Spectral preconditioners for the efficient numerical solution of a continuous branched transport model

L. Bergamaschi [a], E. Facca [b], Á. Martínez [b,*], M. Putti [b]

[a] Department of Civil, Environmental and Architectural Engineering, University of Padova, via Marzolo 9, 35131, Padova, Italy
[b] Department of Mathematics "Tullio Levi-Civita", University of Padova, via Trieste 63, 35100 Padova, Italy

ARTICLE INFO

Keywords:
Optimal transport
Linear systems solving
PCG method
Preconditioning

ABSTRACT

We consider the efficient solution of sequences of linear systems arising in the numerical solution of a branched transport model whose long time solution for specific parameter settings is equivalent to the solution of the Monge–Kantorovich equations of optimal transport. Galerkin Finite Element discretization combined with explicit Euler time stepping yield a linear system to be solved at each time step, characterized by a large sparse very ill conditioned symmetric positive definite (SPD) matrix A. Extreme cases even prevent the convergence of Preconditioned Conjugate Gradient (PCG) with standard preconditioners such as an Incomplete Cholesky (IC) factorization of A, which cannot always be computed. We investigate several preconditioning strategies that incorporate partial approximated spectral information. We present numerical evidence that the proposed techniques are efficient in reducing the condition number of the preconditioned systems, thus decreasing the number of PCG iterations and the overall CPU time.

1. Introduction

We are interested in the numerical solution of the Partial Differential Equation (PDE) based dynamic Optimal Transport (OT) model proposed in [1,2]. The model couples an elliptic equation enforcing mass conservation with an ordinary differential equation governing the dynamics of the transport density. The resulting system possesses a long-time equilibrium point that is conjectured to be the solution of the Monge–Kantorovich PDEs as formulated in [3]. Natural extensions of this dynamic model present long-time evolutions that approach fractal structures reminiscent of the solutions of Branched (or ramified) optimal Transport Problems (BTP) as defined in [4,5]. In BTP, optimality favors aggregate mass movement, spontaneously developing branching structures typically found in communication networks and natural systems [6]. These problems find origins from discrete formulations, typically defined on graphs, as pioneered in the field of communication networks by [7]. More recently, continuous formulations have been proposed [4,8] as limits of discrete counterparts. Not secondarily, these formulations have been proposed as relaxed frameworks for the determination of optimal networks, a typically NP hard problem [8].

The congenital irregularity of the structures resulting from these formulations, typically tree-shaped, poses important numerical challenges. Obvious difficulty arises in the numerical approximation of almost-everywhere discontinuous functions, commonly solved via regularization [8]. In our case, the discontinuous transport density is the diffusion coefficient of the elliptic equation, and determines its coercivity. Numerical discretization by finite elements in space and explicit Euler in time

* Corresponding author.
E-mail addresses: luca.bergamaschi@unipd.it (L. Bergamaschi), facca@math.unipd.it (E. Facca), angeles.martinez@unipd.it (Á. Martínez), mario.putti@unipd.it (M. Putti).

leads to a sequence of sparse linear systems with conditioning influenced by the variability of the transport density. Extreme cases even prevent the convergence of PCG with standard preconditioners such as an IC (with partial fill-in) factorization of A, which cannot always be computed.

In this paper, we are interested in developing efficient preconditioners that explicitly take into consideration the spatial and temporal variability of the transport density by calculated spectral information of the involved stiffness matrices. The idea of using partial spectral knowledge to accelerate linear system solvers has been described in several papers such as [9–11] and, more recently in [12]. In all these papers the authors start with an initial preconditioner P_0 and use an approximation of a few eigenvectors of the preconditioned matrix to update P_0 with a low-rank matrix. Another characteristic shared by all these previous papers is that the coefficient matrix of the linear systems to be solved $A\boldsymbol{x}_k = \boldsymbol{b}_k$ remains unchanged throughout the whole sequence. This allows the incremental refinement of the set of eigenvectors used to update the low-rank correction matrix. In this work we consider instead sequences of sparse linear systems with changing coefficient matrices dynamically depending on the transport density. We present numerical evidence that the proposed techniques are efficient in reducing the condition number of the preconditioned systems, thus decreasing the number of PCG iterations and the CPU time.

The remaining of the paper is organized as follows: Section 2 describes the BTP model. Section 3 introduces the spectral preconditioner and two different strategies to obtain an approximated set of eigenvectors needed to form the low rank updating matrix. Section 4 describes the implementation details regarding the construction of the preconditioner. We also include in this section detailed algorithms of the iterative solution phase by the PCG method for the two different spectral information recovering techniques. Numerical results are shown in Section 5. Section 6 summarizes the main conclusions.

2. The branched transport model

We consider the extension of the dynamic L^1 Monge–Kantorovich OT problem formulated by [1]: find (μ, u) such that:

$$-\nabla\cdot(\mu(t,x)\nabla u(t,x)) = f(x) = f^+(x) - f^-(x) \tag{1}$$

$$\mu'(t,x) = (\mu(t,x)|\nabla u(t,x)|)^\beta - \mu(t,x) \tag{2}$$

$$\mu(0,x) = \mu_0(x) > 0 \tag{3}$$

where μ is the transport density, u is the transport potential, f is the zero-average forcing function (f^+ and f^- being the initial and final mass configurations), and $\beta > 0$ is related to the branching exponent. Here we assume that $x \in \Omega \subset \mathbb{R}^d$ ($d = 2$ in our case), with the spatial domain Ω characterized by a sufficiently regular boundary $\Gamma = \partial\Omega$, and $t \in [0, T]$. Zero Neumann conditions are imposed on the entire boundary Γ, and the gradient and divergence operators are computed with respect to the spatial coordinate x, while μ' indicates time differentiation. For $\beta = 1$, [1] conjecture that the large time solution ($T \to \infty$) of the above problem is equivalent to the solution of the PDE-based Monge–Kantorovich equations of optimal transport [3]. For $\beta > 1$, experimental evidence suggests that the above formulation reaches and equilibrium state that resembles BTP solutions. Fig. 1 reports some spatial configurations of the numerical approximation μ_h^* of the equilibrium transport density μ^* for different values of $\beta \geq 1$ when displacing a given mass from the left (f^+) to the right (f^-) rectangle (left column), or from 30 Dirac Delta points randomly distributed on the unit square towards one single point located on the lower left portion of the domain (right column). In all these tests, the numerical approximation of the transport density displays a tree-like structure, approaching a pre-imposed lower bound of 10^{-10} in most part of Ω and showing a drastically increasing upper bound at higher values of the exponent β.

Numerical discretization is achieved by means of Galerkin FEM in space and explicit Euler in time, and is a straightforward extension of the procedure described for $\beta = 1$ in [2]. Two different triangulations, $\mathcal{T}_h$ and $\mathcal{T}_{h/2}$, the second one obtained from the first by uniform refinement, are used to approximate the unknowns. The transport density μ is approximated by μ_h on $\mathcal{T}_h$ with piecewise constant polynomials ($\mathcal{P}_0(\mathcal{T}_h)$), while the transport potential u is approximated by u_h on $\mathcal{T}_{h/2}$ by means of piecewise linear polynomials ($\mathcal{P}_1(\mathcal{T}_{h/2})$). Hence, the first equation is projected on the FEM space $\mathcal{V}_1 \subset \mathcal{P}_1(\mathcal{T}_{h/2})$ and the second equation on $\mathcal{V}_0 \subset \mathcal{P}_0(\mathcal{T}_h)$. An explicit backward Euler method is used for the time discretization, thus introducing a stability limitation on the time-step that is controlled empirically.

The complete numerical solution algorithm proceeds as follows. Given $\mu_h^{(0)} = \Pi_h\mu_0$, Π_h being the L^2 projector on $\mathcal{T}_h$, solve for $k = 0, 1, \ldots$ until equilibrium:

$$A[\boldsymbol{\mu}^{(k)}]\boldsymbol{u}^{(k)} = b \tag{4}$$

$$\boldsymbol{\mu}^{(k+1)} = \boldsymbol{\mu}^{(k)} + \Delta t^k\left[B[\boldsymbol{u}^{(k)}]\left(\boldsymbol{\mu}^{(k)}\right)^\beta - \boldsymbol{\mu}^{(k)}\right] \tag{5}$$

where $A[\boldsymbol{\mu}^{(k)}]$ is the $\mathcal{P}_1$-stiffness matrix evaluated at $\boldsymbol{\mu}^{(k)}$, vector $\boldsymbol{\mu}$ collects the elemental values of μ_h, $\boldsymbol{u}$ is the vector of nodal values of u_h, and B is the matrix defining the norm of the gradient of u_h raised to the power β. Equilibrium is considered achieved by repeating the above algorithm until

$$\frac{\|\mu_h^{(k+1)} - \mu_h^{(k)}\|_{L^2(\Omega)}}{\Delta t^k\|\mu_h^{(k)}\|_{L^2(\Omega)}} < \tau.$$

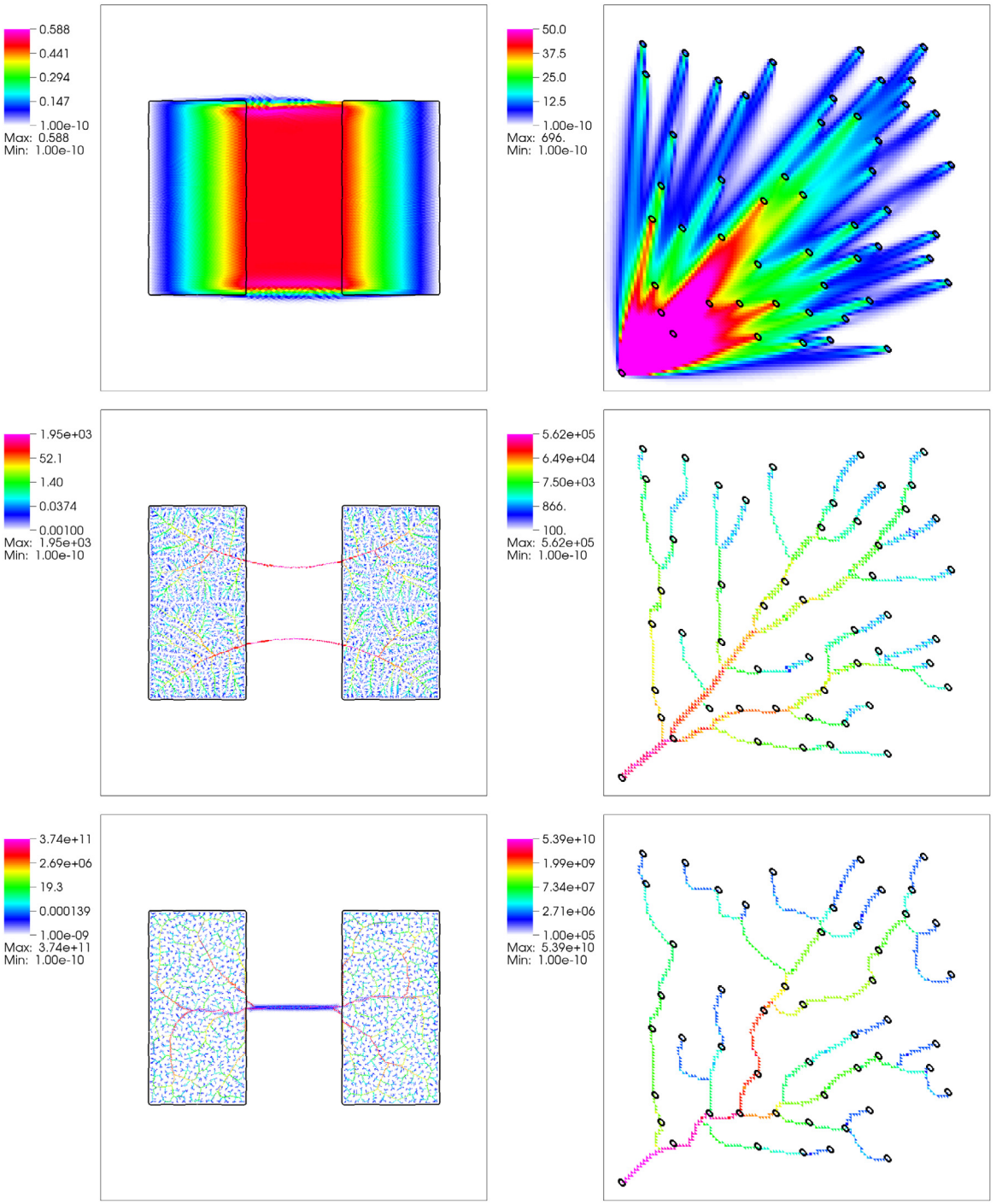

Fig. 1. Spatial distribution of μ_h^* obtained with two different forcing terms. The left column shows a piecewise unitary forcing function, where the black rectangles indicate the supports of f^+ (left) and f^- (right). In right column f^+ is the sum of 30 Dirac sources randomly distributed in the square $[0.1, 0.9] \times [0.1, 0.9]$, while f^- is concentrated in $P = (0.05, 0.05)$ and is the sum of the 30 Dirac sources. Different β are represented by rows, with values ranging from top to bottom from $\beta = 1.0$, corresponding to the L^1 case, $\beta = 1.5$ and $\beta = 5.0$. Note that in the last two panels on the left column we used a color scale that starts from the minimum value of $\mu_h^* = 10^{-10}$, in order to highlight the different values on the branches. The white areas indicate where the minimal value is achieved. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

We generally use $\tau = 10^{-4}$, value that ensures no further changes in the topological structure of the results and very small variations in the system matrix $A[\boldsymbol{\mu}^{(k)}]$. The above algorithm requires the solution of the large, sparse, symmetric, and positive definite linear system (4). Because of typically large dimensions, we use a preconditioned conjugate gradient

(PCG) solver. PCG convergence becomes increasingly difficult as time progresses since the condition number of the system matrix grows with β. In fact, the dynamics of the model is such that μ_h tends to zero in large portions of Ω. To avoid non-coerciveness of the elliptic partial differential equation, we impose a minimum threshold for μ_h equal to 10^{-10}. However, the maximum value of μ_h increases as its support concentrates along thinner paths appearing for increasing values of β. Since $\lambda_{\min}(A) \leq C_1 h^2 \mu_{\min}$ and $\lambda_{\max}(A) \geq C_2 \mu_{\max}$ [13], with the constants depending on the domain Ω and the triangulations $\mathcal{T}_h$ and $\mathcal{T}_{h/2}$, the condition number of $A[\mu_h^k]$ increases with time, possibly leading to non-convergence of the PCG iteration with a standard preconditioner such as an Incomplete Cholesky (IC) factorization with partial fill-in.

We observe experimentally that the sequence of system matrices varies relatively slowly in time as a function $\Delta t^{(k)}$. Hence, we would like to exploit our knowledge on the system matrix sequence to devise efficient preconditioners for PCG. In the paper we investigate several strategies that incorporate incomplete spectral information [14] on previous matrices to update the preconditioner for the current and future system.

3. The spectral preconditioner

In this paper we consider sequences of linear systems of the form

$$A_k \boldsymbol{x}_k = \boldsymbol{b}, \tag{6}$$

where $A_k \in \mathbb{R}^{n\times n}$ is an SPD matrix, $\boldsymbol{x}_k,\ \boldsymbol{b} \in \mathbb{R}^n$. For a given linear system $A_k \boldsymbol{x}_k = \boldsymbol{b}$ we study the acceleration of the PCG solver provided by the following spectral preconditioner:

$$P = P_0 + V_p \Lambda_p^{-1} V_p^T, \tag{7}$$

where $V_p = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p]$ and $\boldsymbol{v}_j, j = 1, \ldots, p$ are approximate leftmost eigenvectors either of $P_0 A_k$ or of A_k; $\Lambda_p = \mathrm{diag}(\lambda_1, \ldots, \lambda_p)$, and $\lambda_j,\ j = 1, \ldots, p$ are the corresponding eigenvalues. When V_p contains eigenvectors of $P_0 A_k$, the effect of the low-rank correction is easily shown to be:

$$P A_k \boldsymbol{v}_j = (\lambda_j + 1)\boldsymbol{v}_j, \qquad j = 1, \ldots, m,$$

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

We propose two different ways to obtain the approximated eigenvectors needed to construct the spectral preconditioner: evaluating the sought eigenpairs with an external eigensolver (Deflation-Accelerated Conjugate Gradient, DACG) or approximating them directly from the PCG iterations at previous time-steps. For simplicity, from now on we will write A for A_k when no confusion arises.

3.1. Approximating the smallest eigenpairs by DAGC

Algorithm 1 DACG method

- INPUT: tolerance τ_{DACG}, P_0, p. Set $V_p = 0$.
- FOR $j = 1$ TO p
  1. Choose a unit $2-$norm $\boldsymbol{x}_0$ such that $V_p^T \boldsymbol{x}_0 = 0$;
  2. Find the minimum of the RQ over all $\boldsymbol{x}$ such that $V_p^T \boldsymbol{x} = 0$ by a nonlinear PCG procedure, with starting point $\boldsymbol{x}_0$ and preconditioner P_0. Stop whenever the following test is satisfied:
  $$\frac{\|A\boldsymbol{x}_{\mathrm{DACG}} - q(\boldsymbol{x}_{\mathrm{DACG}})\boldsymbol{x}_{\mathrm{DACG}}\|}{q(\boldsymbol{x}_{\mathrm{DACG}})\|\boldsymbol{x}_{\mathrm{DACG}}\|} \leq \tau_{\mathrm{DACG}} \tag{8}$$
  3. Set $\lambda_j = q(\boldsymbol{x}_{\mathrm{DACG}}), \quad \boldsymbol{v}_j = \dfrac{\boldsymbol{x}_{\mathrm{DACG}}}{\|\boldsymbol{x}_{\mathrm{DACG}}\|}, \quad V_p = [V_p,\ \boldsymbol{v}_j]$.
- END FOR

Following [14], we propose to approximate some of the leftmost eigenvectors of a given coefficient matrix A_k by performing some preliminary iterations of an eigenvalue solver. We chose the DACG eigensolver [15,16], which is based on the preconditioned conjugate gradient (nonlinear) minimization of the Rayleigh Quotient (RQ) $q(\boldsymbol{x}) = \boldsymbol{x}^T A\boldsymbol{x}/\boldsymbol{x}^T\boldsymbol{x}$. The leftmost eigenpairs are computed sequentially, by minimizing RQ over a subspace orthogonal to the previously computed eigenvectors. This method, which applies only to SPD matrices, has been proven very efficient in the solution of eigenproblems arising from discretization of PDEs in [16]. DACG also proved very suited to parallel implementation as documented in [17] where an efficient parallel matrix–vector product has been employed. Our implementation of DACG is shown in Algorithm 1. The main computational cost of one DACG iteration is given by:

1. One matrix–vector product.
2. One application of the preconditioner.
3. Orthogonalization of the search direction against the previously computed eigenpairs (columns of matrix V_p). The cost of this step is increasing with the number of eigenpairs begin sought.

Convergence of DACG is strictly related to the relative separation between consecutive eigenvalues, namely

$$\xi_j = \frac{\lambda_j}{\lambda_{j+1} - \lambda_j}. \tag{9}$$

When two eigenvalues are relatively close, DACG convergence may be very slow. Also DACG takes advantage of preconditioning, which in our case is chosen to be the IC factorization of matrix A.

Once a small number of leftmost eigenvectors has been computed and stored as columns of V_p, different low-rank corrections of a given preconditioner P_0 can be defined as e.g. described in [18]. For example a BFGS-style preconditioner can be written as

$$\begin{aligned} P &= V_p(V_p^T A V_p)^{-1} V_p^T + \left(I - V_p(V_p^T A V_p)^{-1} V_p^T A\right) P_0 \left(I - A V_p (V_p^T A V_p)^{-1} V_p^T\right) \\ &\approx V_p \Lambda_p^{-1} V_p^T + \left(I - V_p V_p^T\right) P_0 \left(I - V_p V_p^T\right). \end{aligned} \tag{10}$$

A simplified version of this BFGS preconditioner neglects the left and right projectors on P_0, and thus takes the same form as in (7):

$$P = V_p \Lambda_p^{-1} V_p^T + P_0.$$

It can be shown [18] that the preconditioned matrix PA has a better spectral distribution than $P_0 A$.

3.2. Recovering spectral information by the Lanczos process

Another strategy we are going to use is to recover the partial eigenspectrum of A from the Krylov subspace built by the linear solver, using the Lanzcos process embedded within the PCG algorithm. Denoting again by P_0 an initial preconditioner for matrix A, during the PCG method we save the first m preconditioned residuals as columns of a matrix W_m:

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

Matrix W_m is such that $W_m^T P_0^{-1} W_m = I_m$, in view of the P_0-orthogonality of the residuals generated by the PCG method. Moreover, we can form the Lanczos tridiagonal matrix using the PCG coefficients α_k, β_k as follows:

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

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

$$W_m^T A W_m = T_m.$$

After eigensolving T_m we obtain $T_m = Q \Lambda_m Q^T$, where the coefficients of the diagonal matrix Λ_m approximate the eigenvalues of $P_0 A$ while the columns of $V_p = W_m Q_p$ (where Q_p contains the first p columns of Q) are approximations of the p leftmost eigenvectors of $P_0 A$. In fact, first note that $V_p^T A V_p = Q_p^T W_m^T A W_m Q_p = Q_p^T T_m Q_p = \Lambda_p \equiv \mathrm{diag}(\lambda_1, \ldots, \lambda_p)$. Then, let $U = P_0^{-1/2} V_p$ we obtain:

$$U^T U = V_p^T P_0^{-1} V_p = I_m \tag{11}$$

$$\Lambda_p = V_p^T A V_p = U^T P_0^{1/2} A P_0^{1/2} U \tag{12}$$

corresponding to the Lanczos process applied to matrix $P_0^{1/2}AP_0^{1/2}$. Hence the columns of U approximate the eigenvectors of $P_0^{1/2}AP_0^{1/2}$ and the columns of V_p approximate the eigenvectors of P_0A, as can be seen from the following relationships:

$$P_0^{1/2}AP_0^{1/2}U \approx U\Lambda_p \iff P_0AP_0^{1/2}U \approx P_0^{1/2}U\Lambda_p$$
$$\iff P_0AV_p \approx V_p\Lambda_p.$$

4. Implementation

Approximation of a number of leftmost eigenpairs is a costly task and cannot be performed at each linear system solution. To reduce the impact of this cost on the overall process we devise different strategies depending on how we obtain the spectral information.

4.1. Initial preconditioner P_0

For all the experiments the initial preconditioner is an IC preconditioner obtained by setting the maximum number of nonzero elements per row LFIL= 30 and a drop tolerance $\tau_{IC} = 10^{-4}$. The use of a smaller LFIL and/or a larger τ_{IC} does not guarantee the existence of the IC factorization for all systems leading to a breakdown of the simulations. This choice of parameters produced a rather dense Cholesky factor with a number of nonzero elements roughly 8 times that of the triangular part of A. For this reason, the computation of this preconditioner for each linear system of the sequence was not effective. We decided to compute the IC preconditioner for a given matrix A_k if $k = 1$ or the number of PCG iterations in the previous linear system was above a fixed value, it_{chol}. We used the previously computed IC preconditioner, otherwise.

4.2. Eigenpairs of A obtained by DACG

The computation of a number of the leftmost eigenpairs by DACG is a preprocessing stage that in principle should be executed prior to every system solution. However, in view of the slow variability of the system matrices A_k at increasing k, we propose to evaluate selectively the eigenpairs, whenever the PCG solution of a generic linear system $A_k\boldsymbol{x}_k = \boldsymbol{b}$ takes more than a fixed number of iterations ($it_k \geq it_{\text{prec}}$). In this case, except for $k = 1$, it is effective to use as initial DACG guess the previously computed eigenvectors. The final algorithm is reported in Algorithm 2.

Algorithm 2 PCG with spectral DACG preconditioner

- INPUT: it_{prec}, it_{chol}, p, τ_{DACG}.
- Set chol_switch = TRUE; switch = TRUE;
- FOR $k = 1$ TO n_sys
  - IF chol_switch THEN
    compute $P_0 = IC(A_k)$; set chol_switch = FALSE;
  - IF switch THEN
    1. Compute the p leftmost eigenpairs by the DACG procedure with preconditioner P_0 and accuracy τ_{DACG}.
    2. Form matrices V_p, Λ_p.
    3. Solve the k-th linear system by PCG preconditioned by $P_0 + V_p\Lambda_p^{-1}V_p^T$.
    4. switch = FALSE.
  - IF $it_k > it_{\text{prec}}$ switch = TRUE
  - IF $it_k > it_{\text{chol}}$ chol_switch = TRUE

  END FOR

4.3. Eigenpairs of P_0A obtained by Lanczos–PCG

Computation of matrices T_m and W_m is carried out during the PCG process and adds negligible computational costs due to the saving of the PCG residual vectors. The main computational burden in this strategy is given by the matrix–matrix product $V_m = W_mQ_p$ implemented via BLAS-3 subroutines, with a consequent optimal use of memory accesses. Due to

the slow convergence of the Lanczos process to the smallest eigenvalues, and also for memory reasons, it is convenient to recover a relatively small number of eigenpairs (independently of the size m of V_m, which nonetheless should be taken sufficiently large to ensure the completeness of the calculated leftmost eigenspectrum). In the Lanczos process we use only the $p = \{10, 20\}$ smallest eigenvalues and corresponding eigenvectors thus obtaining a $n \times p$ matrix V_p and a $p \times p$ diagonal matrix Λ_p. The final algorithm is reported in Algorithm 3.

Algorithm 3 PCG with spectral Lanczos preconditioner

- INPUT: it_{prec}, it_{chol}, $m_{\max}$, p.
- Set chol_switch = TRUE; switch = TRUE;
- FOR $k = 1$ TO n_sys
  - IF chol_switch THEN
    compute $P_0 = IC(A_k)$; set chol_switch = FALSE;
  - IF switch THEN
    1. Solve the k-th linear system by the PCG method preconditioned by P_0.
    2. Construct the tridiagonal Lanczos matrix T_m, with $m = \min\{m_{\max}, it_k\}$.
    3. Extract from T_m and W_m the p smallest eigenpairs and form matrices V_p, Λ_p.
    4. switch = FALSE.
  - ELSE
    1. Solve the k-th linear system by PCG preconditioned by $P_0 + V_p \Lambda_p^{-1} V_p^T$.
  - IF $it_k > it_{\text{prec}}$ switch = TRUE
  - IF $it_k > it_{\text{chol}}$ chol_switch = TRUE

END FOR

5. Numerical results

In this section we illustrate the behavior of the spectral preconditioner on a sequence of linear systems arising in the discretization of (1). The code is written in Fortran 90. All the experiments were run on a 2 x Intel Xeon CPU E5645 at 2.40 GHz (six core) and with 4GB RAM for each core. Times are expressed in seconds. The stopping criterion for the linear solver is independent of the preconditioner used and it is based on the relative residual:

$$\frac{\|A_k \boldsymbol{x}_k - \boldsymbol{b}\|}{\|\boldsymbol{b}\|} < \varepsilon = 10^{-11}.$$

We solve the piecewise-constant source test case (shown in the left column of Fig. 1) for $\beta = 5$ because it is the case that presents the strongest time-variability of the transport density among the considered test cases. All the simulations employ a mesh $\mathcal{T}_h$ of 412417 nodes and $n_{nz} = 1647617$ nonzero elements. Efficient simulations that ensure stability of explicit Euler are obtained using an initial time step size $\Delta t^{(0)} = 10^{-3}$ and then increasing $\Delta t^{(k)}$ by a factor 1.05 at each time step up to a maximum value of $\Delta t^{(k)} = 10^{-1}$. This leads to a sequence of almost 4000 linear systems of type (6) to reach equilibrium at the chosen tolerance τ.

For this type of problems, homogeneous Neumann boundary conditions are natural but lead to a singular system matrix, with a non trivial kernel containing the constant vectors $\boldsymbol{c}$. However, the use of unnatural Dirichlet conditions often yielded linear solver failures, due most probably to extreme matrix ill-conditioning. Hence, we employ homogeneous Neumann condition and guarantee the well-posedness of the resulting linear systems (and of the PCG process) by projecting the right hand side onto the range of A_k, $R(A_k)$, as follows:

$$\tilde{\boldsymbol{b}} = \boldsymbol{b} - \frac{\boldsymbol{c}^T \boldsymbol{b}}{\|\boldsymbol{c}\|^2} \boldsymbol{c}$$

(see [19]). Note that such projection simply corrects quadrature errors in the construction of $\boldsymbol{b}$, since f is assumed to have zero-mean.

Denoting with $\lambda_1 = 0 < \lambda_2 < \cdots \lambda_n$ the eigenvalues of A_k in this case, the effective spectral condition number of matrix A_k is $\kappa(A_k) = \lambda_n/\lambda_2$ since the zero eigenvalue does not affect convergence of the PCG iteration after the projection of $\boldsymbol{b}$ on $R(A_k)$. On the other hand, in the case of Dirichlet boundary conditions, all the eigenvalues of A_k^D change and the zero eigenvalue occurring in the Neumann case is shifted to a positive value close to zero (near 10^{-6} for the example shown in Fig. 2), yielding a spectral condition number much greater than in the Neumann case: $\kappa(A_k^D) = \lambda_n^D/\lambda_1^D \gg \kappa(A_k)$.

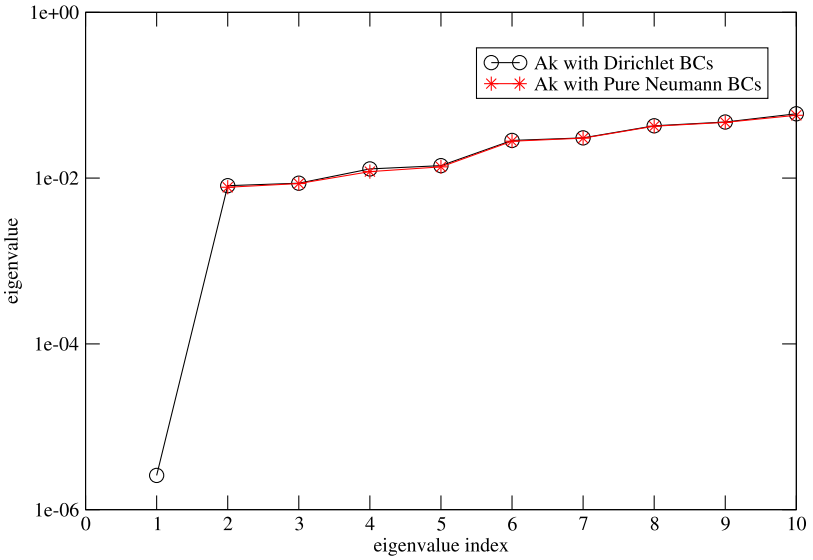

Fig. 2. Effect of imposing Dirichlet boundary conditions on the smallest eigenvalues of $P_k A_k$, for system $k = 200$ in the sequence (no spectral acceleration).

Table 1
Influence of the DACG tolerance on the performance of the PCG with spectral preconditioner.

τ_{DACG}	ITER	T_{eig}	T_{prec}	T_{PCG}	T_{tot}
–	20 646	0.00	187.9	1687.8	1875.7
0.1	9 907	326.9	117.9	1002.1	1446.2
0.3	10 006	198.5	117.2	1011.5	1327.8
0.5	10 055	150.0	117.2	1017.7	1284.9

All the linear systems have been symmetrically scaled with the diagonal of A_k in order to reduce their initial condition number, namely, defining $D = \mathrm{diag}(a_{11}, \ldots, a_{nn})$:

$$\text{solve } D^{-1/2} A_k D^{-1/2} \boldsymbol{y}_k = D^{-1/2} \tilde{\boldsymbol{b}},$$
$$\text{compute } \boldsymbol{x}_k = D^{1/2} \boldsymbol{y}_k.$$

The efficacy of the proposed algorithms is verified by looking at the overall iteration count of the PCG solver and the CPU times for solving the entire linear system sequence. We test different numbers of eigenvectors p used to build the low rank correction to the initial preconditioner for both the Lanczos (LAN(p)) and DACG (DACG(p)) eigensolution algorithms. We report CPU timings accounting for the computation of the preconditioner (T_{prec}), of the approximated eigenvectors (T_{eig}), the PCG solver (T_{PCG}), and the total CPU time (T_{tot}).

5.1. Influence of eigenvector accuracy in DACG preprocessing

We first perform a preliminary study on the influence of accuracy of eigenvector computation in the PCG acceleration provided by the resulting spectral preconditioner. To this end we considered the first 200 linear systems and use three different tolerances for the relative eigenresidual test (8): $\tau_{DACG} \in \{0.1, 0.3, 0.5\}$. Other parameters were: $it_{eig} = 60$, $it_{chol} = 60$, $p = 20$. The results are shown in Table 1. As a benchmark, we also solved the first 200 systems by the PCG method preconditioned by an IC factorization computed selectively (with $it_{chol} = 100$), with no spectral update (see first row of Table 1). The results show that high accuracy in eigenpair computation is not needed to reduce the number of PCG iterations. With a very low accuracy ($\tau_{DACG} = 0.5$) (last row of Table 1) the number of iterations is halved and the CPU time reduced of a factor 1.5 with respect to the IC preconditioner without spectral correction (first row of Table 1).

5.2. Smallest eigenvalues of $P_0 A_k$

In Fig. 3 we plot the computed eigenvalues of $P_0 A_k$, where P_0 is the IC preconditioner of A_k, by the spectral Lanczos–PCG procedure. In particular we plot the 10 smallest eigenpairs of preconditioned systems #1, 21, 40, ..., 181. From the figure we notice that the "stars" are vertically clustered, showing that the smallest eigenvalues of the preconditioned matrices only slightly change among systems at close simulation times.

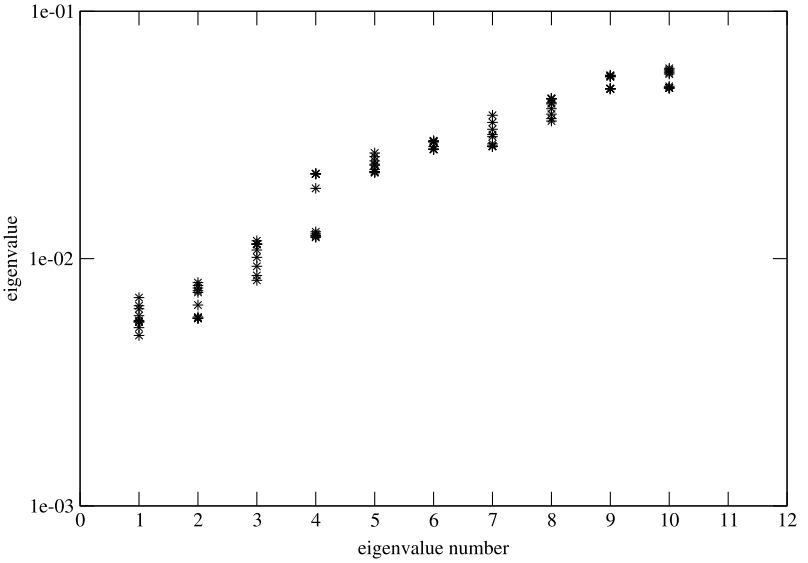

Fig. 3. 10 smallest eigenvalues of $P_k A_k$ for $k = 20j + 1, j = 0, \ldots, 9$.

Table 2
Timings and iterations related to the whole sequence of linear systems corresponding to two different values of β using PCG with different preconditioners and parameters.

Prec. (p)	it_{chol}	it_{eig}	ITER	T_{prec}	T_{eig}	T_{PCG}	T_{tot}
$\beta = 1.5$							
DACG(10)	25	25	170 111	6 262.3	1869.0	14 276.7	22 758.7
LAN(10)	25	25	219 515	11 753.4	2077.5	17 604.6	31 800.1
IC	25	–	261 964	16 909.1	0.0	19 348.9	36 644.2
DACG(10)	30	30	190 075	3 453.1	1228.5	15 949.9	20 977.9
LAN(10)	30	30	230 242	8 080.1	1437.3	18 699.8	28 572.5
IC(10)	30	–	272 779	13 043.8	0.0	19 981.0	33 392.3
DACG(10)	40	40	232 004	1 660.2	811.1	19 329.4	22 146.1
LAN(10)	40	40	259 569	3 591.5	688.1	21 420.5	26 049.4
IC(10)	40	–	299 100	9 040.8	0.0	21 858.1	31 254.6
$\beta = 5$							
DACG(10)	60	60	220 110	1 305.8	446.3	18 853.5	20 785.3
LAN(10)	60	60	208 481	910.2	166.2	17 834.9	19 092.0
IC	60	–	263 477	9 532.6	0.0	19 900.5	29 632.0
IC	–	–	257 219	13 041.1	0.0	19 415.9	32 666.6

5.3. Results of the simulations

We report in Table 2 the results of the complete simulation corresponding to different values of the parameter β i.e. the cumulative number of PCG iterations and CPU times in solving the sequence of almost 4000 linear systems needed to reach the steady-state. In addition to the previously described parameters we used as the maximum size of the Lanczos subspace $m_{\max} = 80$, which experimentally revealed the optimal value.

Inspection of Table 2 reveals that both DACG and Lanczos acceleration provide an improvement in the number of iterations and total CPU time. For the easier $\beta = 1.5$ case we tried various values for parameters it_{chol} and it_{eig}. For the challenging case $\beta = 5$, the optimal spectral preconditioner turns out to be the one based on the Lanczos approach, which provides a gain of more than 40% CPU time with respect to using the Cholesky preconditioner computed at each time step.

5.4. Further analysis on a portion of the simulation

To better inspect the optimal choice of the parameters we analyzed the first 800 time steps, after which the solution is near its steady-state. From Table 3 we notice that the proposed low-rank update of preconditioners is effective in both variants, providing an important reduction of the number of iterations as well of the CPU time. On the average, our spectral preconditioners provide a halving of the total CPU time and a 30% − 40% reduction in the number of iterations. Using $p = 10$

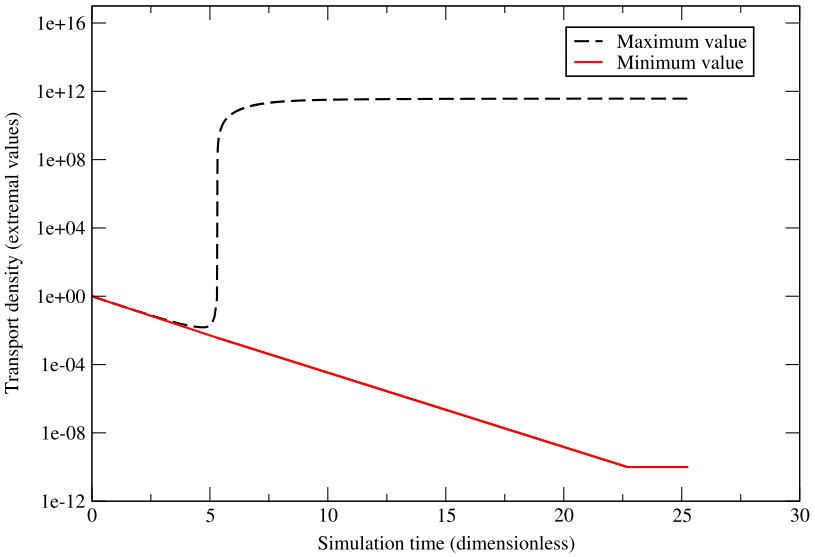

Fig. 4. Maximum and minimum value of μ_h during the simulation.

Table 3
Timings and iterations related to the first 800 systems in the sequence corresponding to $\beta = 5$. PCG with different preconditioners and parameters.

Prec. (p)	it_{chol}	it_{eig}	ITER	T_{eig}	T_{prec}	T_{PCG}	T_{tot}
IC	–	–	64 248	0.0	2841.2	5432.5	8273.7
IC	100	–	74 511	0.0	447.4	6062.9	6510.3
LAN(10)	–	60	41 148	29.6	2814.2	3372.1	6215.9
LAN(10)	50	70	44 765	30.9	1767.2	3746.4	5544.5
LAN(10)	60	60	44 041	196.0	572.7	3606.9	4375.6
LAN(20)	60	60	41 738	190.4	459.2	3775.8	4425.4
DACG(10)	60	60	45 502	185.4	516.0	3811.6	4512.9
DACG(20)	60	60	42 050	263.5	272.2	3922.0	4457.7

or $p = 20$ eigenvectors produces only slight variations in the number of iterations/CPU time. Hence, the choice $p = 10$ seems to be preferred in terms of memory storage.

Surprisingly, the DACG variant, although affected by a CPU-intensive offline (outside the PCG algorithm) phase for the eigenvector approximation, reveals as effective as the Lanczos variant. This is mainly due to the fact that after the initial assessment of the leftmost eigenpairs, the subsequent computations are very cheap since the previously computed eigenvectors are very good initial guesses for the next systems. However, we may expect a different behavior of the two techniques in cases of higher variations of the matrices involved. In this case, the DACG preprocessing time will increase as opposite to the Lanczos technique. Moreover, the Lanczos approach can be accelerated by employing a method similar to that described in [20]. This is a topic for a future work.

5.5. Handling high density variations

As clear from Fig. 4 there is a portion of the simulation in which the largest value of the density vector abruptly increases and rapidly reaches its maximum value.

Since, as anticipated in Section 2, $\lambda_{\min}(A) \leq C_1 h^2 \mu_{\min}$ and $\lambda_{\max}(A) \geq C_2 \mu_{\max}$, the sudden increase of $\mu_{\max}$ produces a high variation in the condition number of the matrices in the sequence. Hence, in this time interval, the spectral properties of the system matrices change significantly and PCG is not able to take advantage of the spectral information provided by the previous systems.

To this aim we forced the code to recompute the Cholesky preconditioner whenever the following test on μ_h is satisfied:

$$\frac{\|\mu_h^{(k+1)} - \mu_h^{(k)}\|_{L^2(\Omega)}}{\Delta t^k \|\mu_h^{(k)}\|_{L^2(\Omega)}} > \delta \tag{13}$$

with $\delta = 100$ in the experiments. Also the time step is dynamically reduced in this portion of the simulation to correctly capture the dynamics.

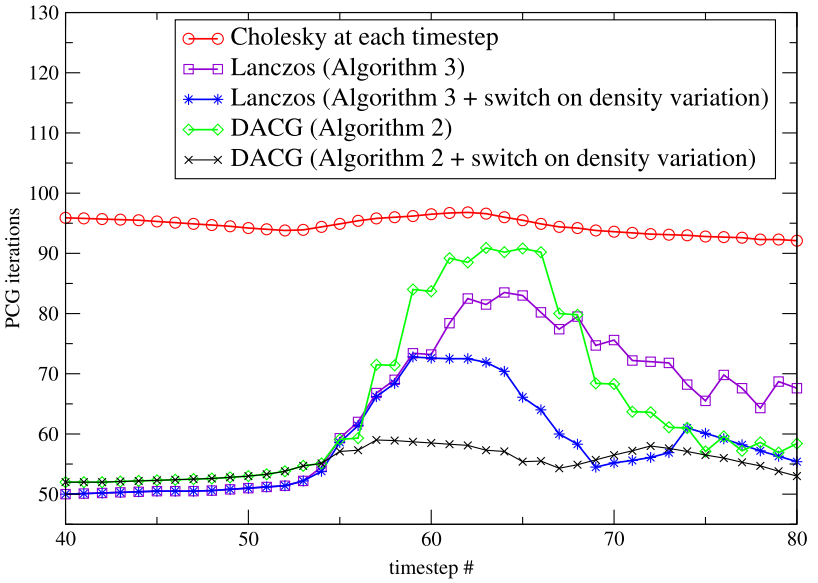

Fig. 5. PCG iterations in solving systems #40 to #80 with various spectral and switching strategies.

To appreciate the benefit of this modification we report in Fig. 5 the number of iterations (averaged over the last 5 linear systems) needed by the PCG solver for systems #40 to #80 (corresponding to time interval: [4, 5.43] where the μ_h variation is more pronounced). We display in the figure results with the Cholesky preconditioner only and Lanczos(10) and DACG(10) spectral acceleration with and without test (13).

The new switching strategy is clearly effective and particularly so in combination with the DACG spectral acceleration.

6. Conclusions

We have proposed a class of spectral preconditioners with the aim to accelerate the PCG solution of a sequence of very ill-conditioned linear systems arising from the discretization of a continuous branched transport model. Using the fact that the matrices involved vary only mildly as simulation time progresses, we use eigeninformation obtained at a given time step to accelerate subsequent linear system solutions. Numerical results on a realistic test case reveal that the CPU time required by the eigenanalysis is almost negligible, being in our experiments less than 5% of the overall CPU time when $p = 10$. Moreover, the proposed spectral preconditioners are able to consistently reduce the number of iterations and almost halve the CPU time of the total simulation time with respect to the case in which the IC preconditioner is used at each time step.

Acknowledgments

This research was partially supported by the University of Padua project # CPDA155834/15: *Stable and efficient discretizations of the mechanics of faults* and by the GNCS project *Numerical Methods for large constrained optimization problems and applications*.

References

[1] E. Facca, F. Cardin, M. Putti, A continuous model of slime mold dynamics, SIAM J. Appl. Math. (2017). In print. Accessible at https://arxiv.org/abs/1610.06325.
[2] E. Facca, S. Daneri, F. Cardin, M. Putti, Numerical solution of Monge-Kantorovich equations via a dynamic formulation, SIAM J. Sci. Comput. (2017). submitted for publication. Accessible at https://arxiv.org/abs/1709.06765.
[3] L.C. Evans, W. Gangbo, Differential equations methods for the Monge-Kantorovich mass transfer problem, Mem. AMS 137 (653) (1999).
[4] Q. Xia, Optimal paths related to transport problems, Bell Syst. Tech. J. 5 (2) (2003) 251–279.
[5] F. Santambrogio, Optimal Transport for Applied Mathematicians, in: Calculus of Variations, PDEs, and Modeling, vol. 87, Birkhäuser, Cham, 2015.
[6] Q. Xia, Motivations, ideas and applications of ramified optimal transportation, ESAIM Math. Model. Numer. Anal. 49 (6) (2015) 1791–1832.
[7] E.N. Gilbert, Minimum cost communication networks, Vol. 46, 1967, pp. 2209–2227.
[8] E. Oudet, F. Santambrogio, A Modica-Mortola approximation for branched transport and applications, Arch. Ration. Mech. Anal. 201 (1) (2011) 115–142.
[9] B. Carpentieri, I.S. Duff, L. Giraud, A class of spectral two-level preconditioners, SIAM J. Sci. Comput. 25 (2) (2003) 749–765 (electronic).
[10] I.S. Duff, L. Giraud, J. Langou, E. Martin, Using spectral low rank preconditioners for large electromagnetic calculations, Internat. J. Numer. Methods Engrg. 62 (2005) 416–434.
[11] L. Giraud, S. Gratton, E. Martin, Incremental spectral preconditioners for sequences of linear systems, Appl. Numer. Math. 57 (11–12) (2007) 1164–1180. Numerical Algorithms, Parallelism and Applications (2).

[12] J. Mas, J. Cerdán, N. Malla, J. Marín, Application of the Jacobi-Davidson method for spectral low-rank preconditioning in computational electromagnetics problems, SeMA 67 (2015) 39–50.
[13] A. Quarteroni, A. Valli, Numerical Approximation of Partial Differential Equations, in: Springer Series in Computational Mathematics, vol. 23, Springer-Verlag, Berlin, 1994.
[14] L. Bergamaschi, A. Martínez, Two-stage spectral preconditioners for iterative eigensolvers, Numer. Linear Algebra Appl. 24 (3) (2017) 1–14.
[15] L. Bergamaschi, G. Gambolati, G. Pini, Asymptotic convergence of conjugate gradient methods for the partial symmetric eigenproblem, Numer. Linear Algebra Appl. 4 (2) (1997) 69–84.
[16] L. Bergamaschi, M. Putti, Numerical comparison of iterative eigensolvers for large sparse symmetric matrices, Comput. Methods Appl. Mech. Engrg. 191 (45) (2002) 5233–5247.
[17] L. Bergamaschi, A. Martínez, G. Pini, Parallel Rayleigh Quotient optimization with FSAI-based preconditioning, J. Appl. Math. 2012 (2012) 14. Article ID 872901.
[18] A. Martínez, Tuned preconditioners for the eigensolution of large spd matrices arising in engineering problems, Numer. Linear Algebra Appl. 23 (3) (2016) 427–443.
[19] E.F. Kaasschieter, Preconditioned conjugate gradients for solving singular systems, J. Comput. Appl. Math. 24 (1–2) (1988) 265–275.
[20] A. Stathopoulos, K. Orginos, Computing and deflating eigenvalues while solving multiple right-hand side linear systems with an application to quantum chromodynamics, SIAM J. Sci. Comput. 32 (1) (2010) 439–462.