenvalues (though increasing the latter ones).



j	$\angle\left(\mathbf{v}_j, \mathbf{v}_j^{P1}\right)$	$\angle\left(\mathbf{v}_j, \mathbf{v}_j^{P2}\right)$
1	1.7445e-05	2.6952e-04
2	3.6118e-05	4.3401e-04
3	9.6668e-05	1.5363e-04
4	1.9743e-04	4.5222e-03
5	1.7449e-04	4.9131e-03

Figure 1. Eigenvalues and angle between eigenvectors of A and IC-preconditioned A.

Consider again the previous matrix and compare the eigenpairs $(\lambda_j, \mathbf{v}_j)$ of A, with those of the preconditioned matrix with two choices of P_0: $IC(0)$ $(\mu_j^{P1}, \mathbf{v}_j^{P1})$ and the IC factorization with `droptol` $= 1e-2$ $(\mu_j^{P2}, \mathbf{v}_j^{P2})$. As expected, the preconditioners shift bottom-up the smallest eigenvalues. However the angle between the corresponding eigenvectors is close to zero showing that they are almost parallel.

4.1. Using previous solution vectors

Computing a subset of the extremal eigenvectors of a given large and sparse matrix may reveal computationally costly. In the next Section we will develop some methods to save on this cost. However another possibility for the vectors $\mathbf{w}_j$ is to use some of the solutions to the previous linear systems in the sequence. Assume now that the matrices in the sequence are allowed to (slightly) change, i.e. we want to solve

$$A_k\mathbf{x}_k = \mathbf{b}_k, \qquad , k = 1, \ldots .$$

In this case computing leftmost eigenvectors of A_k is inefficient since this preprocessing should be done at each linear system.

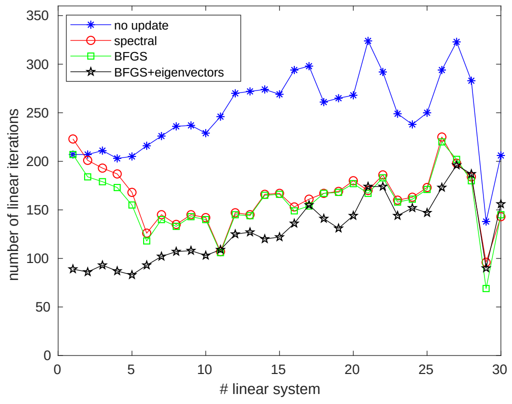
We then use, for the jth linear system $(j > 1)$, the $j_{eff} = \min\{j-1, p\}$ solutions to the previous linear systems as columns of $W \equiv W_j$, thus yielding:

$$W_j = \begin{bmatrix} \mathbf{x}_{j-j_{eff}} & \ldots & \mathbf{x}_{j-1} \end{bmatrix} \tag{10}$$

The idea beyond this choice, which is completely cost-free, comes from the reasoning that a solution $\mathbf{x}_k$ of the k-th linear system has with high probability non-negligible components in the directions of the leftmost eigenvectors of A_k since, if $\mathbf{b}_k = \sum_{i=1}^{n} \alpha_i \mathbf{u}_i$, then $\mathbf{x}_k = \sum_{i=1}^{n} \frac{\alpha_i}{\lambda_i}\mathbf{u}_i$, with the largest weights provided by the smallest eigenvalues.
To give experimental evidence of the efficiency of this approach we report some results in solving a sequence of 30 linear systems arising from the Finite Element/Backward Euler discretization of the branched transport equation [26] which gives raise to a sequence of (2×2) block nonlinear systems in turn solved by the Inexact Newton method. After linearization and block Gaussian Elimination the problem reduces to the solution of a sequences of linear systems of the form

$$(A + BEB^T)_k\mathbf{x} = \mathbf{b}_k, \quad k = 1, \ldots ,$$

Figure 2. Number of iterations to solve each linear system in the sequence by MINRES with: no update, BFGS update with previous solutions, spectral update with previous solutions, BFGS update with leftmost eigenvectors. On the right also the CPU time is reported.



strategy	iter.	CPU (s)
no update	7491	51.53
BFGS + previous solutions	4769	50.14
Spectral + previous solutions	4925	41.45
BFGS + eigenvectors	3854	145.07

where A is SPD, B is rectangular with full row rank, D is diagonal and possibly indefinite. We take 30 consecutive linear systems from this sequence, and solve them with the MINRES iterative method using as the initial preconditioner the incomplete Cholesky factorization of A with drop tolerance 10^{-2} and the following updates

1. BFGS, with update directions as in (10).
2. Spectral preconditioner, with update direction as in (10).
3. BFGS, with as update directions the accurate leftmost eigenvectors of A_k.

The results are reported in Figure 2 where we can appreciate the number of iterations reduction provided by the low-rank update. The (cheaper) spectral preconditioner provides as good acceleration as the BFGS approach with update directions as in (10). Moreover the number of iterations is comparable to those obtained with the "exact" eigenvectors (which are obtained by the Matlab `eigs` function).

5. Cost-free approximation of the leftmost eigenpairs

The most appropriate candidate vectors for the columns of W are anyway the eigenvectors of the preconditioned matrix corresponding to the smallest eigenvalues.

5.1. The Lanczos-PCG connection

A simple way to recover some of the extremal eigenpairs of the preconditioned matrix is to exploit the so called *Lanczos connection* [27,28]. During the PCG method, preconditioned with P_0, it is possible to save the first m (scaled) preconditioned residuals as columns of a matrix V_m:

$$V_m = \left[\frac{P_0 \mathbf{r}_0}{\sqrt{\mathbf{r}_0^T P_0 \mathbf{r}_0}}, \frac{P_0 \mathbf{r}_1}{\sqrt{\mathbf{r}_1^T P_0 \mathbf{r}_1}}, \ldots, \frac{P_0 \mathbf{r}_{m-1}}{\sqrt{\mathbf{r}_{m-1}^T P_0 \mathbf{r}_{m-1}}} \right] = \left[\frac{\mathbf{z}_0}{\sqrt{\rho_0}}, \frac{\mathbf{z}_1}{\sqrt{\rho_1}}, \ldots, \frac{\mathbf{z}_{m-1}}{\sqrt{\rho_{m-1}.}} \right].$$

Matrix V_m is such that $V_m^T P_0^{-1} V_m = I_m$, in view of the P_0-orthogonality of the residuals generated by the PCG method. The Lanczos tridiagonal matrix can be formed using the PCG coefficients α_k, β_k:

$$T_m = \begin{bmatrix} \frac{1}{\alpha_0} & -\frac{\sqrt{\beta_1}}{\alpha_0} & & & \\ -\frac{\sqrt{\beta_1}}{\alpha_0} & \frac{1}{\alpha_1} + \frac{\beta_1}{\alpha_0} & -\frac{\sqrt{\beta_2}}{\alpha_1} & & \\ & & \ddots & & \\ & & & & -\frac{\sqrt{\beta_{m-1}}}{\alpha_{m-2}} \\ & & & -\frac{\sqrt{\beta_{m-1}}}{\alpha_{m-2}} & \frac{1}{\alpha_{m-1}} + \frac{\beta_{m-1}}{\alpha_{m-2}} \end{bmatrix}$$

Matrices V_m and T_m obey to the classical Lanczos relation i.e.:

$$V_m^T A V_m = T_m.$$

After eigensolving T_m, thus obtaining $T_m = Q\Lambda_m Q^T$, the eigenvectors corresponding to the p smallest eigenvalues are selected as columns of matrix Q_p. Finally, the columns of $W_p = V_m Q_p$ are expected to approximate p of the leftmost eigenvectors of $P_0 A$.

This procedure can be implemented to a very little computational cost but it has a number of disadvantages: First, it requires the storage of m preconditioned residuals, Second, as the convergence for the Lanczos process to the smallest eigenvalues is relatively slow, it sometimes happens that PCG convergence takes place before eigenvector convergence. Third, some of the leftmost eigenpairs can be missing by the non-restarted Lanczos procedure.

To partially overcome these drawbacks in [29] the authors suggest to implement a thick restart within the PCG iteration (without affecting it) and incrementally correct the eigenvector approximations during subsequent linear system solvers. The final set of directions is, in that paper, used only to deflate the initial vector and not to form a preconditioner (`eigCG` algorithm) in the solution of a sequence of hundreds of linear systems in lattice quantum chromodynamics.

A number of alternative ways to approximate eigenvectors during the PCG iteration have been can be found in the literature. In [7] a technique is suggested to refine them during subsequent linear systems with the same matrix but different right-hand-sides by using the p smallest harmonic Ritz values. In this approach the vector stored within the PCG iteration are the conjugate directions $\mathbf{p}_k$ instead of the preconditioned residuals (they actually lie in the same Krylov subspace).

6. Numerical results

The technique previously describe are very powerful when applied to sequences of linear systems where the coefficient matrix remains the same. In this case it is possible to refine the eigenvalues of the preconditioned matrix during the first linear systems solutions up to machine precision [29]. However, in most of the situations this high accuracy is not really needed (see the discussion in Section 6.5).

In this section we will consider a particular case of sequences of linear systems with (slightly) changing matrices as

$$A_k \mathbf{x} = \mathbf{b}_k, \qquad A_k = A_0 + \alpha_k A_1,$$

which will be denoted as *shifted linear systems.* These sequences appear e.g. in the Finite Difference or Finite Element discretization of transient PDEs or as an inner linear system within iterative eigensolvers. We will show that the low-rank techniques could be successfully employed in the acceleration of iterative methods in the solution of such linear systems.

6.1. FE Discretization of a parabolic equation

Consider the parabolic PDE

$$S_h \frac{\partial u(\vec{x},t)}{\partial t} \quad = \quad \nabla \left(K(\vec{x}) \nabla u(\vec{x},t)\right) \qquad + \text{ BCs}$$

u is the pressure of the fluid, S_h the elastic storage, K the (SPD) hydraulic conductivity tensor. This equation is used to model water pumping in the Beijing plan to predict land subsidence in that area [30]. The 3D computational domain turns out to be very heterogeneous and geometrically irregular as sketched in Figure 3:

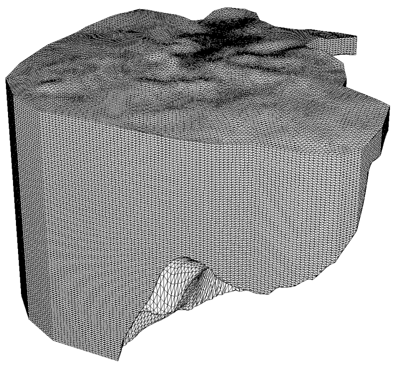


Figure 3. 3D domain and triangulation.

FE discretization in space needs highly irregular tetrahedra, which, combined with the high heterogeneity of the medium, gives raise to an ill-conditioned discretized diffusion matrix. After FE discretization a system of ODEs is left:

$$B \frac{\partial \mathbf{u}(t)}{\partial t} \quad = \quad -A\mathbf{u}(t) + \mathbf{b}(t).$$

where A is the SPD stiffness matrix, B is the SPD mass matrix. The right-hand-side account for source terms and Neumann boundary conditions. Using a time marching scheme (e.g. Crank-Nicolson for stiff problems) yields a sequence of linear systems

$$(A + \sigma_k B)\mathbf{u}_k = \mathbf{b}_k, \quad \sigma_k = \frac{2}{\Delta t_k}, \qquad k = 1, Nstep. \tag{11}$$

The discretized FE matrices have both $n = 589661$ and a nonzero number $nz = 8833795$. We now report the results of a complete simulation which took $Nstep = 32$ steps. The near steady state is reached after $T = 36000$ days. Initially we set $\Delta t_1 = 60$ (days). Then we incremented the timesteps as

$$\Delta t_k = \min(1.2\Delta t_{k-1}, 7300, T - \Delta t_{k-1}) \Longrightarrow \sigma_k = \frac{2}{\Delta t_k}, \qquad k = 2, \ldots, Nstep$$

The initial preconditioner P_0 has been evaluated employing the MatLAB function `ichol` with

(1) `droptol` $= 10^{-5}$ applied to $A_1 = A + 5B$ to enhance diagonal dominance; this resulted in a triangular Cholesky factor with density 5.18 and
(2) `droptol` $= 10^{-4}$ applied to $A_2 = A + 20B$ (density 2.83).

We solved the first linear system to a high accuracy to assess as accurately as possible the $p = 10$ leftmost eigenvectors. Using the residual $\delta_j = \dfrac{\|P_0 A \mathbf{w}_j - \mu_j \mathbf{w}_j\|}{\|\mathbf{w}_j\|}$ to measure the accuracy of the approximate eigenvectors we found:

- Case (1). 179 PCG iterations for the first linear system; $\delta_j \in [1.1\,10^{-3}, 5.8\,10^{-3}]$.
- Case (2). 370 PCG iterations for the first linear system; $\delta_j \in [1.1\,10^{-3}, 3.3\,10^{-3}]$.

This set of vectors is used them to construct deflation, tuned and spectral preconditioners for all the subsequent linear systems in the sequence (with exit test on relative residual and tolerance TOL$= 1e - 10$).

Table 6. Number of iterations and total CPU time for solving the sequence (11) of shifted linear system. $\dagger =$ for some of the system in the sequence convergence not achieved.

	IC$(A_1, 1e-5)$			IC$(A_2, 1e-4)$		
update	# its	CPU	CPU per it.	# its	CPU	CPU per it.
no update	8231	805.8	0.098	16582	1177.2	0.071
spectral	5213	557.3	0.107	10225	817.5	0.080
SR1 tuned	5161	551.7	0.107	10152	811.4	0.080
BFGS tuned	5178	612.3	0.118	10198	924.6	0.091
deflated	†	†	†	†	†	†

The results are summarized in Table 6. The eigenvectors estimated during the first linear system are adequate to accelerate the subsequent iterations for both the tuned and spectral approaches which provide an improvement in both iteration number and CPU time. Note also that the time per iteration is only slightly increased with respect to the "no update" case.

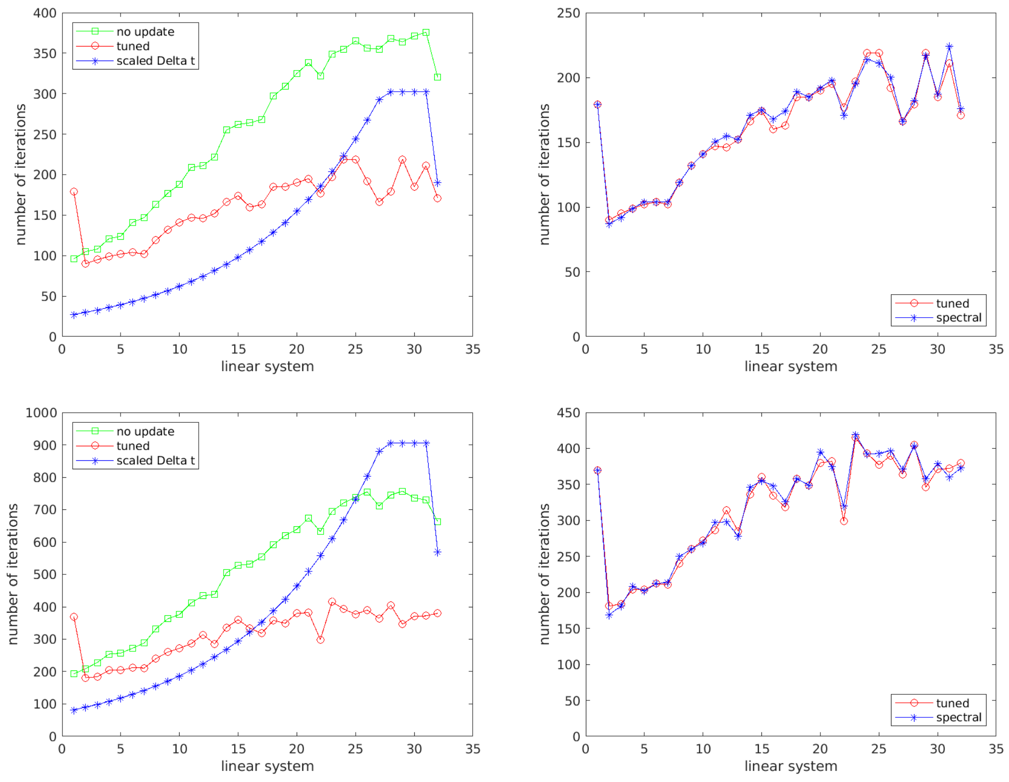


Figure 4. Number of iterations for each linear system in the sequence and various preconditioning strategies. Initial preconditioner: IC$(A_1, 1e-5)$ (upper plots) and IC$(A_2, 1e-4)$ lower plots.

Instead, the PCG residuals obtained with the deflation preconditioner in some instances stagnate before reaching the exit test. The lack of robustness of the deflation approach is also observed in [7] where the authors suggest a reorthogonalization of the residuals at the price of increasing costs per iteration and also in [16] in the framework of iterative eigensolvers. Regarding tuned preconditioners, SR1 is definitely superior to the BFGS approach due do its lower computational cost.

In Figure 4, left, we plot the number of iterations needed by the "no update" and tuned preconditioners for every system in the sequence compared also with the scaled $\sqrt{\Delta t_k} \approx \sigma_k^{-1/2}$ values as indicators of the increasing condition number of the coefficient matrices. On the right plot the behavior of the tuned vs spectral preconditioners is displayed, showing the substantial equivalence between the two approaches.

6.2. Iterative eigensolvers

When computing the smallest or a number of the interior eigenvalues of a large and sparse matrix, most iterative solvers require the solution of a shifted linear system of the type

$$(A - \sigma B)\mathbf{x} = \mathbf{b}.$$

This is true for instance in the shift-invert Arnoldi method, in the inverse power iteration, in the Rayleigh Quotient iteration. A related linear system is also to be solved within the *correction equation* in the Jacobi-Davidson method. Other gradient-based methods such as LOBPCG [31] or DACG [32] implicitly solve a shifted linear system. The sequence of linear systems are characterized by a constant or a slowly varying parameter σ so informations on matrix A or on the pencil (A, B) can be profitably used for all the sequence.

We denote as $0 < \lambda_1 \le \lambda_2 \le \dots \lambda_m \dots \le \lambda_n$ its eigenvalues and $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m, \dots \mathbf{v}_n$ the corresponding eigenvectors. Our aim is to investigate the efficiency of the SR1 tuned preconditioner in the acceleration of the so-called correction equation in the simplified Jacobi-Davidson (JD) method [33, 34]. To assess eigenpair $(\lambda_j, \mathbf{v}_j)$ a number of linear systems have to be solved of the form

$$\begin{aligned} J_k^{(j)}\mathbf{s} &= -(A - \theta_j^{(k)} I)\mathbf{u}_k; \quad \text{for} \quad \mathbf{s} \perp \mathbf{u}_k, \qquad \text{where} \qquad (12)\\ J_k^{(j)} &= (I - QQ^T)(A - \theta_j^{(k)} I)(I - QQ^T), \; \theta_j^{(k)} = q(\mathbf{u}_k) \equiv \frac{\mathbf{u}_k^T A \mathbf{u}_k}{\mathbf{u}_k^T \mathbf{u}_k},\\ Q &= \begin{bmatrix} \mathbf{v}_1 & \dots \mathbf{v}_{j-1} & \mathbf{u}_k \end{bmatrix} \end{aligned}$$

The PCG method is proved to converge if applied to systems (12) as the Jacobian $J_k^{(j)}$ is shown to be SPD in the subspace orthogonal to $\mathbf{u}_k$ [12,34]. Following the works in [17,35], which are based on a two-stage algorithm, the columns of W are orthonormal *approximate* eigenvectors of A provided by a gradient method, DACG [32], satisfying

$$\begin{aligned} A\tilde{\mathbf{v}}_s = \tilde{\lambda}_s \tilde{\mathbf{v}}_s + \mathbf{res}_s, \qquad \|\mathbf{res}_s\| &\le \tau\lambda_s \qquad (13)\\ & \qquad\qquad s = 1, \dots, m.\\ \tilde{\lambda}_s - \lambda_s &\le \tau\lambda_s \qquad (14) \end{aligned}$$

These vectors are also used as initial guesses for the Newton phase. To update a given initial preconditioner for $(\lambda_j, \mathbf{v}_j)$ we use the next approximate eigenvectors $\mathbf{v}_{j+1}, \dots, \mathbf{v}_m$ to define an SR1 tuned preconditioner as

$$P_j = P_0 - Z\left(Z^T A W_j\right)^{-1} Z^T, \quad Z = P_0 A W_j - W_j, \text{ with } W_j = \begin{bmatrix} \tilde{\mathbf{v}}_{j+1} & \dots & \tilde{\mathbf{v}}_m \end{bmatrix} \qquad (15)$$

We recall the following result stated in [17, Lemma 3.1]:

Lemma 1. *Let P_j a block tuned preconditioner satisfying condition (15). In the hypothesis (13) each column of W_j i.e. $\tilde{\mathbf{v}}_s, s = j+1, \dots, m$, is an approximate eigenvector of $P_j(A - \theta_j I)$ satisfying*

$$P_j(A - \theta_j I)\tilde{\mathbf{v}}_s = \left(1 - \frac{\theta_j}{\tilde{\lambda}_s}\right)\tilde{\mathbf{v}}_s + \boldsymbol{\varepsilon}_s, \qquad \textit{with} \qquad \|\boldsymbol{\varepsilon}_s\| \le \tau\lambda_{j+1}\|P_j\|.$$

The effect of applying a tuned preconditioner to $A - \theta_j I$ is to set a number of eigenvalues of $P(A - \theta_j I)$ to a value that is close to one only under the conditions that the eigenvalues are well separated, i.e. $\dfrac{\lambda_j}{\lambda_{j+1}} \ll 1$, which is not always the case in realistic problems.

In order to define a more effective preconditioner for shifted linear systems one can allow the preconditioned matrix PA to have eigenvalues different from one corresponding to the columns of matrix W. In [35] a **generalized block tuned** (GBT) preconditioner is defined:

Definition 3. *Given a preconditioner P_0, an $n \times p$ matrix W with full column rank, and a diagonal matrix $\Gamma = diag(\gamma_1, \dots, \gamma_p)$, a generalized block tuned preconditioner for matrix A is a matrix P obtained by correcting P_0 with a low-rank matrix depending on A, W and Γ and satisfying*

$$PAW = W\Gamma. \tag{16}$$

The generalized SR1 preconditioner is defined as

$$P = P_0 - Z\Pi^{-1}Z^T, \quad \text{where} \quad Z = P_0AW - W\Gamma, \quad \text{and} \quad \Pi = Z^TAW. \tag{17}$$

Note that the above preconditioner is not in general symmetric as small matrix Π is not and hence its use would prevent convergence either of the DACG eigensolver or the inner PCG iteration within the Newton method. However, this drawback can be circumvented when $W \equiv W_j$ represents the matrix of the (approximate) eigenvectors corresponding to the diagonal matrix with the eigenvalues of A: $\Lambda_j = \text{diag}(\tilde{\lambda}_{j+1}, \tilde{\lambda}_{j+2}, \dots, \tilde{\lambda}_m)$. In such case we can approximate Π as

$$\Pi = W_j^TAPAW_j - \Gamma W_j^TAW_j \approx W_j^TAPAW_j - \Gamma\Lambda_j = \tilde{\Pi}, \tag{18}$$

so restoring symmetry. This modified preconditioner $\widetilde{P}_j = P_0 - Z\tilde{\Pi}^{-1}Z^T$ satisfies only approximately the tuning property as

$$\widetilde{P}_jA\tilde{\mathbf{v}}_s = \gamma_s\tilde{\mathbf{v}}_s + \mathcal{E}_s, \quad \|\mathcal{E}_s\| \le \tau\lambda_s\|Z\tilde{\Pi}^{-1}\Gamma\|, \qquad s = j+1, \dots, m. \tag{19}$$

The following theorem states that it is however possible to have p eigenvalues of the preconditioned matrix $\widetilde{P}_j(A - \theta_j I)$ very close to one depending on how the columns of matrix W approximate the eigenvectors of A. We also define the reciprocal of the relative separation between pairs of eigenvalues as

$$\xi_j = \frac{\lambda_{j+1}}{\lambda_{j+1} - \lambda_j}. \tag{20}$$

Theorem 3. *Let matrix W_j be as in (15), $\widetilde{P}_j$ an approximate GBT preconditioner satisfying condition (19), with $\gamma_s = \tilde{\lambda}_s/(\tilde{\lambda}_s - \tilde{\lambda}_j)$, $s = j+1, \dots, m$, then each column of W_j is an approximate eigenvector of $\widetilde{P}_j(A - \theta_j I)$ corresponding to the approximate eigenvalue*

$$\mu_s = \frac{\tilde{\lambda}_s - \theta_j}{\tilde{\lambda}_s - \tilde{\lambda}_j}.$$

satisfying

$$\widetilde{P}_j(A-\theta_j I)\tilde{\mathbf{v}}_s = \mu_s\tilde{\mathbf{v}}_s + \boldsymbol{\varepsilon}_s, \qquad \textit{with} \qquad \|\boldsymbol{\varepsilon}_s\| \le \tau\left(\lambda_s\|Z\widetilde{\Pi}^{-1}\Gamma\| + \lambda_{j+1}\|\widetilde{P}_j\|\right).$$

Proof. See [35]. □

The eigenvalues μ_s can be very close to one depending on the initial tolerance τ as stated in Corollary 1, under reasonable additional hypotheses

Corollary 1. *Let θ_j, τ such that, for all $j = 1, m$:*

$$\begin{aligned} \lambda_j &\le \theta_j \le \tilde{\lambda}_j, & (21)\\ \tau &< (2\xi_j)^{-1}, & (22) \end{aligned}$$

then

$$1 \le \mu_s \le 1 + 2\tau(\xi_j - 1), \qquad s = j+1, \ldots, m.$$

Proof. See [35]. □

From Corollary 1 it is clear that μ_s can be made arbitrarily close to one by appropriately reducing the tolerance τ. As an example, if $\xi_j = 10^2$, $\tau = 10^{-3}$ then all μ_s are expected to be in $(1, 1.2)$.

In [35, Theorem 2] is also stated that the eigenvalues of the preconditioned projected Jacobian matrix (12) are characterized in the same way as stated in Theorem 3, i.e. that for a suitable function $C(\tau)$, increasing in τ

$$P_Q J_k^{(j)}\tilde{\mathbf{v}}_s = \mu_s\tilde{\mathbf{v}}_s + \mathbf{err}, \qquad \|\mathbf{err}\| \le \tau C, \tag{23}$$

where $\widetilde{P}_j$ a generalized block tuned preconditioner, $P_Q = (I - QQ^T)\tilde{P}_j(I - QQ^T)$.

To limit the cost of the application of the low-rank correction it is customary to fix the maximum number of columns of matrix W_j, parameter $l_{\max}$. Conversely, it is required to enlarge it when assessing eigenpairs with $j \approx m$. The first DACG stage is hence used to compute an extra number (win) of approximated eigenpairs. In this way the number of columns of W_j is $m_j = \min\{l_{\max}, m + \text{win} - j\}$.

The construction (C) of $\widetilde{P}_j$ and its application (A) as $\widetilde{P}_j\mathbf{r}$ are sketched below. MVP = matrix-vector products, $Z_j = Z_0$(:,j+1,j+m_j) and $\Pi_j = \Pi_0$(j+1:j+m_j, j+1:j+m_j).

phase	when	what	relevant cost
C	once and for all	• $Z_0 = P_0 A V_0$	m + win MVPs and applications of P_0
		• $\Pi_0 = Z_0^T A V_0$	$(m + \text{win})^2/2$ dot products.
C	for every eigenpair	• $Z = Z_j - V_j\Gamma$	m_j daxpys
		• $\tilde{\Pi} = \Pi_j - \Gamma\Lambda_j$	
A	at each iteration	• $\mathbf{h} = Z^T\mathbf{r}$	m_j dot products
		• $\mathbf{g} = \tilde{\Pi}\backslash\mathbf{h}$	1 system solve of size m_j
		• $\mathbf{w} = P_0\mathbf{r} - Z\mathbf{g}$	1 application of P_0, m_j daxpys

We report from [35] the results of the described preconditioner in the computation of the 20 smallest eigenpairs of matrix THERMOMEC, which is available in the SuiteSparse Matrix Collection at `https://sparse.tamu.edu/`. This is a challenging test due to the high clustering of its smallest eigenvalues. 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. The exit test is on the relative eigenresidual: $\dfrac{\|A\mathbf{u} - q(\mathbf{u})\mathbf{u}\|}{q(\mathbf{u})} \le \varepsilon = 10^{-8}$.

The results of the runs are displayed in Table 7 where the number of iterations (and CPU time) of the initial DACG method are reported as well as the cumulative number of iterations (and CPU time) needed for solving all the linear systems (12) within the Newton iteration for the 20 eigenpairs.

Especially the second (Newton) phase takes great advantage by the GBT preconditioner as also accounted for by Figure 5 which shows the number of iterations taken by the Newton phase with the fixed IC, SR1 tuned and GBT preconditioners. The almost constant GBT curve confirms the property of this preconditioner which makes the number of iterations nearly independent on the relative separation between consecutive eigenvalues.

Table 7. Timings and iterations for the DACG-Newton method for the computation of $m = 20$ eigenpairs of matrix THERMOMEC.

			DACG			Newton			Total	
						Iterations		CPU		
Prec.	win	$l_{\max}$	τ	Its.	CPU	OUT	Inner		MVP	CPU
Fixed	0	0	10^{-4}	1510	15.86	153	2628	34.12	4291	52.97
Spectral	10	10	10^{-3}	1335	14.89	137	2187	32.19	3659	51.48
GBT	5	10	10^{-3}	777	11.16	44	607	9.42	**1428**	**20.74**

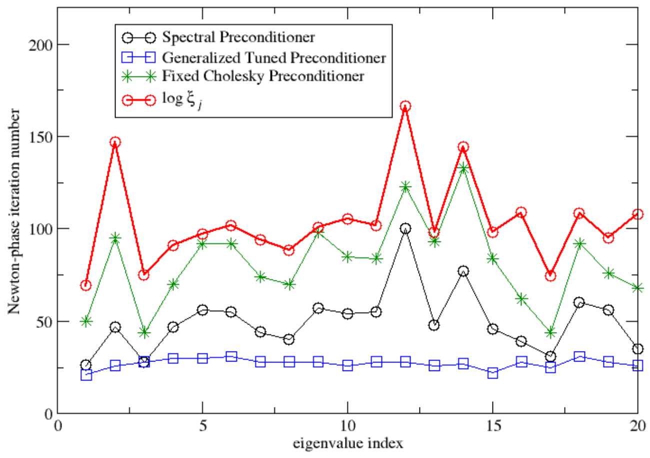


Figure 5. Number of iterations for the Newton phase with fixed, SR1 tuned and GBT preconditioners. In red the (scaled) logarithm of the indicator ξ_j.

7. Conclusions

We have presented a general framework of updating a given preconditioner P_0 with a low-rank matrix with the aim of further clustering eigenvalues of the preconditioned matrix P_0A. We have shown that this approach is particularly efficient when a sequence of linear systems has to be solved. In this case the cost of computation of the leftmost eigenvector of the preconditioned matrices is payed for by the number of system solutions. Alternatively, the vector forming the low-rank updates can be chosen as the previous solutions in the sequence. In summary we have reviewed three updating processes:

- Deflation, aimed at shifting to zero a number of approximate eigenvalues of P_0A.
- Tuning, aimed at shifting to one a number of approximate eigenvalues of P_0A.
- Spectral preconditioners, aimed at adding one to a number of approximate eigenvalues of P_0A.

The most popular choices of the vectors (such as leftmost eigenvectors of either A or P_0A) have been considered together with some techniques to cheaply assess them.

Finally we have considered a number of applications, such as discretization of evolutionary PDES and solution of large eigenvalue problems. In all these applications the low-rank correction is much beneficial to reduce the number of iterations of the iterative solver of choice.

As a general recommendation we can say that accelerating a given preconditioner by a low-rank matrix can be useful when both the following situations occur:

1. A sequence of linear systems with constant or slightly varying matrices has to be solved.
2. Either the smallest or the largest eigenvalues do not form a cluster.

Acknowledgments: This work has been partially supported by the INdAM Research group GNCS (Year 2020 project: *Optimization and Advanced Linear Algebra for Problems Governed by PDEs*) and by the project funded by CARIPARO: *Matrix-Free Preconditioners for Large-Scale Convex Constrained Optimization Problems (PRECOOP)*.

References

1. Nicolaides, R.A. Deflation of Conjugate Gradients with Applications to Boundary Value Problems. *SIAM Journal on Numerical Analysis* **1987**, *24*, 355–365.
2. Dostál, Z. Conjugate gradient method with preconditioning by projector. *International Journal of Computer Mathematics* **1988**, *23*, 315–323.
3. Mansfield, L. On the use of deflation to improve the convergence of conjugate gradient iteration. *Communications in Applied Numerical Methods* **1988**, *4*, 151–156.
4. Morgan, R.B. A Restarted GMRES method augmented with eigenvectors. *SIAM J. Matrix Anal. Appl.* **1995**, *16*, 1154–1171.
5. Frank, J.; Vuik, C. On the Construction of Deflation-Based Preconditioners. *SIAM Journal on Scientific Computing* **2001**, *23*, 442–462.
6. Gutknecht, M.H. Spectral deflation in Krylov solvers: a theory of coordinate space based methods. *Electron. Trans. Numer. Anal.* **2012**, *39*, 156–185.
7. Saad, Y.; Yeung, M.; Erhel, J.; Guyomarc'h, F. A deflated version of the conjugate gradient algorithm. *SIAM J. Sci. Comput.* **2000**, *21*, 1909–1926. Iterative methods for solving systems of algebraic equations (Copper Mountain, CO, 1998), doi:10.1137/S1064829598339761.
8. Freitag, M.A.; Spence, A. Convergence of inexact inverse iteration with application to preconditioned iterative solves. *BIT Numerical Mathematics* **2007**, *47*, 27–44. doi:10.1007/s10543-006-0100-1.
9. Freitag, M.A.; Spence, A. A tuned preconditioner for inexact inverse iteration applied to Hermitian eigenvalue problems. *IMA J. Numer. Anal.* **2008**, *28*, 522–551.
10. Bergamaschi, L.; Bru, R.; Martínez, A.; Putti, M. Quasi-Newton preconditioners for the inexact Newton method. *Electron. Trans. Numer. Anal.* **2006**, *23*, 76–87.
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.; Martínez, A. Efficiently preconditioned Inexact Newton methods for large symmetric eigenvalue problems. *Optimization Methods & Software* **2015**, *30*, 301–322.
13. 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. doi:10.1016/j.mcm.2012.01.013.
14. Bergamaschi, L.; Bru, R.; Martínez, A.; Putti, M. Quasi-Newton Acceleration of ILU Preconditioners for Nonlinear Two-Phase Flow Equations in Porous Media. *Advances in Engineering Software* **2012**, *46*, 63–68.
15. Martínez, A. Tuned preconditioners for the eigensolution of large SPD matrices arising in engineering problems. *Numer. Linear Algebra Appl.* **2016**, *23*, 427–443.
16. Freitag, M.A.; Spence, A. Shift-invert Arnoldi's method with preconditioned iterative solves. *SIAM J. Matrix Anal. Appl.* **2009**, *31*, 942–969.
17. Bergamaschi, L.; Martínez, A. Two-stage spectral preconditioners for iterative eigensolvers. *Numer. Linear Algebra Appl.* **2017**, *24*, 1–14. e2084.
18. Bergamaschi, L.; Marín, J.; Martínez, A. Compact Quasi-Newton preconditioners for SPD linear systems. *arXiv:2001.01062, math.NA* **2020**. submitted.
19. Nabben, R.; Vuik, C. A comparison of deflation and the balancing preconditioner. *SIAM J. Sci. Comput.* **2006**, *27*, 1742–1759.
20. Xue, F.; Elman, H.C. Convergence analysis of iterative solvers in inexact Rayleigh quotient iteration. *SIAM J. Matrix Anal. Appl.* **2009**, *31*, 877–899. doi:10.1137/080712908.

21. Bergamaschi, L.; De Simone, V.; di Serafino, D.; Martínez, A. BFGS-like updates of constraint preconditioners for sequences of KKT linear systems. *Numer. Linear Algebra Appl.* **2018**, *25*, 1–19. e2144.
22. Gratton, S.; Mercier, S.; Tardieu, N.; Vasseur, X. Limited memory preconditioners for symmetric indefinite problems with application to structural mechanics. *Numer. Linear Algebra Appl.* **2016**, *23*, 865–887.
23. Bergamaschi, L.; Gondzio, J.; Martínez, A.; Pearson, J.; Pougkakiotis, S. A New Preconditioning Approach for an Interior Point-Proximal Method of Multipliers for Linear and Convex Quadratic Programming. *arXiv:1912.10064, math.NA* **2019**. submitted.
24. Carpentieri, B.; Duff, I.S.; Giraud, L. A class of spectral two-level preconditioners. *SIAM J. Sci. Comput.* **2003**, *25*, 749–765 (electronic).
25. Duff, I.S.; Giraud, L.; Langou, J.; Martin, E. Using spectral low rank preconditioners for large electromagnetic calculations. *Int. J. Numer. Methods Engrg.* **2005**, *62*, 416–434.
26. Bergamaschi, L.; Facca, E.; Martínez, A.; Putti, M. Spectral preconditioners for the efficient numerical solution of a continuous branched transport model. *J. Comput. Applied Math.* **2019**, *254*, 259–270.
27. Golub, G.H.; van Loan, C.F. *Matrix Computation*; Johns Hopkins University Press: Baltimore, 1991.
28. Saad, Y. *Iterative Methods for Sparse Linear Systems. Second edition*; SIAM: Philadelphia, PA, 2003.
29. Stathopoulos, A.; Orginos, K. Computing and deflating eigenvalues while solving multiple right-hand side linear systems with an application to quantum chromodynamics. *SIAM J. Sci. Comput.* **2010**, *32*, 439–462. doi:10.1137/080725532.
30. Zhu, L.; Gong, H.; Li, X.; Wang, R.; Chen, B.; Dai, Z.; Teatini, P. Land subsidence due to groundwater withdrawal in the northern Beijing plain, China. *Engineering Geology* **2015**, *193*, 243 – 255.
31. Knyazev, A. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM J. Sci. Comput.* **2001**, *23*, 517–541.
32. Bergamaschi, L.; Gambolati, G.; Pini, G. Asymptotic Convergence of Conjugate Gradient Methods for the Partial Symmetric Eigenproblem. *Numer. Linear Algebra Appl.* **1997**, *4*, 69–84.
33. Sleijpen, G.L.G.; van der Vorst, H.A. A Jacobi-Davidson method for Linear Eigenvalue Problems. *SIAM J. Matrix Anal. Appl.* **1996**, *17*, 401–425.
34. Notay, Y. Combination of Jacobi-Davidson and conjugate gradients for the partial symmetric eigenproblem. *Numer. Linear Algebra Appl.* **2002**, *9*, 21–44. doi:10.1002/nla.246.
35. Bergamaschi, L.; Martínez, A. Generalized Block Tuned preconditioners for SPD eigensolvers. *Springer INdAM Series* **2019**, *30*, 237–252.