Multilevel preconditioners for solving eigenvalue problems occuring in the design of resonant cavities

Multilevel preconditioners for solving eigenvalue problems occuring in the design of resonant cavities

Peter Arbenz * and Roman Geus

Institute of Scientific Computing, Swiss Federal Institute of Technology, CH-8092 Zurich, Switzerland

Abstract

We investigate eigensolvers for computing a few of the smallest eigenvalues of a generalized eigenvalue problem resulting from the finite element discretization of the time independent Maxwell equation. Various multilevel preconditioners are employed to improve the convergence and memory consumption of the Jacobi-Davidson algorithm and of the locally optimal block preconditioned conjugate gradient (LOBPCG) method. We present numerical results of very large eigenvalue problems originating from the design of resonant cavities of particle accelerators.

Key words: Maxwell equation, generalized eigenvalue problem, Jacobi-Davidson, LOBPCG, smoothed aggregation AMG preconditioner
PACS: 02.60.Dc, 02.70.Dh, 42.60.Da

1 Introduction

Many applications in electromagnetics require the computation of some of the eigenpairs of the curl-curl operator,

$$\mathbf{curl}\,\mu_r^{-1}\mathbf{curl}\,\mathbf{e}(\mathbf{x}) - k_0^2\,\varepsilon_r\,\mathbf{e}(\mathbf{x}) = \mathbf{0}, \qquad \operatorname{div}\mathbf{e}(\mathbf{x}) = 0, \qquad \mathbf{x}\in\Omega, \tag{1.1}$$

in the bounded three-dimensional domain Ω with homogeneous boundary conditions $\mathbf{e}\times\mathbf{n} = \mathbf{0}$. Here, ε_r and μ_r are the relative permittivity and permeability, respectively. Equations (1.1) are obtained from the Maxwell equations after separation of the time and space variables and after elimination of the

* Corresponding author.
Email address: arbenz@inf.ethz.ch (Peter Arbenz).

magnetic field intensity. While ε_r and μ_r are complex numbers in problems from waveguide or laser design, in simulation of accelerator cavities the materials can be assumed to be loss-free, thus admitting real ε_r and μ_r, whence all eigenvalues are real. In fact, we will assume $\varepsilon_r = \mu_r = 1$. Thus, the discretization of (1.1) by finite elements leads to a real symmetric generalized matrix eigenvalue problem

$$A\mathbf{x} = \lambda M\mathbf{x}, \qquad C^T\mathbf{x} = \mathbf{0}, \tag{1.2}$$

where A is positive semidefinite and M is positive definite. In this paper we consider eigensolvers for computing a few, i.e., five to ten of the smallest eigenvalues and corresponding eigenvectors of (1.2) as efficiently as possible with regard to execution time and consumption of memory space. In earlier studies [1,2] we found the Jacobi-Davidson algorithm [24] and the locally optimal block preconditioned conjugate gradient (LOBPCG) method [17] to be the most effective solvers for this task. We now have incorporated a sophisticated multilevel preconditioner that is the combination of a hierarchical basis [3] and a smoothed aggregation AMG preconditioner [26,22]. We review eigensolvers and preconditioners and tell how we employ them in sections 3 to 4. In section 5 we report on experiments that we conducted by means of problems originating in the design of the RF cavity of the 590 MeV ring cyclotron installed at the Paul Scherrer Institute (PSI) in Villigen, Switzerland. These experiments indicate that the implemented multilevel preconditioner is indeed optimal in that the number of iteration steps until convergence only slightly depends on the problem size.

2 The application: the cavity eigenvalue problem

The finite element discretization is based on the weak formulation of (1.1) as suggested in [16].

Find $(\lambda_h, \mathbf{e}_h, p_h) \in \mathbb{R} \times N_h \times L_h$ *such that* $\mathbf{e}_h \neq \mathbf{0}$ *and*

$$\begin{aligned} &\text{(a)}\ (\mathbf{curl}\,\mathbf{e}_h, \mathbf{curl}\,\boldsymbol{\Psi}_h) + (\mathbf{grad}\,p_h, \boldsymbol{\Psi}_h) = \lambda_h(\mathbf{e}_h, \boldsymbol{\Psi}_h), && \forall \boldsymbol{\Psi}_h \in N_h \\ &\text{(b)}\ (\mathbf{e}_h, \mathbf{grad}\,q_h) = 0, && \forall q_h \in L_h \end{aligned} \tag{2.1}$$

where $N_h \subset H_0(\mathbf{curl}; \Omega) = \{\mathbf{v} \in L^2(\Omega)^3 \mid \mathbf{curl}\,\mathbf{v} \in L^2(\Omega)^3, \mathbf{v} \times \mathbf{n} = \mathbf{0} \text{ on } \partial\Omega\}$ and $L_h \subset H_0^1(\Omega)$. In order to avoid spurious modes we choose the subspaces N_h and L_h, respectively, to be the Nédélec (or edge) elements [13,23] and the Lagrange (or node-based) finite elements [5] both of matching degree, in our implementation of degree 2. Let $\{\boldsymbol{\Phi}_i\}_{i=1}^n$ be a basis of N_h and $\{\varphi_l\}_{l=1}^m$ be a

basis of L_h. Then (2.1) defines the matrix eigenvalue problem

$$\begin{bmatrix} A & C \\ C^T & O \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \lambda \begin{bmatrix} M & O \\ O & O \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}, \tag{2.2}$$

respectively, where A and M are n-by-n and C is n-by-m with elements

$$a_{i,j} = (\mathbf{curl}\, \mathbf{\Phi}_i, \mathbf{curl}\, \mathbf{\Phi}_j), \quad m_{i,j} = (\mathbf{\Phi}_i, \mathbf{\Phi}_j), \quad c_{i,l} = (\mathbf{\Phi}_i, \mathbf{grad}\, \varphi_l).$$

(2.2) can equivalently be written in the form (1.2). The reason for approximating the electric field $\mathbf{e}$ by Nédélec elements and the Lagrange multipliers by Lagrange finite elements is that [13, §5.3]

$$\mathbf{grad}\, L_h = \{\mathbf{v}_h \in N_h \mid \mathbf{curl}\, \mathbf{v}_h = \mathbf{0}\}. \tag{2.3}$$

By (2.1)(b), $\mathbf{e}_h$ is in the orthogonal complement of $\mathbf{grad}\, L_h$. On this subspace A in (2.2) is positive definite. Notice that $\mathbf{e}_h$ is divergence-free only in a discrete sense. Because of (2.3) we can write $\mathbf{grad}\, \varphi_l = \sum_{j=1}^n \mathbf{\Phi}_j y_{jl}$, whence

$$(\mathbf{\Phi}_i, \mathbf{grad}\, \varphi_l) = \sum_{j=1}^{n} (\mathbf{\Phi}_i, \mathbf{\Phi}_j) y_{jk}, \qquad \text{or} \qquad C = MY, \tag{2.4}$$

where $Y = ((y_{jk})) \in \mathbb{R}^{n\times m}$. Similarly one obtains

$$H := C^T Y = Y^T M Y, \qquad h_{kl} = (\mathbf{grad}\, \varphi_k, \mathbf{grad}\, \varphi_l). \tag{2.5}$$

Notice that Y is very sparse. Its rows contain at most two nonzero entries that are 1 or -1. H is the system matrix that is obtained when solving the Poisson equation with the Lagrange finite elements L_h. From (2.4) we see that $C^T\mathbf{x} = \mathbf{0}$ is equivalent to requiring $\mathbf{x}$ to be M-orthogonal to the eigenspace $\mathcal{N}(A) = \mathcal{R}(Y)$ corresponding to the eigenvalue 0. Thus, the solutions of (2.2) are precisely the eigenpairs of

$$A|_{\mathcal{N}(C^T)}\mathbf{x} = \lambda M|_{\mathcal{N}(C^T)}\mathbf{x}. \tag{2.6}$$

We enforce the constraint by applying the M-orthogonal projector $P_{\mathcal{N}(C^T)} = I - YH^{-1}C^T$ onto $\mathcal{N}(C^T)$ whenever a vector is not in this subspace.

It may seem possible to simply neglect the restrictions in (2.6). This is in principle feasible if the eigenvalue problem is solved by the Lanczos algorithm with the shift-and-invert spectral transformation and the systems $(A - \sigma M)\mathbf{x} = \mathbf{y}$ are solved directly, provided that the initial subspace satisfies the constraint. If a preconditioned iterative solver is applied the approximate solutions do not satisfy the constraints anymore.

3 Solving the matrix eigenvalue problem

Factorization-free methods are the most promising for effectively solving very large eigenvalue problems

$$A\mathbf{x} = \lambda M\mathbf{x}. \tag{3.1}$$

The factorization of A or M or a linear combination of them which is needed if a spectral transformation like shift-and-invert is applied requires far too much memory. If the shift-and-invert spectral transformation in a Lanczos type method is solved iteratively then high accuracy is required to establish the three-term recurrence. This is very time-consuming even if the conjugate gradient method is applied with a good preconditioner. In most of our experiments the Jacobi-Davidson algorithm was much more effective than the implicitly restarted Lanczos algorithm as implemented in ARPACK [18] for solving the cavity problem and other eigenvalue problems [1,2].

In this paper we investigate the two probably most powerful factorization-free algorithms for solving (3.1), the *symmetric Jacobi-Davidson algorithm (JDSYM)* and the *locally optimal block PCG method (LOBPCG)*. In this section we give details on our actual implementation.

3.1 The symmetric Jacobi-Davidson algorithm

The Jacobi-Davidson has been introduced by Sleijpen and van der Vorst [24]. There are variants for all types of eigenvalue problems. Here, we use a variant adapted to the generalized symmetric eigenvalue problem (1.2) as described in [1,12].

The Rayleigh quotient corresponding to the matrix pencil (A, M) of (1.2) is defined as

$$\rho(\mathbf{x}) = \frac{\mathbf{x}^T A\mathbf{x}}{\mathbf{x}^T M\mathbf{x}}, \qquad C^T\mathbf{x} = \mathbf{0}.$$

Let us assume that in an intermediate step of the algorithm we have available a search space $\mathcal{R}(V_k) \subset \mathcal{N}(C^T)$ with $V_k = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$ and that $(\tilde{\rho}, \tilde{\mathbf{q}})$ with $\tilde{\rho} = \rho(\tilde{\mathbf{q}})$ is the Ritz pair in $\mathcal{R}(V_k)$ that best approximates the searched eigenpair. In general, $\mathcal{R}(V_k)$ is not a Krylov subspace. As the eigenpairs of $A\mathbf{x} = \lambda M\mathbf{x}$ are the stationary values of the Rayleigh quotient, it is straightforward to apply Newton's method to

$$\mathbf{grad}\,\rho(\mathbf{x}) = \frac{2}{\mathbf{x}^T M\mathbf{x}}\,(A - \rho(\mathbf{x})M)\,\mathbf{x} = \mathbf{0}, \tag{3.2}$$

to improve the approximation $\tilde{\mathbf{q}}$. The Newton correction $\mathbf{t}$ at $\tilde{\mathbf{q}}$ is determined

(1) *Choose* $\mathbf{v}_1$ *with* $\|\mathbf{v}_1\|_M = 1$. *Choose target* τ. *Set* $Q := [\,]$. $k := 1$.
(2) $\mathbf{v}_1 = (I - YH^{-1}C^T)\mathbf{v}_1$.
(3) $V_1 := [\mathbf{v}_1]; \quad \tilde{\mathbf{q}} := \mathbf{v}_1; \quad \tilde{\rho} := \rho(\tilde{\mathbf{q}}); \quad \mathbf{r} := A\tilde{\mathbf{q}} - \tilde{\rho}\, M\tilde{\mathbf{q}}; \quad \widetilde{Q} = [\tilde{\mathbf{q}}]$.
(4) while $rank(Q) < p$ do
(5) *Choose shift:* either $\eta_k := \tau$ or $\eta_k := \tilde{\rho}$.
(6) *Solve approximately for* $\mathbf{t}$ ***(Correction equation)***
$\{(I - M\widetilde{Q}\widetilde{Q}^T)(A - \eta_k M)(I - \widetilde{Q}\widetilde{Q}^T M)\mathbf{t} = -\mathbf{r}, \quad \widetilde{Q}^T M\mathbf{t} = 0.\}$
(7) $\mathbf{t} = (I - YH^{-1}C^T)\mathbf{t}$.
(8) $\mathbf{v}_{k+1} := (I - V_k V_k^T M)\mathbf{t}; \quad \mathbf{v}_{k+1} := \mathbf{v}_{k+1}/\|\mathbf{v}_{k+1}\|_M$.
(9) $k := k + 1$.
(10) $V_k := [V_{k-1}, \mathbf{v}_k]; \quad H_k = V_k^T A V_k$. ***(Subspace expansion)***
(11) *Compute* $H_k S_k = S_k \Lambda_k$ ***(Spectral decomposition)***
where $S_k^{-1} = S_k^T$ *and* $\Lambda_k = diag(\lambda_1^{(k)}, .., \lambda_k^{(k)})$
with $|\lambda_l^{(k)} - \tau| \le |\lambda_{l+1}^{(k)} - \tau|, \quad l < k$.
(12) repeat ***(Convergence test)***
(13) $\tilde{\rho} := \lambda_1^{(k)}; \quad \tilde{\mathbf{q}} := V_k \mathbf{s}_1; \quad \mathbf{r} := A\tilde{\mathbf{q}} - \tilde{\rho}\, M\tilde{\mathbf{q}}$.
(14) *found* $:= \|\mathbf{r}\|_2 < \varepsilon$ and $k > 1$;
(15) if *found* then
(16) $Q := [Q, \tilde{\mathbf{q}}]; \quad V_{k-1} := V_k[\mathbf{s}_2, \ldots, \mathbf{s}_k]$;
(17) $\Lambda_{k-1} := diag(\lambda_2^{(k)}, \ldots, \lambda_k^{(k)}); \quad S_{k-1} := I_{k-1}; \quad k := k - 1$;
(18) end if
(19) until *not(found)*
(20) $\tilde{Q} := [Q, \tilde{\mathbf{q}}]$;
(21) if $k = j_{max}$ then ***(Restart)***
(22) $V_{j_{min}} := V_k[\mathbf{s}_1, \ldots, \mathbf{s}_{j_{min}}]; \quad k := j_{min}$;
(23) end if
(24) end while

Algorithm 1. JDSYM: The symmetric Jacobi-Davidson algorithm

by the *correction equation*

$$\begin{aligned} &(I - M\widetilde{Q}\widetilde{Q}^T)(A - \tilde{\rho}M)(I - \widetilde{Q}\widetilde{Q}^T M)\mathbf{t} = -\mathbf{r}, \\ &\widetilde{Q}^T M\mathbf{t} = 0, \qquad C^T\mathbf{t} = \mathbf{0}, \qquad \widetilde{Q} = [Q, \tilde{\mathbf{q}}] \end{aligned} \tag{3.3}$$

where $\mathbf{r} = A\tilde{\mathbf{q}} - \tilde{\rho}\, M\tilde{\mathbf{q}}$ is called the residual at $\tilde{\mathbf{q}}$ and Q contains the already computed M-orthonormalized eigenvectors as its columns. Instead of updating $\tilde{\mathbf{q}}$ by $\tilde{\mathbf{q}} + \mathbf{t}$ as in Newton's method the correction $\mathbf{t}$ is made orthogonal to $\mathbf{v}_1, \ldots, \mathbf{v}_k$ and, after normalization, becomes $\mathbf{v}_{k+1}$. V_k is expanded by $\mathbf{v}_{k+1}$ to become V_{k+1}. This procedure is related to Rayleigh quotient iteration [20], whence a local cubic converge rate can be deduced [24].

However, with large problems, the Rayleigh quotient iteration is not feasible, as it requires the factorization of a matrix in each iteration step. In fact,

away from convergence, it is much more effective to solve the correction equation to very low accuracy with $\tilde{\rho}$ replaced by a fixed target value τ. Close to convergence, higher (but not very high) accuracy is needed to have a decent convergence rate. Therefore it is natural to solve the correction equation (3.3) iteratively by a Krylov subspace method. We found that the divergence-free condition $C^T\mathbf{t} = \mathbf{0}$ needs not be enforced during the iteration [2]. Because of the way we derived this finding we called the method simplified augmented system (SAUG) approach. Only at the end of the iteration the approximate solution is projected onto $\mathcal{N}(C^T)$.

For the Krylov subspace method to be efficient a preconditioner is a prerequisite. Following Fokkema et al. [10] we use preconditioners of the form

$$(I - M\widetilde{Q}\widetilde{Q}^T)K(I - \widetilde{Q}\widetilde{Q}^T M), \tag{3.4}$$

where K is a symmetric preconditioner of $A - \tilde{\rho}M$. For efficiency reasons we compute K only once for a fixed shift σ such that $K \approx A - \sigma M$. Often we choose $\sigma = \tau$, the target value. In each preconditioning step an equation of the form

$$(I - M\widetilde{Q}\widetilde{Q}^T)K\mathbf{c} = \mathbf{b} \quad \text{and} \quad \widetilde{Q}^T M\mathbf{c} = \mathbf{0} \tag{3.5}$$

has to be solved. The solution $\mathbf{c}$ is [12]

$$\mathbf{c} = (I - K^{-1}M\widetilde{Q}(\widetilde{Q}^T MK^{-1}M\widetilde{Q})^{-1}\widetilde{Q}^T M)K^{-1}\mathbf{b}. \tag{3.6}$$

In summary, the Krylov subspace method is invoked with the following arguments.

$$\begin{array}{ll} \text{system matrix:} & (I - M\widetilde{Q}\widetilde{Q}^T)(A - \tilde{\rho}M) \\ \text{preconditioner:} & (I - K^{-1}M\widetilde{Q}(\widetilde{Q}^T MK^{-T}M\widetilde{Q})^{-1}\widetilde{Q}^T M)K^{-1} \\ \text{right hand side:} & -(I - M\widetilde{Q}\widetilde{Q}^T)\mathbf{r} \\ \text{initial vector:} & \mathbf{0} \end{array} \tag{3.7}$$

Both the system matrix and the preconditioner are symmetric. However, because of the dynamic shift $\tilde{\rho}$ they can become indefinite. For this reason, the QMRS iterative solver [11] is suited particularly well. We discuss in the Section 4 how we chose the preconditioner K.

3.2 The locally optimal block preconditioned conjugate gradient algorithm

The locally optimal block preconditioned conjugate gradient (LOBPCG) algorithm has been introduced by Knyazev [17]. It is an improvement over the

(1) *Choose random matrix* $X_0 \in \mathbb{R}^{n\times q}$, $C^T X_0 = O$, *with* $X_0^T M X_0 = I_q$.
Set $Q := [\,]$.
(2) *Compute* $(X_0^T A X_0) S_0 = S_0 \Theta_0$, ***(Spectral decomposition)***
where $S_0^T S_0 = I_q$, $\Theta_0 = diag(\vartheta_1, \ldots, \vartheta_q)$, $\vartheta_1 \le \ldots \le \vartheta_q$.
(3) $X_0 := X_0 S_0$; $R_0 := AX_0 - MX_0\Theta_0$; $P_0 := [\,]$; $k := 0$.
(4) **while** *rank*$(Q) < p$ **do**
(5) *Solve the preconditioned linear system* $KH_k = R_k$
(6) $H_k := H_k - Q(Q^T M H_k)$; $H_k := (I - YH^{-1}C^T)H_k$.
(7) $\widetilde{A} := [X_k, H_k, P_k]^T A [X_k, H_k, P_k]$.
(8) $\widetilde{M} := [X_k, H_k, P_k]^T M [X_k, H_k, P_k]$.
(9) *Compute* $\widetilde{A}\widetilde{S}_k = \widetilde{M}\widetilde{S}_k\widetilde{\Theta}_k$ ***(Spectral decomposition)***
where $\widetilde{S}_k^T \widetilde{M} \widetilde{S}_k = I_{3q}$, $\widetilde{\Theta}_k = diag(\vartheta_1, \ldots, \vartheta_{3q})$, $\vartheta_1 \le \ldots \le \vartheta_{3q}$.
(10) $S_k := \widetilde{S}_k[\mathbf{e}_1, \ldots \mathbf{e}_q]$, $\Theta := diag(\vartheta_1, \ldots, \vartheta_q)$.
(11) $P_{k+1} := [H_k, P_k]\, S_{k,2}$; $X_{k+1} := X_k S_{k,1} + P_{k+1}$.
(12) $R_{k+1} := AX_{k+1} - MX_{k+1}\Theta_k$.
(13) $k := k + 1$.
(14) **for** $i = 1, \ldots, q$ **do** ***(Convergence test)***
(15) **if** $\|R_k \mathbf{e}_i\| < \varepsilon$ **then**
(16) $Q := [Q, X_k \mathbf{e}_i]$; $X_k \mathbf{e}_i := \mathbf{t}$, *with* $\mathbf{t}$ *a random vector.*
(17) *M-orthonormalize the columns of* X_k.
(18) **end if**
(19) **end for**
(20) **end while**

Algorithm 2. LOBPCG: The locally-optimal block preconditioned conjugate gradient method

(block) preconditioned conjugate gradient algorithm [14,19,21,9] for eigenvalue problems at the expense of a somewhat higher memory consumption.

In the block preconditioned conjugate gradient algorithm for solving symmetric eigenvalue problems, at step k a set of q Ritz vectors X_k and a set of q *search directions* P_k are available with $C^T X_k = C^T P_k = O$. The X_k satisfy $X^k M X^k = I$ and $X^k A X^k = \Theta_k$ where the diagonal matrix Θ_k has the Ritz values on its diagonal. The P_k are determined as a linear combination of the preconditioned residual H_k (projected onto $\mathcal{N}(C^T)$) and the previous search direction P_{k-1} such that $P_k^T A P_{k-1} = O$. Here, the block residual is $R_k = AX_k - MX_k\Theta_k$. The algorithm proceeds by defining the next approximations X_{k+1} to be the Ritz vectors of (3.1) projected onto $\mathcal{R}([X_k, P_k])$.

In the LOBPCG algorithm the X_{k+1} are defined to be Ritz vectors of (3.1) projected onto $\mathcal{R}([X_k, H_k, P_{k-1}])$, see Algorithm 2. The search directions P_k are defined only after the solution of the Ritz problem. If $\mathbf{d}_j = [\mathbf{d}_{1j}^T, \mathbf{d}_{2j}^T, \mathbf{d}_{3j}^T]^T$, $\mathbf{d}_{ij} \in \mathbb{R}^q$, is the eigenvector corresponding to the j-th eigenvalue of (3.1) restricted to $\mathcal{R}([X_k, H_k, P_{k-1}])$, then the j-th column of X_{k+1} is the corre-

sponding Ritz vector

$$X_{k+1}\mathbf{e}_j := [X_k, H_k, P_{k-1}]\ \mathbf{d}_j = X_k\mathbf{d}_{1j} + P_k\mathbf{e}_j, \tag{3.8}$$

with

$$P_k\mathbf{e}_j := H_k\mathbf{d}_{2j} + P_{k-1}\mathbf{d}_{3j}.$$

Notice that P_0 is an empty matrix such that the eigenvalue problem in step (9) of LOBPCG, displayed in Algorithm 2, has order $2q$ for $k = 0$.

4 The preconditioners

The eigensolvers that we have introduced in the previous section require a good preconditioner K in order to converge rapidly. In JDSYM the preconditioner appears in the solution of the correction equation. In LOBPCG the preconditioned residuals are computed in step (5) of the algorithm. To be a good preconditioner K has to satisfy two conditions. First, K must approximate well $A - \sigma M$ where σ is close to the eigenvalues that we want to compute. Second, systems involving K must be solvable much more quickly than systems with A.

Our preconditioner is a combination of a hierarchical basis preconditioner and an algebraic multigrid (AMG) preconditioner.

We first consider the hierarchical basis preconditioner as we used it in [2]. We recall that our finite element spaces consist of Nédélec and Lagrange finite elements of degree 2 and that we use hierarchical bases [27]. Numbering the linear before the qadratic degrees of freedom the matrices A and M in (1.2) and (2.2) have a 2-by-2 block structure,

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \qquad M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}, \tag{4.1}$$

where the $(1,1)$-blocks correspond to the bilinear forms involving linear basis functions in N_h.

The hierarchical basis preconditioners as discussed by Bank [3] are stationary iteration methods for solving

$$\begin{bmatrix} A_{11}^\sigma & A_{12}^\sigma \\ A_{21}^\sigma & A_{22}^\sigma \end{bmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix}, \qquad A_{ij}^\sigma = A_{ij} - \sigma M_{ij}. \tag{4.2}$$

that respect the 2-by-2 block structure of A and M.

If the underlying stationary method is block Jacobi iteration then

$$\begin{aligned} \mathbf{x}_1 &:= (A_{11}^{\sigma})^{-1}\mathbf{b}_1 \\ \mathbf{x}_2 &:= (\tilde{A}_{22}^{\sigma})^{-1}\mathbf{b}_2 \end{aligned} \tag{4.3}$$

whence

$$K = \begin{bmatrix} A_{11}^{\sigma} & \\ & \widetilde{A}_{22}^{\sigma} \end{bmatrix}.$$

If the underlying stationary method is the symmetric block Gauss-Seidel iteration then

$$\begin{aligned} \mathbf{x}_1' &:= (A_{11}^{\sigma})^{-1}\mathbf{b}_1, \\ \mathbf{x}_2 &:= (\tilde{A}_{22}^{\sigma})^{-1}(\mathbf{b}_2 - A_{21}^{\sigma}\mathbf{x}_1'), \\ \mathbf{x}_1 &:= (A_{11}^{\sigma})^{-1}(\mathbf{b}_1 - A_{12}^{\sigma}\mathbf{x}_2), \end{aligned} \tag{4.4}$$

which in matrix notation is

$$K = \begin{bmatrix} A_{11}^{\sigma} & \\ A_{21}^{\sigma} & \widetilde{A}_{22}^{\sigma} \end{bmatrix} \begin{bmatrix} A_{11}^{\sigma} & \\ & \widetilde{A}_{22}^{\sigma} \end{bmatrix}^{-1} \begin{bmatrix} A_{11}^{\sigma} & A_{12}^{\sigma} \\ & \widetilde{A}_{22}^{\sigma} \end{bmatrix}.$$

The approximation $\widetilde{A}_{22}^{\sigma}$ of A_{22}^{σ} again represents a stationary iteration method. With weighted Jacobi iteration we have

$$\tilde{A}_{22}^{\sigma} = 1/\omega D_{22} \tag{4.5}$$

and with SSOR iteration

$$\tilde{A}_{22}^{\sigma} = (D_{22} + \omega L_{22})D_{22}^{-1}(D_{22} + \omega L_{22}^{T}). \tag{4.6}$$

Here, D_{22} is the diagonal and L_{22} is the strictly lower triangle of A_{22}^{σ}. For $\omega = 1$ SSOR becomes symmetric Gauss-Seidel iteration which is the approach that we took in our experiments. Notice that the block methods and the inner methods can be combined arbitrarily.

In [2] we solved systems involving the (1,1) block A_{11}^{σ} of K by the direct solver SuperLU [6,7]. These systems appear in the first equation in (4.3) and in the first and third equation in (4.4). We also used SuperLU to solve with the Poisson matrix H of (2.5) that occurs in the projector $I - YH^{-1}C^{T}$.

For very large problems the direct solve with A_{11}^{σ} and H becomes inefficient and in particular consumes far too much memory due to fill-in. In order to reduce the memory requirements of the two-level hierarchical basis preconditioner but at the same time not lose its optimality with respect to iteration count we replaced the direct solves by one of two ways.

(1) The direct solution with A_{11}^{σ} or H is replaced by the highly accurate solution with a Krylov subspace method, QMRS or PCG, precondition by

one V-cycle of an algebraic multigrid (AMG) solver. Notice that systems involving H actually have to be solved to high accuracy as they are part of the application of a projector.

(2) The direct solution with A_{11}^{σ} is replaced by a single V-cycle of the same AMG preconditioner. This makes the preconditioner a true multilevel preconditioner. The two finest levels are defined by means of the two levels of the hierarchical basis. The coarser levels are obtained by a plain AMG approach.

We found ML 2.0 [15] the AMG solver of choice as it can handle unstructured systems that originate from the Maxwell equation discretized by linear Nédélec finite elements. ML implements a smoothed aggregation AMG method [26] that extends the non-smoothed aggregation approach of Reitzinger and Schöberl [22]. In the latter paper it is shown how the prolongation from one level to the next finer has to be constructed in order that discrete gradients of the coarser level are transfered to discrete gradients of the finer level. This implies in particular that the null space of the coarser level is mapped into the null space of the next finer level such that an equation corresponding to (2.3) holds on all levels. The construction of a smooth aggregation that respects these properties is discussed in [4]. The latter is implemented in ML 2.0 [15].

5 Numerical experiments

We conduct our numerical experiments in two steps. First we compare the previous preconditioners for solving the system (4.1) that represents up to the projections the correction equation of the JDSYM algorithm. These preconditioners are used 'stand-alone' in the LOBPCG algorithm. The preconditioners include combinations of the symmetric Gauss-Seidel variant of the hierarchical basis preconditioner combined with various solvers for the (1,1) block of $A - \sigma M$. We execute similar experiments for the Poisson equation (2.5). Second we compare JDSYM and LOBPCG using the best preconditioner of the first step.

5.1 A comparison of the preconditioners by means of relevant systems of equations

We compare the quality of various methods and preconditioners for solving the linear systems of equations occurring in Algorithm 1,

$$(A - \sigma M)\mathbf{x} = \mathbf{b} \tag{5.1}$$

and the discrete Poisson equation

$$H\mathbf{s} = \mathbf{t}. \tag{5.2}$$

A and M are the matrices obtained from FE-discretization of (2.1) using Nédélec elements as indicated in Section 2. H is the Poisson matrix defined in (2.5). For the following experiments we construct these matrices from two series of grids with increasing fineness as indicated in Tab. 1 in which orders n and numbers of *stored* non-zeros nnz of both the shifted operator $A - \sigma M$ and the discrete Laplacian H are listed.

grid	$n_{A-\sigma M}$	$\mathrm{nnz}_{A-\sigma M}$	n_H	nnz_H
box10k	63514	1197326	12243	280999
box60k	355738	7058180	71331	1798827
box170k	1030518	20767052	209741	5447883
box300k	1826874	36969839	374003	9795657
cop10k	50144	991995	9683	220085
cop40k	231668	4811786	46288	1163834
cop300k	1822854	39298588	373990	10098456

Table 1
Matrices used for numerical experiments

The symmetric indefinite system (5.1) is solved by the QMRS iteration method using the block symmetric Gauss-Seidel variant (4.4) of the two-level hierarchical basis preconditioner. $\tilde{A}_{22}$ in (4.4) is defined by (4.6) with $\omega = 1$. The iteration is stopped as soon as the residual norm has been reduced by a factor of 10^{-6}. For solving with the (1,1)-block of the preconditioner we investigate four variants:

- *2lev(lu)*: Systems with the (1,1)-block are solved exactly by the direct solver SuperLU [7]. To reduce fill-in the matrix is reordered with the symmetric minimum degree algorithm.
- *2lev(qmrs,ml)*: Systems with the (1,1)-block are solved to high accuracy (residual norm smaller than 10^{-14}) using the QMRS iteration method, preconditioned by AMG V-cycles.
- *2lev(ml)*: Solving with the (1,1)-block is replaced by a single AMG V-cycle.
- *2lev(sgs)*: Solving with the (1,1)-block is replaced by a single symmetric Gauss-Seidel (SGS) step.

The results are summarized in Tab. 2. The systems are solved to an accuracy of $\varepsilon = 10^{-6}$ using the QMRS iteration method. The *2lev(lu)* preconditioner could not be computed for the two largest systems due to memory restrictions. t_{init}

preconditioner		box10k	box60k	box170k	box300k	cop10k	cop40k	cop300k
2lev(lu)	t_{init}	15.9	1133.7	11573.1		7.8	271.5	
	t_{solv}	10.8	136.2	534.4		7.4	71.4	
	n_{it}	19	21	20		20	22	
	c	33.4	92.6	142.2		25.6	57.2	
2lev(qmrs,ml)	t_{init}	1.0	5.0	16.2	30.2	0.6	3.3	54.6
	t_{solv}	360.6	1767.7	6041.0	11664.2	234.3	1582.5	26592.6
	n_{it}	19	21	20	18	20	22	24
2lev(ml)	t_{init}	1.0	5.0	16.2	30.2	0.6	3.3	54.6
	t_{solv}	20.4	89.4	304.8	628.1	14.7	101.7	1354.1
	n_{it}	40	33	38	43	42	54	55
2lev(sgs)	t_{solv}	54.2	640.3	2798.5	6312.4	58.4	609.3	>13320
	n_{it}	233	454	708	899	322	573	>1000

Table 2
Solving the indefinite system (5.1) using four variations of the two-level hierarchical basis preconditioner. Execution times are in seconds.

is the time for constructing the preconditioner or the LU-factorization, respectively. For some methods t_{init} is omitted, indicating that the setup time is much smaller than t_{solv}, the time needed for solving one linear system (not including the setup time). n_{it} is the number of iterations executed until convergence. c is the operator complexity [25] of the LU-factorization, meaning that the LU-factors require c times as much memory as the system matrix. The execution times are reported in seconds.

Inside our eigensolvers the linear system solvers will be called many times, while there is only one setup phase. Therefore, a linear system solver (or more precisely a preconditioner) suits our needs if solution times and memory consumption are low. The numbers indicate, that *2lev(lu)* is the best method for the three smallest systems. However, as the matrices get larger, the fill-in produced by the LU factorization grows much faster than the matrix order and so do the memory consumption and the execution times.

For the more relevant larger problem sizes *2lev(ml)* is clearly the fastest. This is a true multilevel preconditioner. Solving the (1,1)-block to higher accuracy using more than one V-cycle as it is done in *2lev(qmrs,ml)* is much slower. This reduces the number of outer iterations n_{it} to the level of *2lev(lu)*. So, *2lev(qmrs,ml)* is in fact a 2-level method. On the other hand, replacing the V-cycle of *2lev(ml)* by a simple SGS step is evidently not enough to get an optimal preconditioner. The iteration count n_{it} grows considerably as h goes

method	grid	box10k	box60k	box170k	box300k	cop10k	cop40k	cop300k
lu	t_{init}	21.9	1343.2			12.4	557.4	
	t_{solv}	0.2	3.3			0.1	1.6	
	c	12.8	33.2			10.7	25.6	
cg, ml	t_{init}	1.0	9.8	36.3	73.1	0.9	6.9	133.9
	t_{solv}	4.1	21.3	71.4	139.9	2.8	12.9	218.9
	n_{it}	27	29	30	33	26	28	35
cg,sgs	t_{solv}	1.0	7.8	30.3	67.9	0.7	5.4	124.1
	n_{it}	41	44	59	69	40	48	81
cg,2lev,lu	t_{init}	0.1	9.9	139.8	640.5	0.1	3.0	492.1
	t_{solv}	1.5	16.5	77.2	182.9	1.0	8.9	178.7
	n_{it}	38	38	39	39	36	39	40
cg,2lev(cg,ml)	t_{solv}	6.5	34.6	96.6	177.1	7.0	21.5	210.4
	n_{it}	38	38	39	39	36	39	40
cg,2lev(ml)	t_{solv}	2.4	12.0	35.3	63.7	2.1	7.5	86.3
	n_{it}	38	39	40	40	37	39	41

Table 3
Solving the Poisson equation (5.2) to high accuracy using various methods. Execution times are given in seconds.

to zero.

The AMG method used for the *2lev(ml)* and *2lev(qmrs,ml)* preconditioners exhibits only a modest operator complexity which is less than 1.4 for all investigated grids. The number of generated multigrid levels ranges from 2 to 3.

Next, we compare methods for solving the symmetric positive definite Poisson equation (5.2) that is generated by Lagrange finite elements of degree two. The bases are again hierarchical such that we can again use two-level approaches as before. Here, however, the system matrix is symmetric positive definite such that the preconditioned conjugate gradient method can be applied in the iterative solves.

We investigate six solvers:

- *lu*: The system is solved by SuperLU. To reduce fill-in, symmetric minimum degree matrix reordering is applied.
- *cg,ml*: The system is solved using the conjugate gradient method preconditioned by a V-cycle of the smoothed aggregation AMG preconditioner

provided by ML.

- *cg,sgs*: The system is solved using the conjugate gradient method preconditioned with a symmetric Gauss-Seidel (SGS) step.
- *cg,2lev(lu)*: The system is solved using the conjugate gradient method with the symmetric Gauss-Seidel variant of the two-level hierarchical basis preconditioner. The (1,1)-block is solved exactly using the SuperLU direct solver. To reduce the fill-in, symmetric minimum degree matrix reordering is used.
- *cg,2lev(cg,ml)*: The system is solved using the conjugate gradient method with the two-level preconditioner. The (1,1)-block is solved to high accuracy (residual norm smaller than 10^{-14}) using the conjugate gradient method, preconditioned by AMG V-cycles.
- *cg,2lev(ml)*: The system is solved using the conjugate gradient method with the two-level preconditioner. Solving with the (1,1)-block is replaced by a single AMG V-cycle.

The results are summarized in Tab. 3. The systems are solved either directly or using the CG iterative method to an accuracy of $\varepsilon = 10^{-14}$.

As in the previous example, the direct solution (*lu*) is the best method for solving small problems. However, as the matrices get larger, the fill-in produced by the LU factorization grows much faster than the matrix order and so do the memory consumption and the execution times. The LU factorization of H could not even be computed for the three largest systems due to memory restrictions.

The preconditioned conjugate gradient method is better suited for solving the large systems. Surprisingly, the cheap SGS preconditioner is very efficient and the corresponding iteration numbers only grow moderately as the problem size increases. The SGS preconditioner is the most efficient except for the two largest problems where it is beat by *cg,2lev(ml)*. *cg,sgs* and *cg,2lev(ml)* are quite close for medium sized problems. The SGS preconditioner is more than twice as fast as the AMG preconditioner *cg,ml* for all grids except for cop300k for which it is still considerably faster.

The three variants of the two-level hierarchical preconditioner are all optimal in the sense that n_{it} does not depend on the problem size. However *cg,2lev(lu)* and *cg,2lev(cg,ml)* are both not competitive since their computational cost is too high. The former consumes memory excessively, too. Interestingly, *cg,2lev(ml)* needs just as many iterations as *cg,2lev(cg,ml)*.

The AMG method used for *cg,ml*, *cg,2lev(ml)*, *cg,2lev(cg,ml)* exhibits only a modest operator complexity which is below 1.8 for all investigated grids. The number of generated multigrid levels ranges from 2 to 4.

Finally, we present results for computing a few of the smallest eigenvalues and corresponding eigenvectors using both the JDSYM and LOBPCG eigenvalue solvers. For the following experiments we choose the *2lev(ml)* preconditioners, which turned out to be the best choice for large problem sizes for solving both the symmetric indefinite problem (5.1) and the Poisson equation (5.2) as shown in Section 5.1.

		box10k	box60k	box170k	box300k	cop10k	cop40k	cop300k
jdsym	t_{eig}	376.2	2237.8	7223.1	14153.9	258.1	1853.2	25823.3
$p = 5$	n_{out}	33	33	34	35	32	35	34
	n_{in}	13.1	14.8	16.2	17.4	12.5	18.1	19.7
lobpcg	t_{eig}	580.8	3463.6	11652.2	22676.1	549.6	2584.9	38207.1
$p = 5$	n_{out}	33	40	42	45	40	45	58
jdsym	t_{eig}	852.6	5150.8	15960.2	30785.9	559.2	3794.5	48283.0
$p = 10$	n_{out}	67	68	70	70	65	70	70
	n_{in}	14.0	15.81	16.8	18.0	13.1	17.5	19.0
lobpcg	t_{eig}	1175.9	7311.2	22476.4	48159.2	955.2	5019.2	60719.7
$p = 10$	n_{out}	41	46	47	61	38	48	54

Table 4
Computing some of the smallest positive eigenvalues of (1.2) using JDSYM and LOBPCG. The execution times are reported in seconds.

The results for computing the $p = 5$ and $p = 10$ smallest eigenvalues of (1.2) using JDSYM and LOBPCG are given in Tab. 4. The residual norms of the computed eigenvectors were required to be below $\varepsilon = 10^{-6}$. t_{eig} is the total time spent for solving the eigenvalue problem. Matrix assemblies and setup of the preconditioner are not included. n_{out} is the number of JDSYM iterations taken, or the number of LOBPCG block iterations, respectively. n_{in} is the *average* number of inner iterations per outer iteration. The total number of matrix-vector multiplications and applications of the preconditioner is $n_{\text{out}} \cdot n_{\text{in}}$ for JDSYM and approximately $(p+1)n_{\text{out}}$ because the block size in LOBPCG was $q = p+1$. This number is approximate as we shrink the block size when eigenpairs are found. In JDSYM we set $j_{\min} = p+1$ and $j_{\max} = p+10$.

In JDSYM n_{out} is almost constant. n_{in} grows only slightly with the problem size indicating that the *2lev(ml)* preconditioner is effective also in the context of the JDSYM eigenvalue solver. In analogy, for LOBPCG n_{out} grows moderately with the problem size.

The number of applications of the operator and of the preconditioner is considerably smaller for LOBPCG than for JDSYM. Nevertheless, JDSYM is the faster method for all investigated problems no matter whether five or ten eigenpairs were computed. The main reason is that JDSYM applies the projector $P_{\mathcal{N}(C^T)} = I - YH^{-1}C^T$ only once per *outer* iteration, while LOBPCG applies it $p + 1$ times in each block iteration. Applying the projector essentially amounts to solving a Poisson equation. Also notice that the sizes of the problems considered are reasonably large such that the blocks that are employed by LOBPCG cannot be kept in cache. But still many of the operations in LOBPCG can be executed by level-2 or level-3 BLAS [8]. However the smoothed aggregation AMG preconditioner of ML 2.0 can only deal with one vector at a time.

During these experiments JDSYM also proved to be the more stable method. On several occasions LOBPCG suffered from breakdowns which could only be cured by complete restarts using random vectors.

6 Conclusions

We have presented efficient algorithms for computing a few of the lowest eigenpairs of a Maxwell eigenvalue problem that occurs in the design of particle accelerator cavities. The large sparse unstructured matrix eigenvalue problem that is obtained by discretization using a combination of Nédélec and Lagrange finite elements has been solved by the symmetric Jacobi-Davidson algorithm and by the LOBPCG method. We have investigated various preconditioners to enhance the rate of convergence of these algorithms. The preconditioners are extensions of a two-level hierarchical basis preconditioner presented in earlier papers. As the sizes of our problems increased the direct solution of the coarse grid system became infeasible due to memory limitations. It turned out that the most effective method results when replacing the direct coarse grid solver by a single V-cycle of an AMG preconditioner, in our case the ML smoothed aggregation AMG preconditioner. In this way we obtain eigensolvers that are scalable with respect to operation count and memory consumption. For solving our Maxwell eigenvalue problem the symmetric Jacobi-Davidson algorithm is superior to LOBPCG. The latter requires many more applications of the projector that forces the approximations into the subspace of (discrete) divergence free finite element functions.

LOBPCG offers some opportunities for improving the cache-efficiency that have not been not considered in our current implementation of the algorithm. In particular the sparse matrix-vector products in step (12) of Algorithm 2 are computed in a non-blocked fashion. Blocked versions of the matrix-vector products and of the AMG preconditioner should improve the performance of

our LOBPCG implementation considerably.

We have solved problems with several millions degrees of freedom. These problems are however too large to be included in a comparative study. Their solution took many hours even with our most efficient version of the Jacobi-Davidson algorithm. The next step in our code development will therefore be concerned with its parallelization. The dense and sparse matrix-vector products in Algorithms 1 and 2 offer a good degree of parallelism. A parallel version of the AMG preconditioner provided by the ML library is also available.

References

[1] P. Arbenz and R. Geus. A comparison of solvers for large eigenvalue problems originating from Maxwell's equations. *Numer. Lin. Alg. Appl.*, 6(1):3–16, 1999.

[2] P. Arbenz, R. Geus, and S. Adam. Solving Maxwell eigenvalue problems for accelerating cavities. *Phys. Rev. ST Accel. Beams*, 4:022001, 2001. (Electronic journal available from `http://prst-ab.aps.org/`).

[3] R. E. Bank. Hierarchical bases and the finite element method. *Acta Numerica*, 5:1–43, 1996.

[4] P. Bochev, C. Garasi, J. Hu, A. Robinson, and R. Tuminaro. An improved algebraic multigrid method for solving Maxwell's equations. Technical Report Sand2002-8222J, Sandia National Laboratories, Albuquerque NM, May 2002. To appear in SISC.

[5] Ph. G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland, Amsterdam, 1978. (Studies in Mathematics and its Applications, 4).

[6] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Anal. Appl.*, 20(3):720–755, 1999.

[7] J. W. Demmel, J. R. Gilbert, and X. S. Li. *SuperLU Users' Guide*, December 2002. Available from `http://acts.nersc.gov/superlu/main.html`.

[8] J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. van der Vorst. *Numerical Linear Algebra for High-Performance Computers*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1998.

[9] Y. T. Feng and D. R. J. Owen. Conjugate gradient methods for solving the smallest eigenpair of large symmetric eigenvalue problems. *Internat. J. Numer. Methods Engrg.*, 39(13):2209–2229, 1996.

[10] D. R. Fokkema, G. L. G. Sleijpen, and H. A. van der Vorst. Jacobi-Davidson style QR and QZ algorithms for the partial reduction of matrix pencils. *SIAM J. Sci. Comput.*, 20(1):94–125, 1998.

[11] R. Freund and N. M. Nachtigal. Software for simplified Lanczos and QMR algorithms. *Appl. Numer. Math.*, 19:319–341, 1995.

[12] R. Geus. *The Jacobi-Davidson algorithm for solving large sparse symmetric eigenvalue problems*. PhD Thesis No. 14734, ETH Zurich, 2002. (Available at URL `http://e-collection.ethbib.ethz.ch/show?type=diss&nr=14734`).

[13] V. Girault and P.-A. Raviart. *Finite Element Methods for the Navier-Stokes Equations*. Springer-Verlag, Berlin, 1986. (Springer Series in Computational Mathematics, 5).

[14] M. R. Hestenes and W. Karush. A method of gradients for the calculation of the characteristic roots and vectors of a real symmetric matrix. *Journal of Research of the National Bureau of Standards*, 47:45–61, 1951.

[15] J. Hu, C. Tong, and R. S. Tuminaro. ML 2.0 smoothed aggregation user's guide. Tech. Report SAND2001-8028, Sandia National Laboratories, December 2000.

[16] F. Kikuchi. Mixed and penalty formulations for finite element analysis of an eigenvalue problem in electromagnetism. *Comput. Methods Appl. Mech. Eng.*, 64:509–521, 1987.

[17] A. V. Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM J. Sci. Comput.*, 23(2):517–541, 2001.

[18] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems by Implicitely Restarted Arnoldi Methods*. SIAM, Philadelphia, PA, 1998. (The software and this manual are available at URL `http://www.caam.rice.edu/software/ARPACK/`).

[19] D. E. Longsine and S. F. McCormick. Simultaneous Rayleigh–quotient minimization methods for $Ax = \lambda Bx$. *Linear Algebra Appl.*, 34:195–234, 1980.

[20] B. N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice Hall, Englewood Cliffs, NJ, 1980. (Republished by SIAM, Philadelphia, 1998.).

[21] A. Perdon and G. Gambolati. Extreme eigenvalues of large sparse matrices by Rayleigh quotient and modified conjugate gradients. *Comput. Methods Appl. Mech. Engrg.*, 56:125–156, 1986.

[22] S. Reitzinger and J. Schöberl. An algebraic multigrid method for finite element discretizations with edge elements. *Numer. Linear Algebra Appl.*, 9(3):223–238, 2002.

[23] P. P. Silvester and R. L. Ferrari. *Finite Elements for Electrical Engineers*. Cambridge University Press, Cambridge, 3rd edition, 1996.

[24] G. L. G. Sleijpen and H. A. van der Vorst. A Jacobi-Davidson iteration method for linear eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 17(2):401–425, 1996.

[25] K. Stüben. An introduction to algebraic multigrid. In U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Appendix A. Academic Press, 2000.

[26] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid based on smoothed aggregation for second and fourth order problems. *Computing*, 56(3):179–196, 1996.

[27] H. Yserentant. Hierarchical bases. In R. E. O'Malley, editor, *ICIAM 91: Proceedings of the Second International Conference on Industrial and Applied Mathematics*, pages 256–276. SIAM, Philadelphia, PA, 1992.