Computers and Mathematics with Applications 72 (2016) 1118–1130

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Interpolating preconditioners for the solution of sequence of linear systems[☆]

Daniele Bertaccini[a,*], Fabio Durastante[b]

[a] Dipartimento di Matematica, Università di Roma "Tor Vergata", via della Ricerca Scientifica 1, I-00133, Roma, Italy

[b] Dipartimento di Scienza e Alta Tecnologia, Università dell'Insubria, via Valleggio 11, 22100 Como, Italy

ARTICLE INFO

Article history:
Received 2 January 2016
Received in revised form 6 May 2016
Accepted 18 June 2016
Available online 15 July 2016

Keywords:
Preconditioners
Sparse matrices
Iterative methods

ABSTRACT

A new strategy for updating preconditioners by polynomial interpolation of factors of approximate inverse factorizations is proposed here. The computational cost per iteration is linear in the number of degree of freedom, the same order of most of the strategies for updating an incomplete factorization proposed in the last decade. The effectiveness of the technique is confirmed by some experiments.

1. Introduction

The numerical solution of sequences of large scale and sparse linear systems of algebraic equations

$$A_i x_i = b_i, \quad i = 0, 1, \ldots, s, \tag{1}$$

is ubiquitous in models for applied sciences.

Models requiring the solutions of such problems are based on partial differential equations, on systems of coupled differential equations, on hybrid differential–algebraic problems, on equilibrium problems and others. Direct solvers are robust and deterministic approaches which are often used as a standard tool in small and medium scale problems. On the other hand, iterative methods are more appropriate for large scale problems; here we concentrate on the Krylov subspace iterative solvers GMRES and BiCGstab [1]. Unfortunately, iterative solvers can fail if not preconditioned. However, if s in (1) is not small, the computation of several preconditioners can be expensive and reusing the same preconditioner not appropriate; see tests, e.g., in [2,3]. In order to overcome these issues, in recent years there has been some interest on updating preconditioners. The literature begins considering simple problems where the matrices A_i in the sequence (1) differ from one another by a scalar multiple of the identity matrix; see [4,3]. In the former A_i are symmetric M-matrices and the preconditioner an incomplete Cholesky factorization while the latter considers generic SPD matrices and updates a single approximate inverse preconditioner; see [3] for details. Then, sequences of complex symmetric matrices differing by a complex diagonal matrix are considered in [5]. A generalization considering sequences of generic symmetric positive definite matrices from a finite volume discretization of a nonlinear selective diffusion equation for imaging is considered in [6] while general nonsymmetric matrices in [2]. Sequences of nonsymmetric matrices are considered in [7] and more recently, e.g., in [8], both using strategies for updating that are different from the paradigms introduced in [3].

[☆] Work supported in part by grants MIUR-PRIN no. 20083KLJEZ and Italian Ministry of Health no. RF-2009-1470310.
[*] Corresponding author.
E-mail addresses: bertaccini@mat.uniroma2.it (D. Bertaccini), fdurastante@uninsubria.it (F. Durastante).

http://dx.doi.org/10.1016/j.camwa.2016.06.023
0898-1221/© 2016 Elsevier Ltd. All rights reserved.

Several other papers were published, also recently, on the subclass of shifted linear systems. They are not mentioned here because we focus on more general sequences of matrices.

Note that, in all the above mentioned literature, the update of a *single* preconditioner is computed by using some information on the actual matrix.

In [3, Section 6] we suggested that *matrix interpolation* could be another chance for updating preconditioner. In [9] the author proposes a first promising attempt in this sense based on linear interpolation for sequences of SPD matrices.

In this paper we propose the update of few preconditioners in factorized inverse form by interpolating the inverse factors of few decompositions. We concentrate on quadratic matrix interpolation and therefore start building our preconditioner from three preconditioners computed for three appropriately chosen different matrices in the sequence (1). However, this paradigm is fairly general and can be applied to higher degree or spline-like interpolation.

We stress that our approach requires that the matrices used for interpolation should be known in advance while this is not required in the other above mentioned "non interpolation-based" updating paradigms. On the other hand, the strategies proposed here can use the p matrices of the sequence $\{A_i\}$ to build the needed preconditioners and then use interpolation for the others.

The paper is divided as follows. In Section 2 a brief outline of algorithms for obtaining an approximate inverses in factorized form while in Section 3 we introduce our new preconditioning technique by describing it both from the algorithmic and theoretical point of view. Finally, in Section 4 we test our strategy with problem arising from the finite element approximation of boundary value problems and partial differential equations.

2. Approximate inverse preconditioners

In order to build up the underlying interpolated preconditioners, we need to provide preconditioners P_{i_j} for (a few) matrices A_{i_j}, $j = 0, 1, 2, \ldots, p$, in approximate inverse form. Given A_{i_j}, $j = 0, 1, 2, \ldots, p$ ($p = 2$ in our tests in Section 4), we need to generate $p + 1$ well defined and sparse approximations in factorized form

$$P_{i_j} = W_{i_j} D_{i_j}^{-1} Z_{i_j}^T \tag{2}$$

for $A_{i_j}^{-1}$. Here we concentrate on incomplete factorization algorithms. Given ϵ the drop tolerance of the algorithm generating the incomplete factorizations, i.e., the threshold below which the extra-diagonal elements are set to zero, it is intended that

$$\lim_{\epsilon \to 0} \|A_{i_j}^{-1} - W_{i_j} D_{i_j}^{-1} Z_{i_j}^T\| = 0, \quad j = 0, 1, \ldots, p$$

for any matrix norm $\|\cdot\|$, i.e., that for $\epsilon = 0$ the factorizations for the inverse matrices are exact.

The algorithms for approximate inverse factorization considered here belong to two classes: *inverse ILU* and *approximate inverse preconditioners* or *AINV* for short. See also [10] for recent notes and comments on their usage.

The inverse ILU technique

Inverse ILU technique for a matrix $A \in \mathbb{R}^{n\times n}$ amounts to compute first a *threshold* ILU factorization supposed well defined, see, e.g., [1]

$$P \approx \tilde{L} \tilde{D} \tilde{U}^T, \tag{3}$$

where $\tilde{L}$ and $\tilde{U}$ are unit lower triangular and $\tilde{D}$ is a diagonal matrix. We aim to approximate the factorization of the inverse of A

$$A^{-1} = WD^{-1}Z^T,$$

i.e., we look for sparse $\tilde{Z}$ and $\tilde{W}$ such that

$$\tilde{Z} \approx \hat{Z} = \tilde{L}^{-T}, \qquad \tilde{W} \approx \hat{W} = \tilde{U}^{-T}. \tag{4}$$

Consequently, P takes the form

$$P = \tilde{W}\tilde{D}^{-1}\tilde{Z}^T. \tag{5}$$

Clearly, this procedure is not feasible if the reference ILU factorization breaks down or if it is very ill-conditioned. This can happen in the case of, e.g., strongly indefinite matrices. A way to circumvent this relies often in allowing more fill-in and/or pivoting.

To get $\tilde{Z}$ and $\tilde{W}$, an approximation is needed because they can be full even if $\tilde{L}$ and $\tilde{U}$ are sparse. Under some hypothesis on P, like, e.g., diagonal dominance or being M-matrix or a suitable reordering with lumping etc., we can ensure a fast decay of the entries of the inverse factors. Nonetheless, sparsification of $\hat{Z}$ and $\hat{W}$ even with very slow (or no) decay of the entries away from the main diagonal can result in surprisingly good results in several important cases in practice. Effective schemes for computing $\tilde{Z}$ and $\tilde{W}$ consist in dropping fill as soon as it occurs within a sparse vector update, see [11,10]. In order to control the number of fills, we can easily impose a restriction on the allowed number of nonzero entries. This approach is effective on shared memory machines since the only concurrent access to the same memory location is a read type access. The inversion and sparsification part of the underlying algorithm is perfectly scalable because there are no dependencies between the calculation of the columns of the triangular factors, see again [11,10].

Approximate Inverses Preconditioners or AINV

A reliable method computing directly an approximate factorization of the inverse of the underlying matrices is the *Approximate Inverses Preconditioners* or *AINV* for short, described in [12] and used for updating preconditioners for shifted linear systems [3,5].

The method was proposed in [12] and later extended. It is based on the observation that if a matrix $A \in \mathbb{R}^{n\times n}$ is nonsingular, and if we have two vector sequences $\{z_i, i = 1 \ldots n\}$ and $\{w_i, i = 1 \ldots n\}$ which are A-biconjugate, i.e. $z_i^T A w_j = 0$ if and only if $i \neq j$, we can express the biconjugation relation as follows:

$$Z^T AW = D = \mathrm{diag}(p_1, p_2, \ldots, p_n) \tag{6}$$

where $p_i = z_i^T A w_i \neq 0$. Thus, Z and W must be nonsingular, since D is nonsingular. Therefore

$$A = Z^{-T} DW^{-1}$$

from which it readily follows that

$$A^{-1} = WD^{-1}Z^T. \tag{7}$$

If Z and W are triangular, then they are actually the inverses of the triangular factors in the *LDU* decomposition.

Sparsity in the inverse factors is obtained by carrying out the biconjugation process incompletely.

Details on the incomplete biconjugation process can be found in [12] and in [10], a recent paper revisiting some crucial aspects and issues.

3. Interpolated preconditioners

Given the underlying sequence of $n\times n$ matrices $\{A_i\}_{i=0}^p$, let us begin with the computation of three reference approximate inverse preconditioners for the matrices A_{i_0}, A_{i_1} and A_{i_2} chosen appropriately from the sequence $\{A_i\}_i$, i.e.,

$$\left\{\text{approximate inverse factorization of } A_i^{-1}\right\} = W_i D_i^{-1} Z_i^T, \quad i = i_0, i_1, i_2. \tag{8}$$

The choice of A_{i_0}, A_{i_1} and A_{i_2} is problem-dependent and it is made in order to maximize the probabilities to get a reasonable approximation for all the interpolated preconditioners. We will not focus on this aspect here, leaving more details for some specific case study in a further research.

The factorizations (8) can be produced by various algorithms. Here we focus on inversion and sparsification algorithms and AINV as revisited in [10]. We build the preconditioner factors by means of quadratic interpolation of the points (α_i, Z_i^T), (α_i, W_i), $i = i_0, i_1, i_2$, where $\{\alpha_i\}_{i=0}^s$ is the discretization of a parameter α linked to the problem and to the i indices, i.e., if we are dealing with variable time step integrator is the time step, otherwise could be some interpolation parameter or some parameter linked to the time in PDEs with variable in time coefficients. The functions for the interpolation are given by the quadratic polynomial

$$p_2(\alpha; \cdot) = a + b\alpha + c\alpha^2, \tag{9}$$

where we obtain, for a generic triplet of matrices M_1, M_2, M_3, the expression for the coefficients a, b, c:

$$a = \frac{\alpha_0 (\alpha_0 - \alpha_1) \alpha_1 M_3 + \alpha_2 (\alpha_1 (\alpha_1 - \alpha_2) M_1 + \alpha_0 (\alpha_2 - \alpha_0) M_2)}{(\alpha_0 - \alpha_1)(\alpha_0 - \alpha_2)(\alpha_1 - \alpha_2)}, \tag{10}$$

$$b = \frac{\left(\alpha_1^2 - \alpha_0^2\right) M_3 + \left(\alpha_0^2 - \alpha_2^2\right) M_2 + \left(\alpha_2^2 - \alpha_1^2\right) M_1}{(\alpha_0 - \alpha_1)(\alpha_0 - \alpha_2)(\alpha_1 - \alpha_2)}, \tag{11}$$

$$c = \frac{(\alpha_0 - \alpha_1) M_3 + (\alpha_1 - \alpha_2) M_1 + (\alpha_2 - \alpha_0) M_2}{(\alpha_0 - \alpha_1)(\alpha_0 - \alpha_2)(\alpha_1 - \alpha_2)}. \tag{12}$$

Therefore, we build the approximations for the Z_α and W_α matrices as functions of the parameter α by using equation (9) with the coefficients a, b, c computed for the $\{Z_{i_j}\}_{j=0}^2$ and $\{W_{i_j}\}_{j=0}^2$ matrices coming from the factorized approximate inverses of the reference matrices, so we have

$$Z_\alpha = p_2(\alpha; \{Z_{i_j}\}_{j=0}^2) \quad \text{and} \quad W_\alpha = p_2(\alpha; \{W_{i_j}\}_{j=0}^2). \tag{13}$$

We can take as a preconditioner the approximate inverse of the generic matrix A_i of the sequence given by

$$M_i^{-1} = W_\alpha (D_i + [Z_\alpha \, \Delta \, W_\alpha]_k)^{-1} Z_\alpha^T,$$

where D_i is the diagonal matrix coming from one of the reference preconditioner (2). The matrix Δ is given by

$$\Delta = A_i - A_{i_j}, \tag{14}$$

and the operator $[\cdot]_k$ extracts the k upper and lower diagonals of a matrix ($[\cdot]_0$ gives the main diagonal). Therefore, what we do from the computational point of view is working with $[\Delta]_k$ and the k banded approximation of Z_α and W_α. In this way, the matrix Δ is not completely computed. Then, to choose between the different reference matrices, i.e., between the different D_i, taking into account the value assumed by the α parameter for each single linear system, we take the index $i = i_j$ as the one that realizes

$$i^* = \arg \min_{i_j = i_0, i_1, i_2} \|A_i - A_{i_j}\|_F, \qquad \Delta = A_i - A_{i^*}. \tag{15}$$

Therefore we build a preconditioner update in factorized form

$$M_{i,k}^{-1} = W_\alpha (D_{i^*} + E_k)^{-1} Z_\alpha^T, \quad \text{with} \begin{cases} Z_\alpha = p_2(\alpha; \{Z_{i_j}\}_{j=0}^2), \\ W_\alpha = p_2(\alpha; \{W_{i_j}\}_{j=0}^2), \\ E_k = \left[Z_\alpha^T \Delta W_\alpha\right]_k. \end{cases} \tag{16}$$

For the *memory occupation* we observe that is contained, assuming a reasonable worst case with a non cancellation rule, at most in three times the sum of the nonzero element of the reference matrices:

$$\text{nnz}(p_2(\alpha, M_1, M_2, M_3)) \le 3 \sum_i \text{nnz}(M_i),$$

see, e.g., Fig. 1, referring to the three experiments in the next section.

Moreover, the asymptotic computational cost for the construction of the preconditioner $M_{i,k}^{-1}$ and its application is the same of linear interpolation and of the updates in [5,2,9].

Using the upper bound for condition numbers of $n \times n$ regular triangular matrices from [13] we can also get a bound for the condition number of both the matrices Z_α and W_α obtained by the quadratic interpolation formula (9).

Corollary 1 (*Bound for $\kappa_2(\cdot)$*). *Let us consider the unit upper triangular matrices of order n given by*

$$Z_\alpha = p_2(\alpha, Z_{i_0}, Z_{i_1}, Z_{i_2}), \qquad W_\alpha = p_2(\alpha, W_{i_0}, W_{i_1}, W_{i_2}). \tag{17}$$

We have that if the non-diagonal elements of Z_α have absolute values not larger than a_Z, we can bound $\kappa_2(Z)$ as

$$\begin{aligned} \|Z_\alpha^{-1}\|_2 &\le \|Z_\alpha^{-1}\|_F \le \sqrt{n\left(\frac{2}{a_Z+2}\right) + \frac{(a_Z+1)^{2n}-1}{(a_Z+2)^2}}, \\ \|Z_\alpha\|_2 &\le \|Z_\alpha\|_F \le \sqrt{n + \frac{n(n-1)}{2} a_Z^2}. \end{aligned} \tag{18}$$

If the non-diagonal elements of W_α have absolute values not larger than a_W, we can bound $\kappa_2(W)$ as

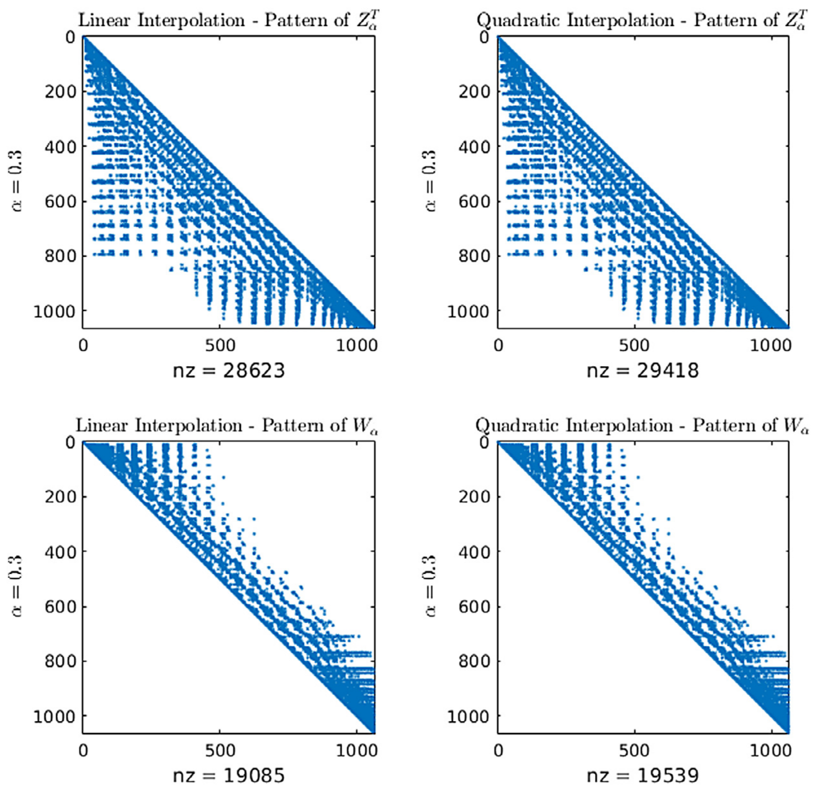
$$\begin{aligned} \|W_\alpha^{-1}\|_2 &\le \|W_\alpha^{-1}\|_F \le \sqrt{n\left(\frac{2}{a_W+2}\right) + \frac{(a_W+1)^{2n}-1}{(a_W+2)^2}}, \\ \|W_\alpha\|_2 &\le \|W_\alpha\|_F \le \sqrt{n + \frac{n(n-1)}{2} a_W^2}. \end{aligned} \tag{19}$$

Moreover, we can express the values of a_Z and a_W as

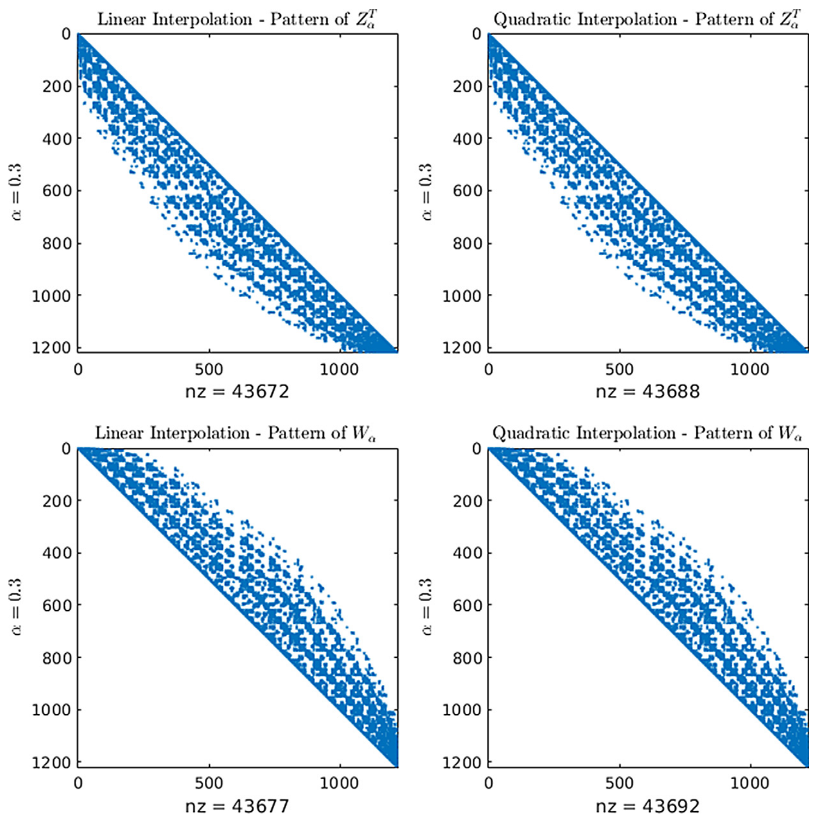
$$\begin{aligned} a_Z &= p_2\left(\alpha; \max_{i,j} |z_{i,j}^{(i_0)}|, \max_{i,j} |z_{i,j}^{(i_1)}|, \max_{i,j} |z_{i,j}^{(i_2)}|\right), \\ a_W &= p_2\left(\alpha; \max_{i,j} |w_{i,j}^{(i_0)}|, \max_{i,j} |w_{i,j}^{(i_1)}|, \max_{i,j} |w_{i,j}^{(i_2)}|\right). \quad \square \end{aligned} \tag{20}$$

The condition number of the interpolated matrices is bounded by the condition number of the reference matrices. Therefore, reference preconditioners with a reasonable condition number will potentially generate interpolated preconditioners with a reasonable condition number too.

Let us write the difference between the interpolated preconditioner (computed on the basis of the *exact* factorizations for the inverses of A_i) and its target matrix.



(a) Experiment 2.



(b) Experiment 4.

Fig. 1. Memory occupation for the Z_α^T matrices.

Lemma 2. *Let $L_iD_iU_i$ be the exact factorization of $A_i \in \mathbb{R}^{n\times n}$ and consider the quadratic interpolated matrix $M_{i,k} = L_\alpha D_\alpha U_\alpha$, where*

$$L_\alpha = p_2(\alpha, \{L_{i_j}\}_{j=0}^2), \qquad U_\alpha = p_2(\alpha, \{U_{i_j}\}_{j=0}^2), \qquad D_\alpha = D_{i^*} + E_k.$$

We have

$$M_{i,k} - A_i = (L_\alpha D_{i^*} U_\alpha - L_iD_iU_i) + (Z_\alpha^{-T} E_k W_\alpha^{-1} - \Delta),$$

where $\Delta = A_i - A_{i^*}$, as in Eq. (15) and

$$M_{i,k} - A_i = 0, \quad i = i_0, i_1, i_2, \tag{21}$$

i.e., the interpolated matrix $M_{i,k}$[1] *computed on the basis of the* exact *factorization for the inverses is exact for interpolation points corresponding to $\alpha = \alpha_i,\ i = 0, 1, 2$.*

Proof.

$$\begin{aligned} M_{i,k} - A_i &= L_\alpha(D_{i^*} + E_k)U_\alpha - (A_i - A_{i^*} + A_{i^*}) \\ &= L_\alpha D_{i^*} U_\alpha + L_\alpha E_k U_\alpha - \Delta - A_{i^*} \\ &= (L_\alpha D_{i^*} U_\alpha - A_{i^*}) + (L_\alpha E_k U_\alpha - \Delta) \\ &= (L_\alpha D_{i^*} U_\alpha - L_iD_iU_i) + (Z_\alpha^{-T} E_k W_\alpha^{-1} - \Delta). \quad \square \end{aligned}$$

We define for later use the matrix

$$C_\alpha = L_\alpha D_{i^*} U_\alpha - L_iD_iU_i. \tag{22}$$

In the style of [2,5] we prove that the updated preconditioner $M_{i,k}^{-1}$ in (16) applied to the generic matrix of the sequence A_i can cluster eigenvalues of the matrices in the underlying preconditioned sequence.

Theorem 3. *Given the sequence of linear systems $A_ix = b_i$ for $i = 0, 1, \ldots, s$, let us consider the preconditioner defined in Eq. (16). If there exists $\delta \geq 0$ and $t \ll n$ such that*

$$\begin{aligned} &Z_\alpha^{-T} E_k W_\alpha^{-1} - \Delta = U\Sigma V^T, \quad \Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n), \\ &\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_t \geq \delta > \sigma_{t+1} \geq \cdots \geq \sigma_n, \end{aligned} \tag{23}$$

and

$$\max_{\alpha\in(\alpha_1,\alpha_2)} \|D_{i^*}^{-1}E_k\| \leq \frac{1}{2}, \tag{24}$$

then there exist matrices C_α, Δ, F and a scalar constant c_α such that

$$M_{i,k}^{-1}A_i = I + M_{\alpha,k}^{-1}C_\alpha + \Delta + F, \tag{25}$$

with $\operatorname{Rank}(\Delta) = t \ll n$*, independent from α and*

$$\|F\|_2 \leq \frac{2\delta c_\alpha}{\beta} \frac{\sqrt{n\left((n-1)a_W^2 + 2\right)}\sqrt{n\left((n-1)a_Z^2 + 2\right)}}{2\,n\,b_W b_Z}, \tag{26}$$

where

$$\beta = \min\{\max_{r=1,\ldots,n} |d_r|^{-1}, c\}, \quad d_r = (D_{i^*})_{r,r} \tag{27}$$

with c constant used to avoid zero pivots in the matrix D_{α_1},

$$b_W = \min_{i,j=1,2,\ldots,n} |w_{i,j}^{(i_0)}|^{1/2}, \qquad b_Z = \min_{i,j=1,2,\ldots,n} |z_{i,j}^{(i_0)}|^{1/2}$$

and

$$\|M_{i,k}^{-1}C_\alpha\|_2 \leq 2c_\alpha \kappa_2(L_\alpha)\kappa_2(U_\alpha)\frac{\max_r |d_r|}{\min_r |d_r|} + \|M_{i,k}^{-1}A_{\alpha_1}\|_2.$$

[1] The matrix $M_{i,k}$ is called *interpolated preconditioner* if the interpolation is performed on approximate inverse decompositions for A_i.

Proof. By using the decomposition in Lemma 2 we have that

$$M_{i,k}^{-1}A_\alpha = I + M_{i,k}^{-1}C_\alpha + M_{i,k}^{-1}(Z_\alpha^{-T}E_kW_\alpha^{-1} - \Delta).$$

We can now use the hypothesis (23) on the singular value decomposition of

$$\begin{aligned} & Z_\alpha^{-T}E_kW_\alpha^{-1} - \Delta = \Delta_1 + F_1, \\ & \Delta_1 = U\,\mathrm{diag}(\sigma_1,\ldots,\sigma_t,0,\ldots,0)V^H, \\ & F_1 = U\,\mathrm{diag}(0,\ldots,0,\sigma_{t+1},\ldots,\sigma_n)V^H. \end{aligned} \tag{28}$$

Therefore we get

$$M_{i,k}^{-1}A_i = I + M_{i,k}^{-1}C_\alpha + M_{i,k}^{-1}\Delta_1 + M_{i,k}^{-1}F_1.$$

To proceed we now need to get an estimate of the norm $\|M_{\alpha,k}^{-1}\|_2$:

$$\begin{aligned} \|M_{i,k}^{-1}\|_2 &= \|W_\alpha(D_{i^*}+E_k)^{-1}Z_\alpha\|_2 \leq \|W_\alpha\|_2\,\|Z_\alpha\|_2\|(D_{i^*}+E_k)^{-1}\|_2 \\ &\leq \|W_\alpha\|_2\,\|Z_\alpha\|_2\|D_{i^*}^{-1}(I+D_{i^*}^{-1}E_k)^{-1}\|_2 \\ &\overset{\text{by}(24)}{\leq} \|W_\alpha\|_2\,\|Z_\alpha\|_2\|D_{i^*}^{-1}\|_2\left\|\sum_{n\geq 0}(-1)^n(D_{i^*}^{-1}E_k)^n\right\|_2 \\ &\overset{\text{by}(24)}{\leq} \|W_\alpha\|_2\,\|Z_\alpha\|_2\max_{r=1,2,\ldots,n}(|d_r|^{-1})c_\alpha\left(1-\|D_{i^*}^{-1}E_k\|_2\right) \\ &\leq 2c_\alpha\|W_\alpha\|_2\,\|Z_\alpha\|_2\max_{r=1,2,\ldots,n}(|d_r|^{-1}). \end{aligned}$$

We now define the matrix F of the thesis as the matrix $F = M_{i,k}^{-1}F_1$ and, by using the notation and results of Corollary 1,

$$\begin{aligned} \|F\|_2 &\leq \|M_{i,k}^{-1}\|_2\|F_1\|_2 \overset{\text{by}(23)}{\leq} \delta\|M_{i,k}^{-1}\|_2 \\ &\leq 2\delta c_\alpha\|W_\alpha\|_2\,\|Z_\alpha\|_2\max_{r=1,2,\ldots,n}(|d_r|^{-1}) \\ &\leq \frac{2\delta c_\alpha}{\beta}\frac{\|W_\alpha\|_2\,\|Z_\alpha\|_2}{\min\limits_{r=1,2,\ldots,n}\|w_{:,r}^{(\alpha_1)}\|\min\limits_{r=1,2,\ldots,n}\|z_{:,r}^{(\alpha_1)}\|} \\ &\leq \frac{2\delta c_\alpha}{\beta}\frac{\sqrt{n\left((n-1)a_W^2+2\right)}\sqrt{n\left((n-1)a_Z^2+2\right)}}{2\,n\,b_Wb_Z}. \end{aligned}$$

Let us work on the term $M_{i,k}^{-1}C_\alpha$, for which we observe that is 0 for $\alpha=\alpha_i$, $i=i_0,i_1,i_2$.

$$\begin{aligned} \|M_{i,k}^{-1}C_\alpha\|_2 &\leq \|M_{i,k}^{-1}(L_\alpha D_{i^*}U_\alpha - L_1D_{i^*}U_1)\|_2 \\ &\leq \|D_{i^*}\|_2\|M_{i,k}^{-1}\|_2\|L_\alpha\|_2\|U_\alpha\|_2 + \|M_{i,k}^{-1}A_{i_1}\|_2 \\ &\leq 2c_\alpha\kappa_2(L_\alpha)\kappa_2(U_\alpha)\frac{\max\limits_r|d_r|}{\min\limits_r|d_r|} + \|M_{i,k}^{-1}A_{i_1}\|_2. \end{aligned}$$

By introducing the low rank matrix $\Delta = M_{i,k}^{-1}\Delta_1$ we complete the proof. □

We just proved that our interpolated preconditioners can show a clustering of the spectra of the underlying matrices. This is a behavior that has to be expected in this kind of framework, where the updated preconditioner greatly relies on the reference ones and is consistent with the behavior of the condition number of the updates in Corollary 1.

4. Numerical experiments

We resume here some of the tests performed on the proposed interpolated preconditioners. We compare *AINV fixed, AINV updated, AINV interpolated and AINV interpolated* 2, where the meaning of these abbreviations is explained below.

- **AINV fixed**: compute just an approximate inverse preconditioner and use it for all the other linear systems in the sequence.
- **AINV updated**: compute the approximate inverse preconditioner for the first matrix of the sequence and update it for all the others as in [2].

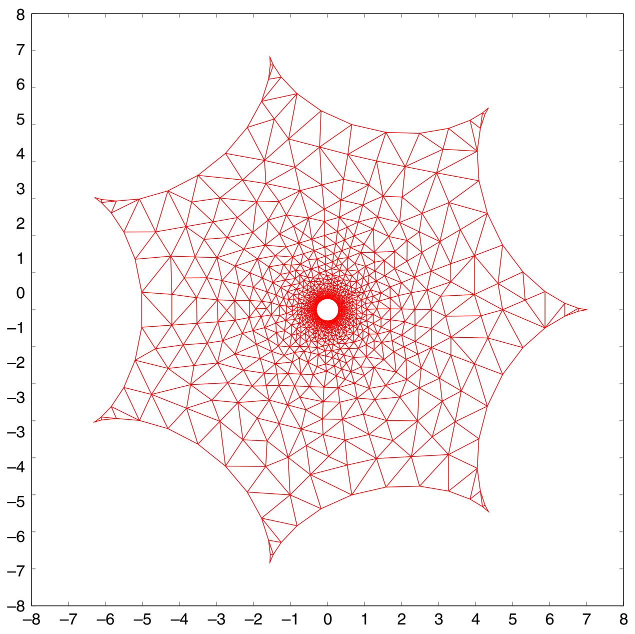


Fig. 2. Mesh for the Experiment 1.

- **AINV interpolated**: compute two approximate inverse preconditioners, one for the first matrix of the sequence and another for the last and use them in the linear matrix interpolation strategy proposed in [9].
- **AINV interpolated 2**: the quadratic polynomial matrix interpolated preconditioner introduced in Section 3.

We also tried ILU(0) and ILUT(ϵ) preconditioners with $\epsilon = 10^{-1}$ and 10^{-2}; see [1] for details. However, the results are not reported because the computed factors are too ill conditioned for all experiments below.

Note that details of tests using *recomputed preconditioners*, i.e., iterative solvers using approximate inverse preconditioners computed from scratch for each linear system, are omitted. Indeed, we experienced that their timings were always greater than all the others in the tables, regardless of the implementations we tried.

The codes are in a prototype stage using Matlab R2015a in order to simplify changes and porting to more powerful platforms. Our machine is a laptop running Linux with 8 Gb memory and CPU Intel(R) Core(TM) i7-4710HQ CPU with clock 2.50 GHz. GMRES and BiCGSTAB are considered with a relative residual stopping criteria of $\epsilon = 10^{-9}$ in order to test the performances of the preconditioners. A similar behavior is observed also with $\epsilon = 10^{-6}$ and is not reported here.

The timings do not include the cost for generating the approximate inverse factorizations because these are not computed in Matlab but use the implementation proposed in [10].

We stress again that for computing one *interpolation preconditioner* the triplet of matrices described in Section 3 should be known in advance.

4.1. Unsteady state case

We consider the finite element approximation of the convex combination ($\alpha \in [0, 1]$) of the following equations

$$\begin{cases} u_t + a_1(x,y)u_x + b_1(x,y)u_y - \nabla\cdot(k_1(x,y)\nabla u) = f_1(x,y), & (x,y) \in \Omega, \\ u_t + a_2(x,y)u_x + b_2(x,y)u_y - \nabla\cdot(k_2(x,y)\nabla u) = f_2(x,y), & (x,y) \in \Omega, \end{cases}$$

and $k_i(x,y), a_i(x,y)b_i(x,y) > 0\ \forall (x,y) \in \Omega\ i = 1, 2$, with the same boundary conditions on the borders of the domain.

Experiment 1. We start considering the boundary given in Fig. 2 parametrized by the equations

$$\mathcal{C}_1 \begin{cases} x = b\cos\left(t\left(\dfrac{a}{b}-1\right)\right) + t\cos(a-b), & a = 7, b = 1, \\ y = t\sin(a-b) - b\sin\left(t\left(\dfrac{a}{b}-1\right)\right), & t \in [0, 2\pi] \end{cases} \setminus \mathcal{C}_2 \begin{cases} x = 0.3\cos(t), \\ y = 0.3\sin(t). \end{cases}$$

The discretization is obtained by applying the backward Euler method for the time derivative and taking the test functions in the space of the piecewise linear continuous polynomials

$$P1_h = \left\{ v \in H^1(\Omega) \mid \forall K \in \mathcal{T}_h,\ \ v|_K \in \mathbb{R}_1[x] \right\}. \tag{29}$$

Table 1
Experiment 1, $\alpha = 0, 0.1, \dots, 0.9, 1$, size of the matrix $n = 1061$, GMRES algorithm.

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
IT	T	IT	T	IT	T	IT	T
1 9	0.020714	1 9	**0.005556**	1 9	0.008338	1 9	0.007451
1 75	**0.050835**	1 93	0.070675	1 149	0.208240	1 171	0.265795
1 139	**0.141679**	1 172	0.205938	1 237	0.451837	1 169	0.255071
1 188	0.243326	1 263	0.442285	1 288	0.634035	1 134	**0.177269**
1 241	0.381868	1 332	0.686958	1 296	0.665371	1 73	**0.072044**
1 258	0.430485	1 362	0.807219	1 329	0.806470	1 25	**0.016961**
1 281	0.508083	1 387	0.924151	1 342	0.867981	1 76	**0.077965**
1 301	0.576041	1 419	1.075130	1 347	0.876915	1 143	**0.194432**
1 318	0.640597	1 437	1.162376	1 354	0.910213	1 199	**0.336698**
1 335	0.704442	1 452	1.230199	1 376	1.004745	1 209	**0.365155**
1 349	0.763383	1 464	1.301722	1 403	1.159710	1 217	**0.389327**

The coefficient functions are chosen as

$$\begin{aligned}
&a_1(x) = 50(1 + 0.9\sin(100\pi x)), \qquad b_1(y) = 50(1 + 0.3\sin(100\pi y)),\\
&k_1(x, y) = x^2y^2\exp(-x^2 - y^2),\\
&a_2(x, y) = \cos^2(2x + y), \qquad b_2(x, y) = \cos^2(x + 2y), \qquad k_2(x, y) = x^4 + y^4,\\
&f_1(x, y) = f_2(x, y) = \pi^2(\sin(x) + \cos(y))
\end{aligned}$$

and boundary conditions as

$$u(x, y, t) = x, \quad (x, y) \in \mathcal{C}_1, \qquad u(x, y, t) = y, \quad (x, y) \in \mathcal{C}_2, \ \forall t \geq 0. \tag{30}$$

The initial condition is the null function over the entire domain. FreeFem++ software [14] is used.

The results of the preconditioning strategies for this problem are reported in Table 1 with *GMRES*. The reference AINV preconditioners are computed with a drop tolerance $\delta = 10^{-2}$, and only the main diagonal of matrix Δ is used, i.e., we are using $[\Delta]_k$ with $k = 1$. The values of α used for reference are $\alpha = 0, 0.5, 1$, respectively. From these experiments the performance of the fixed ILU preconditioner is omitted because, even if convergent, generates matrices close to singular or badly scaled. Also the results with the unpreconditioned GMRES are omitted because they never reach convergence due to stagnation or reach the maximum number of iteration. These results are obtained by choosing the following indices in Eq. (15) $\begin{pmatrix}1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 3 & 3 & 3\end{pmatrix}$ that fit well with the expected behavior of the interpolation, and that will be the same for all the other experiments.

Experiment 2. We consider the same settings of the previous experiment, just changing the coefficient functions as

$$\begin{aligned}
&a_1(x) = 50(1 + 0.9\sin(100\pi x)), \qquad b_1(y) = 50(1 + 0.3\sin(15\pi y)),\\
&k_1(x, y) = x^2y^2\exp(-x^2 - y^2),\\
&a_2(x) = 1 + 0.6\sin(100\pi x), \qquad b_2(y) = 1 + 0.6\sin(100\pi y), \qquad k_2(x, y) = x^2 + y^2,\\
&f_1(x, y) = f_2(x, y) = \pi^2(\sin(x) + \cos(y))
\end{aligned}$$

and the boundary conditions as

$$u(x, y, t) = 0, \quad (x, y) \in \mathcal{C}_1 \cup \mathcal{C}_2, \ \forall t \geq 0. \tag{31}$$

The results of the experiments are collected in Table 2. Again we exclude the results with the unpreconditioned GMRES in Table 2 because it never reached convergence due to stagnation.

For this case we also consider the solution with restarted GMRES, i.e. GMRES(50), with the same settings used for the references AINV preconditioners. Result for this case is collected in Table 3. Again, the results with the unpreconditioned algorithm are omitted because it never reach convergence within the maximum number of allowed iterations.

As a last test for this set of parameters we consider same settings but BiCGSTAB instead of GMRES; see Table 4. The use of BiCGSTAB without preconditioners is not reported because it never converges. The failure of the other methods in Table 4 is caused by one of the scalar quantities calculated becoming too small or too large. The fixed ILU preconditioners are again numerically singular.

4.2. *Steady state case*

We consider now finite element approximation for the steady state equation

$$\begin{cases} -\nabla \cdot (a(x, y)\nabla u) + \mathbf{b}(x, y) \cdot \nabla u + c(x, y)u = f(x, y), & (x, y) \in \Omega,\\ u = 0, & (x, y) \in \partial\Omega \end{cases} \tag{32}$$

Table 2
Experiment 2, $\alpha = 0, 0.1, \dots, 0.9, 1$, size of the matrix $n = 1061$, GMRES algorithm.

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
IT	T	IT	T	IT	T	IT	T
1 8	0.005492	1 8	0.005457	1 8	**0.005451**	1 8	0.005453
1 21	0.010126	1 21	0.010001	1 18	0.009315	1 14	**0.007712**
1 36	0.018278	1 37	0.018474	1 26	0.013336	1 15	**0.008090**
1 51	0.028517	1 54	0.030884	1 28	0.014256	1 13	**0.007350**
1 64	0.040053	1 68	0.042952	1 27	0.013719	1 10	**0.006270**
1 74	0.049302	1 79	0.055230	1 25	0.012694	1 8	**0.006658**
1 82	0.057557	1 89	0.065642	1 21	0.010649	1 10	**0.006150**
1 90	0.067051	1 99	0.078088	1 18	0.009299	1 11	**0.006465**
1 96	0.074336	1 108	0.090274	1 14	0.007616	1 11	**0.006476**
1 103	0.084157	1 116	0.102196	1 11	0.006491	1 10	**0.006114**
1 108	0.090881	1 123	0.113140	1 10	**0.006086**	1 8	0.006414

Table 3
Experiment 2, $\alpha = 0, 0.1, \dots, 0.9, 1$, size of the matrix $n = 1061$, GMRES(50).

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
IT	T	IT	T	IT	T	IT	T
1 8	0.016259	1 8	**0.003615**	1 8	0.003675	1 8	0.003687
1 21	0.008037	1 21	0.008316	1 18	0.007599	1 14	**0.006146**
1 36	0.015967	1 37	0.016629	1 26	0.011299	1 15	**0.006147**
2 1	0.027306	2 5	0.027401	1 28	0.012345	1 13	**0.005531**
2 16	0.030555	2 23	0.033895	1 27	0.011874	1 10	**0.004371**
2 30	0.037148	2 42	0.044547	1 25	0.010785	1 8	**0.003646**
2 43	0.045475	3 11	0.053968	1 21	0.008694	1 10	**0.004297**
3 7	0.052097	3 31	0.062703	1 18	0.007613	1 11	**0.004754**
3 19	0.056502	4 1	0.075398	1 14	0.005755	1 11	**0.004746**
3 30	0.061592	4 29	0.086087	1 11	0.004710	1 10	**0.004418**
3 41	0.068515	4 49	0.099530	1 10	0.004357	1 8	**0.003597**

Table 4
Experiment 2, $\alpha = 0, 0.1, \dots, 0.9, 1$, size of the matrix $n = 1061$, BiCGSTAB. †: the iterative method does not converge.

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
IT	T	IT	T	IT	T	IT	T
4.0	0.013243	4.0	**0.002515**	4.0	0.002529	4.0	0.002629
13.0	0.005550	13.5	0.005791	9.5	0.004888	7.5	**0.003944**
32.5	0.013409	33.5	0.012697	16.5	0.007601	8.5	**0.004374**
88.5	0.032187	97.5	0.035321	18.5	0.008485	6.5	**0.003553**
166.5	0.059718	190.5	0.068083	19.0	0.008605	5.0	**0.002917**
312.5	0.110927	428.5	0.151612	16.5	0.007648	5.0	**0.002782**
530.5	0.188864	260.0	†	15.5	0.007247	5.0	**0.002922**
113.0	†	121.0	†	11.0	0.005364	5.5	**0.003137**
161.0	†	1.0	†	8.0	0.004120	6.0	**0.003331**
132.0	†	1.0	†	5.5	0.003118	5.0	**0.002899**
201.0	†	1.0	†	5.0	**0.002845**	5.5	0.003164

where Ω is the domain whose boundary is parametrized by the curve

$$\begin{cases} x = \cos(t), \\ y = \sin(t)\sin^m(t/2). \end{cases} \cup \begin{cases} x = 0.01\cos(t), \\ y = 0.01\sin(t). \end{cases} \quad t \in [0, 2\pi]. \tag{33}$$

An example of the mesh is reported in Fig. 3. As a test function for the FEM method we use the elements in

$$P2_h = \left\{ v \in H^1(\Omega) \mid \forall K \in \mathcal{T}_h, \;\; v|_K \in \mathbb{R}_2[x] \right\}. \tag{34}$$

We generate a couple of problems $\{A_0, b_0\}$ and $\{A_1, b_1\}$ for different coefficient functions. The generic matrix of the sequence $\{A_\alpha\}_{\alpha=0}^{1}$ is given by the convex combination of parameter $\alpha \in [0, 1]$ of the $\{A_0, A_1\}$ matrices. The right hand sides are obtained in the same way. As for the previous set of experiments, the matrices are generated with the FreeFem++ software [14].

Experiment 3. We consider as a first experiment for the steady state case the following couple of coefficient functions

$$a_1(x, y) = x^2 + y^2, \qquad \mathbf{b}_1(x, y) = (1/2x, -1/2\sin(2\pi y)),$$
$$c_1(x, y) = x + y, \qquad f_1(x) = \cos(x);$$

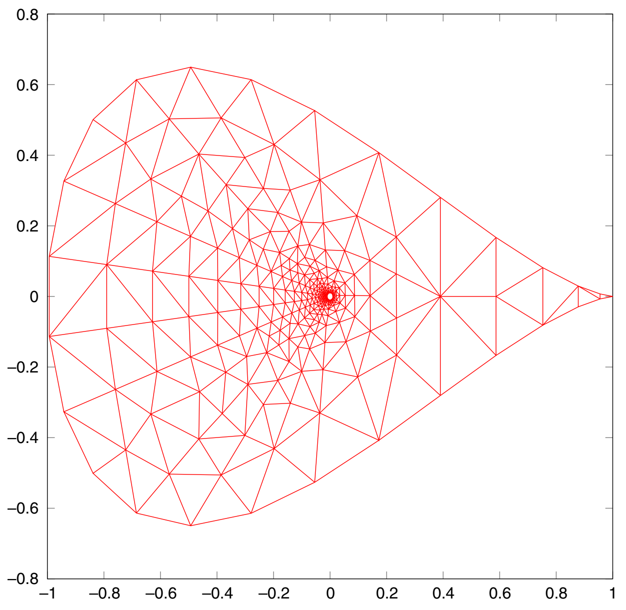


Fig. 3. Mesh for the steady state experiments.

Table 5
Experiment 3, $\alpha = 0, 0.1, \dots, 0.9, 1$, size of the matrix $n = 1218$, GMRES algorithm.

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
1 12	0.011202	1 12	**0.010653**	1 12	0.010820	1 12	0.010888
1 136	0.147877	1 79	0.062908	1 61	0.044824	1 42	**0.029075**
1 173	0.223355	1 90	0.075088	1 56	0.039877	1 30	**0.020560**
1 199	0.286868	1 96	0.083772	1 49	0.034027	1 23	**0.017202**
1 218	0.339314	1 99	0.087773	1 41	0.027457	1 19	**0.014206**
1 236	0.391216	1 99	0.088492	1 35	0.023390	1 18	**0.013939**
1 252	0.441283	1 102	0.093551	1 29	0.019613	1 18	**0.013710**
1 261	0.470766	1 103	0.094702	1 25	0.017333	1 18	**0.014060**
1 268	0.493680	1 104	0.095055	1 22	0.015509	1 18	**0.013954**
1 279	0.531869	1 104	0.094574	1 20	0.014490	1 19	**0.014323**
1 287	0.560723	1 105	0.097394	1 19	**0.013900**	1 19	0.014523

$$a_2(x,y) = x^4 + y^4, \qquad \mathbf{b}_2(x,y) = (1/2x^2 \sin(4\pi x), -1/2y^2),$$
$$c_2(x,y) = \cos(x) + \sin(y), \qquad f_2(x,y) = \exp(-x-y).$$

We use *GMRES* without restart. The reference AINV preconditioners are computed with a dropping tolerance of $\delta = 1e-2$. For what concerns the correction matrix Δ, we stress that again only the main diagonal is considered. The results of the experiment are reported in Table 5. Similarly to the other experiments, the behavior of the fixed ILU preconditioner is not reported due to factors numerically singular. Moreover, the GMRES algorithm without preconditioning stagnates in all cases.

As for the PDE experiment we test the strategy also with GMRES(50). The results for this experiment are collected in Table 6. As expected, also unpreconditioned GMRES(50) does not converge and ILU preconditioners do not work.

Finally, we test our preconditioning strategy by using BiCGSTAB. The results are collected in Table 7. Also nonpreconditioned BiCGSTAB stagnates in all the instances. Moreover, ILU preconditioners are again numerically singular.

Experiment 4. We consider the following coefficient function for the generation of the A_0 and A_1 matrices

$$a_1(x,y) = \exp(-x^2 - y^2), \qquad \mathbf{b}_1(x,y) = (1/2x^2, 1/2y^2),$$
$$c_1(x,y) = \exp(x+y)\sin(x+y), \qquad f_1(x) = x^2 + y^2;$$
$$a_2(x,y) = x^4 + y^4, \qquad \mathbf{b}_2(x,y) = (1/2x^2, -y\sin(y)),$$
$$c_2(x,y) = \exp(-x-y)\cos(x+y), \qquad f_2(x,y) = \exp(-x-y);$$

The results of this experiment, obtained with the same settings of the preconditioner as in the previous experiment, are collected in Table 8. Also in this case results relative to ILU preconditioning are omitted for the same reason of the other cases, same is obtained with the unpreconditioned GMRES algorithm, i.e., stagnation.

Table 6
Experiment 3, $\alpha = 0, 0.1, \dots, 0.9, 1$, size of the matrix $n = 1218$, GMRES(50).

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
IT	T	IT	T	IT	T	IT	T
1 12	0.005275	1 12	0.005428	1 12	**0.005263**	1 12	0.005382
4 31	0.094927	2 36	0.044897	2 15	0.034914	1 42	**0.022818**
5 36	0.125520	2 49	0.054426	2 8	0.032187	1 30	**0.014405**
6 45	0.157440	3 7	0.057231	1 49	0.028282	1 23	**0.010443**
7 26	0.174086	3 10	0.058787	1 41	0.022117	1 19	**0.008493**
7 37	0.178430	3 10	0.058439	1 35	0.017715	1 18	**0.007901**
8 50	0.217248	3 16	0.060779	1 29	0.013693	1 18	**0.007903**
9 38	0.232914	3 23	0.064202	1 25	0.011561	1 18	**0.007959**
10 4	0.245124	3 25	0.065636	1 22	0.009922	1 18	**0.007901**
10 15	0.247101	3 27	0.066140	1 20	0.008901	1 19	**0.008453**
11 4	0.270208	3 28	0.066834	1 19	0.008422	1 19	**0.008419**

Table 7
Experiment 3, $\alpha = 0, 0.1, \dots, 0.9, 1$, size of the matrix $n = 1218$, BiCGSTAB.

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
IT	T	IT	T	IT	T	IT	T
8.0	0.016040	8.0	0.004520	8.0	**0.004680**	8.0	0.005073
95.5	0.041435	39.0	0.016808	33.5	0.016366	21.0	**0.010530**
136.5	0.056062	49.5	0.021100	29.0	0.014159	15.5	**0.008114**
168.0	0.069001	53.0	0.022541	24.0	0.011895	12.5	**0.006647**
200.0	0.081733	56.0	0.023725	20.5	0.010744	10.5	**0.005763**
251.5	0.102740	53.0	0.022485	18.0	0.009162	9.5	**0.005236**
232.0	0.094478	60.5	0.025640	14.5	0.007588	9.5	**0.005254**
250.5	0.103164	54.0	0.022944	13.0	0.006911	10.0	**0.005524**
257.5	0.105048	60.5	0.025658	11.5	0.006205	10.0	**0.005446**
267.5	0.108782	67.0	0.028204	10.5	0.005757	10.0	**0.005530**
302.5	0.122927	62.0	0.026282	10.5	**0.005720**	10.5	0.005748

Table 8
Experiment 4, $\alpha = 0, 0.1, \dots, 0.9, 1$, size of the matrix $n = 1218$, GMRES.

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
IT	T	IT	T	IT	T	IT	T
1 11	0.008624	1 11	0.008237	1 11	**0.007847**	1 11	0.009121
1 782	3.858140	1 145	0.162046	1 108	0.103187	1 68	**0.052373**
1 828	4.336583	1 146	0.165280	1 83	0.069604	1 42	**0.028622**
1 855	4.615671	1 146	0.163970	1 66	0.049852	1 29	**0.019464**
1 877	4.849502	1 146	0.165380	1 53	0.037416	1 22	**0.015385**
1 894	5.029727	1 146	0.164791	1 44	0.030029	1 21	**0.014714**
1 907	5.135682	1 146	0.163435	1 36	0.023935	1 21	**0.014809**
1 919	5.298071	1 146	0.163200	1 29	0.019184	1 21	**0.014793**
1 932	5.462130	1 145	0.160476	1 25	0.016687	1 22	**0.015331**
1 946	5.632789	1 145	0.161281	1 22	0.015117	1 21	**0.014857**
1 958	5.866944	1 144	0.159491	1 21	**0.014538**	1 21	0.014756

We tested also the underlying preconditioners with the same settings with GMRES(50) and collected the results in Table 9. As for all the other cases, there is no convergence both with the fixed ILU preconditioner or GMRES(50) without preconditioning.

As a final test, we consider the application of interpolated preconditioners with BiCGSTAB. The results are in Table 10. Also in this case BiCGSTAB without preconditioning reaches the maximum number of iteration without converging, while the application of the fixed ILU preconditioners presents the same issues of the fixed AINV preconditioner.

5. Conclusions

We proposed a paradigm for the interpolation of (the factors of) approximate inverse preconditioners for sequences of large and sparse linear systems.

Our technique shows promising performances for some problems. We tested it for a convection–diffusion steady state and time-dependent problem approximated by using finite elements. In general, performances seem to be not worse than updated preconditioners introduced and studied in various settings in [3,5,2].

Table 9
Experiment 4, $\alpha = 0, 0.1, \dots, 0.9, 1$, size of the matrix $n = 1218$, GMRES(50). †: the iterative method does not converge.

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
IT	T	IT	T	IT	T	IT	T
1 11	0.005073	1 11	**0.004839**	1 11	0.004939	1 11	0.004866
1000 50	†	5 18	0.112649	3 49	0.086260	2 34	**0.047497**
1000 50	†	5 36	0.123246	3 1	0.059262	1 42	**0.023581**
1000 50	†	5 38	0.123619	2 30	0.044087	1 29	**0.014554**
341 50	†	5 39	0.124417	2 4	0.031545	1 22	**0.010464**
1000 50	†	5 42	0.127286	1 44	0.025142	1 21	**0.009816**
1000 50	†	5 43	0.126812	1 36	0.019024	1 21	**0.009900**
1000 50	†	5 44	0.128279	1 29	0.014564	1 21	**0.009868**
1000 50	†	5 44	0.128254	1 25	0.012129	1 22	**0.010494**
984 50	†	5 44	0.128555	1 22	0.010469	1 21	**0.009951**
246 50	†	5 44	0.128312	1 21	**0.009866**	1 21	0.009931

Table 10
Experiment 4, $\alpha = 0, 0.1, \dots, 0.9, 1$, size of the matrix $n = 1218$, BiCGSTAB. †: the iterative method does not converge.

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
IT	T	IT	T	IT	T	IT	T
7.0	0.016874	7.0	0.004219	7.0	0.004019	7.0	**0.003948**
†	†	102.5	0.043997	75.0	0.038548	46.0	**0.024265**
†	†	104.5	0.043005	58.0	0.030111	26.5	**0.014448**
†	†	113.0	0.046021	45.0	0.023534	19.0	**0.010779**
†	†	103.5	0.042312	36.5	0.019406	14.5	**0.008485**
†	†	106.5	0.043577	28.5	0.015369	13.5	**0.007972**
†	†	105.5	0.043032	23.0	0.012523	13.0	**0.007716**
†	†	108.0	0.044277	18.5	0.010351	14.0	**0.008180**
†	†	107.5	0.043890	16.0	0.009031	14.0	**0.008230**
†	†	110.0	0.045023	14.0	**0.008032**	14.0	0.008297
†	†	110.5	0.045788	13.0	**0.007490**	13.5	0.007947

The computational cost per iteration is similar to updated preconditioners while the initial setting requires the computation of two more approximate inverse factorizations for the quadratic interpolation paradigm.

Generalizations of our interpolating strategies can be based on the use of higher order or of splines-like matrix interpolation. Moreover, the choice of the starting preconditioners and therefore of the matrices in the sequence (1) can be important and we stress that it is a very problem dependent issue. Both topics will be considered in a future work.

Acknowledgments

We are grateful to the referees for useful comments which have improved this presentation.

References

[1] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed., SIAM, Philadelphia, PA, 2003.
[2] S. Bellavia, D. Bertaccini, B. Morini, Nonsymmetric preconditioner updates in Newton Krylov methods for nonlinear systems, SIAM J. Sci. Comput. 33 (2011) 2595–2619.
[3] M. Benzi, D. Bertaccini, Approximate inverse preconditioning for shifted linear systems, BIT 43 (2003) 231–244.
[4] G. Meurant, On the incomplete cholesky decomposition of a class of perturbed matrices, SIAM J. Sci. Comput. 23 (2001) 419–429.
[5] D. Bertaccini, Efficient preconditioning for sequences of parametric complex symmetric linear systems, Electron. Trans. Numer. Anal. 18 (2004) 49–64.
[6] D. Bertaccini, F. Sgallari, Updating preconditioners for nonlinear deblurring and denoising image restoration, Appl. Numer. Math. 60 (2010) 994–1006.
[7] J.D. Tebbens, M. Tuma, Efficient preconditioning of sequences of nonsymmetric linear systems, SIAM J. Sci. Comput. 29 (2007) 1918–1941.
[8] S. Bellavia, B. Morini, M. Porcelli, New updates of incomplete lu factorizations and applications to large nonlinear systems, Optim. Methods Softw. 29 (2014) 321–340.
[9] F. Durastante, Interpolant update of preconditioners for sequences of large linear systems, in: Mathematical Methods, Computational Techniques and Intelligent Systems, Vol. 41, (MAMECTIS '15), WSEAS Press, 2015, pp. 40–47.
[10] D. Bertaccini, S. Filippone, Sparse approximate inverse preconditioners on high performance {GPU} platforms, Comput. Math. Appl. 71 (2016) 693–711.
[11] A.C. Van Duin, Scalable parallel preconditioning with the sparse approximate inverse of triangular matrices, SIAM J. Matrix Anal. Appl. 20 (1999) 987–1006.
[12] M. Benzi, M. Tůma, A sparse approximate inverse preconditioner for nonsymmetric linear systems, SIAM J. Sci. Comput. 19 (1998) 968–994.
[13] F. Lemeire, Bounds for condition numbers of triangular and trapezoid matrices, BIT 15 (1975) 58–64.
[14] F. Hecht, New development in freefem++, J. Numer. Math. 20 (2012) 251–265.