An optimized and scalable eigensolver for sequences of eigenvalue problems

Berljafa, Mario and Wortmann, Daniel and Di Napoli, Edoardo

2014

MIMS EPrint: **2014.35**

ISSN 1749-9097

An Optimized and Scalable Eigensolver for Sequences of Eigenvalue Problems[*]

Mario Berljafa [†] Daniel Wortmann [‡] Edoardo Di Napoli[§]

July 6, 2014

Abstract

In many scientific applications the solution of non-linear differential equations are obtained through the set-up and solution of a number of successive eigenproblems. These eigenproblems can be regarded as a sequence whenever the solution of one problem fosters the initialization of the next. In addition, in some eigenproblem sequences there is a connection between the solutions of adjacent eigenproblems. Whenever it is possible to unravel the existence of such a connection, the eigenproblem sequence is said to be correlated. When facing with a sequence of correlated eigenproblems the current strategy amounts to solving each eigenproblem in isolation. We propose a alternative approach which exploits such correlation through the use of an eigensolver based on subspace iteration and accelerated with Chebyshev polynomials (ChFSI). The resulting eigensolver is optimized by minimizing the number of matrix-vector multiplications and parallelized using the Elemental library framework. Numerical results show that ChFSI achieves excellent scalability and is competitive with current dense linear algebra parallel eigensolvers.

[*]Article based on research supported by the Excellence Initiative of the German federal and state governments and the Jülich Aachen Research Alliance High-Performance Computing.

[†]School of Mathematics, The University of Manchester, Alan Turing Building, M13 9PL, Manchester, United Kingdom. `m.berljafa@maths.man.ac.uk`.

[‡]Institute for Advanced Simulation and Peter Grünberg Institut, Forschungszentrum Jülich and JARA, 52425 Jülich, Germany. `d.wortmann@fz-juelich.de`.

[§]Jülich Supercomputing Centre, Institute for Advanced Simulation, Forschungszentrum Jülich and JARA, Wilhelm-Johnen strasse, 52425 Jülich, Germany. `e.di.napoli@fz-juelich.de`.

1 Introduction

In many scientific applications the solution of Hermitian eigenproblems is key to a successful numerical simulation. A considerable subset of applications requires the solution of not just one eigenproblem but a sequence of them. An exemplary case is provided by the non-linear eigenvalue problem where the solution is computed through a step-wise self-consistent process: at each step the solution of a linearized eigenvalue problem is used to initialize the next one until an application-specific criterion is satisfied. In this context a well-known example is represented by Density Functional Theory (DFT), one of the most important frameworks used in Material Science and Quantum Chemistry.

While there is a large amount of literature on algorithms developed to solve for an isolated Hermitian (standard or generalized) eigenvalue problem, the same cannot be said for a sequence of them. It is the aim of this paper to illustrate an alternative approach to solve a sequence of Hermitian eigenproblems when a degree of correlation between their eigenpairs is made manifest. We take inspiration from sequences of dense eigenproblems arising in DFT, but want to stress that such approach can work as well in other applications provided that the sequence possesses analogous properties.

Eigenproblem sequences — It is important to distinguish between the notion of sequence of eigenproblems and the correlation that might exist between adjacent eigenproblems in a sequence. The concept of sequence is very general and could be loosely borrowed from how sequences of real numbers are defined.

Definition 1.1. *A sequence of eigenproblems is defined as an N-tuple $\{P\}_N \equiv P^{(1)}, \ldots, P^{(\ell)}, \ldots, P^{(N)}$ of problems $P^{(\ell)}$ with same size n such that the eigenpairs of the ℓ-problem are used (directly or indirectly) to initialize the $\ell + 1$-problem.*

The above definition does not, in general, imply that a connection between adjacent problems in a sequence is direct or simple. It just states that the sequence is generated through a output/input process: the solutions of an eigenproblem in the sequence are manipulated to generate the next eigenproblem. In order to express more precisely the idea of "connection" between eigenproblems, we introduced the notion of correlation (not to be confused to the definition of correlation used in probability calculus).

Definition 1.2. *Two eigenproblems $P^{(\ell)}$ and $P^{(\ell+1)}$ are said to be correlated when the eigenpairs of the $\ell + 1$-problem are in some relation with the eigenpairs of the ℓ-problem.*

There may be many ways to measure a correlation between eigenpairs of two distinct eigenproblems. For instance, the eigenvalues of one matrix could be shifted with respect to the eigenvalues of the previous one by a value proportional to the eigenvalue itself. In this paper we address correlation between eigenpairs by measuring the angle between corresponding eigenvectors as a function of the sequence index ℓ. Whenever it is difficult to find such a correlation in exact mathematics a numerical approach may be preferred [13].

The current approach to solve for eigenproblem sequences depends on the properties of the matrices defining the problems and on the percentage of eigenspectrum required. Unless very few eigenpairs are required, dense eigenproblems are solved with direct eigensolvers provided by libraries such as LAPACK [2]. Sparse eigenproblems prefer iterative eigensolvers, the most popular of which is provided by the ARPACK package [22]. In both cases the invoked eigensolvers are used as black-boxes and the correlation between eigenproblems (when present) is not exploited. Moreover, it was recently shown that when used on parallel architectures iterative eigensolvers can still be used on dense problems and be competitive with direct ones [1]. Together, these last two observations, indicate the need for a viable third option specifically dedicated to sequences of eigenproblems.

In this paper we illustrate such an alternative approach to solve sequences which show some degree of eigenvector correlation. The novelty of the approach consists of developing a method which takes advantage of the available information provided by the correlation. To be specific, we exploit the eigenvectors (and the extremal eigenvalues) of the ℓ-problem in order to facilitate the computation of the solution of $\ell+1$-problem. This is made possible by inputting the ℓ-eigenvectors to a simple block eigensolver based on subspace iteration accelerated with Chebyshev polynomials [26, 35]. We show that such a block eigensolver can be further optimized by appropriately tuning the polynomial degree of the accelerator. When parallelized on distributed memory architectures our Chebyshev Filtered Subspace Iteration (ChFSI) is a viable alternative to conventional eigensolvers both in terms of scalability and performance.

Sequences in DFT — Within the realm of condensed-matter physics, DFT is considered the standard model to run accurate simulations of materials. The importance of these simulations is two-fold. On the one hand they are used to verify the correctness of the quantum mechanical interpretation of existing materials. On the other hand, simulations constitute an extraordinary tool to verify the validity of new atomistic models which may ultimately lead to the invention of brand new materials.

Each simulation consists of a series of self-consistent field (SCF) cycles; within each cycle a fixed number $\mathcal{N}_{\mathbf{k}}$ of independent eigenvalue problems is solved. Since dozens of cycles are necessary to complete one simulation, one ends up with $\mathcal{N}_{\mathbf{k}}$ sequences made of dozens of eigenproblems. The properties of these eigenproblems depend on the discretization strategy of the specific DFT method of choice. In this paper we will exclusively consider the Full-Potential Linearized Augmented Plane Waves method (FLAPW). FLAPW gives rise to sequences of dense hermitian generalized eigenproblems (GHEVP) with matrix size typically ranging from 2,000 to 30,000. In FLAPW only a fraction of the lowest part of the eigenspectrum is required. The eigenvalues inside this fraction correspond to the energy levels below Fermi energy and their number never falls below 2% or exceeds 20% of the eigenspectrum. The relatively high number of eigenpairs in combination with the dense nature and the size of the eigenproblems inevitably lead to the choice of direct eigensolvers.

Until very recently, the computational strategy on parallel distributed memory architecture favored the use of ScaLAPACK [6] implementation of the bisection and inverse iteration algorithm (BXINV). Modern and efficient dense libraries, like ELPA [3] and EleMRRR [24], improve the performance but do not change the overall computational strategy: each problem in the sequence is solved in complete independence from the previous one. The latter choice is based on the view that problems in the sequence are considered only loosely connected. In fact despite the solution of one problem ultimately determines the initialization of the next, it does so in such a mathematically indirect manner that the solutions of two successive problems seem to be quite independent.

Due to their properties and the current strategy used to solve them, the FLAPW generated sequences constitute the ideal ground to illustrate the workings for the ChFSI eigensolver. This is made possible because, in spite of the assumed loose connection between adjacent eigenproblems, it has been shown that they possess a good degree of correlation which is explicitly expressed by the evolution of the eigenvectors angles as a function of the sequence index ℓ [13].

This paper is an invited contribution and it has been specifically written as an extension of an earlier publication [5]. In particular we have substantially enlarged each section with new material, included new numerical tests and plots and devoted an entire subsection to the optimization of the polynomial filter. In Sec. 2 we illustrate how sequences of eigenproblems arise in Density Functional Theory and briefly present results on correlated sequences. The ChFSI algorithm is described in Sec. 3. Here we give details on the algorithm structure with special focus on the Chebyshev filter. We also devote a subsection to the MPI parallelization of the whole code. In Sec. 4 we present a number of numerical tests and their analysis. We summarize and conclude in Sec. 5.

2 Sequences of correlated eigenproblems

In this section we describe how sequences of GHEVP arise in electronic structure computations. In the first part of the section, we provide the reader with an outline of the mathematical set-up and illustrate some of the important features which justify interpreting the generated eigenproblems as a sequence. In the second part, we show how these sequences are also correlated and how such correlation can be harnessed and exploited in order to speed-up the solution of the whole sequence of problems. We want to stress once again that the particular application where these sequences appear constitute just a practical example of our method. Such a method can be applied to any sequence of eigenproblems endowed with a reasonable level of correlation between its eigenvectors.

2.1 Eigenproblems in Density Functional Theory

Every DFT method is based on a variational principle stemming from the fundamental work of Kohn and Hohenberg [17], and its practical realization [20]. Central to DFT is the solution of a large number of coupled one-particle Schrödinger-like equations known as Kohn-Sham (KS).

$$\hat{H}_{\rm KS}\phi_a(\mathbf{r}) \equiv \left(\frac{\hbar^2}{2m}\nabla^2 + \mathcal{V}_{\rm eff}[n(\mathbf{r})]\right)\phi_a(\mathbf{r}) = \varepsilon_a\phi_a(\mathbf{r}) \quad ; \quad n(\mathbf{r}) = \sum_a f_a|\phi_a(\mathbf{r})|^2$$

Due to the dependence of the effective potential $\mathcal{V}_{\rm eff}$ on the charge density $n(\mathbf{r})$, in itself a function of the orbital wave functions $\phi_a(\mathbf{r})$, the KS equations are non-linear and are generally solved through a self-consistent process. Schematically, such process is divided in a sequence of outer-iterative SCF cycles: it starts off with an initial charge density $n_{\rm init}(\mathbf{r})$, proceeds through a series of iterations and converges to a final density $n^{(N)}(\mathbf{r})$ such that $|n^{(N)} - n^{(N-1)}| < \eta$, with η as an a priori parameter.

$$\begin{tikzcd}
\boxed{\begin{array}{c}\text{Initial input}\\ n_{\text{init}}(\mathbf{r})\end{array}} \arrow[r, Rightarrow] & \boxed{\begin{array}{c}\text{Compute KS potential}\\ \mathcal{V}_{\text{eff}}[n(\mathbf{r})]\end{array}} \arrow[r] & \boxed{\begin{array}{c}\text{Solve KS equations}\\ \hat{H}_{\text{KS}}\phi_a(\mathbf{r}) = \varepsilon_a\phi_a(\mathbf{r})\end{array}} \arrow[d] \\
\boxed{\begin{array}{c}\text{OUTPUT}\\ \text{Energy, forces, etc.}\end{array}} & \boxed{\begin{array}{c}\text{Converged?}\\ |n^{(\ell)} - n^{(\ell-1)}| < \eta\end{array}} \arrow[l, Rightarrow, "\text{Yes}"'] \arrow[u, "\text{No}"'] & \boxed{\begin{array}{c}\text{Compute new density}\\ n(\mathbf{r}) = \sum_a f_a|\phi_a(\mathbf{r})|^2\end{array}} \arrow[l]
\end{tikzcd} \tag{1}$$

In practice this outer-iterative cycle is still quite computationally challenging and requires some form of broadly defined discretization. Intended in its broadest numerical sense, the discretization translates the KS equations in a non-linear eigenvalue problem.

Eigenproblems generated by distinct discretization schemes have numerical properties that are often significantly different; for sake of simplicity we can group most of the schemes in three classes. The first and the second classes make respectively use of plane waves and localized functions to expand the one-particle orbital wave functions $\phi_a(\mathbf{r})$ appearing in the KS equations

$$\phi_a(\mathbf{r}) \longrightarrow \phi_{\mathbf{k},i}(\mathbf{r}) = \sum_{\mathbf{G}} c^{\mathbf{G}}_{\mathbf{k},i}\psi_{\mathbf{G}}(\mathbf{k},\mathbf{r}). \tag{2}$$

Methods in the third class do not use an explicit basis for the $\phi_a(\mathbf{r})$'s but discretize the KS equations on a grid in real space using finite differences. The eigenvalue problems emerging from the first two discretization classes consist of dense matrices of small-to-moderate size while, within real space methods, one ends up with very large sparse matrices. Due to the dramatically different set of properties of the eigenproblems, each DFT method uses a distinct strategy in solving for the required eigenpairs. For instance it is quite common that methods based on plane waves use direct eigensolvers while real space methods make use of iterative eigensolver based on Krylov- or Davidson-like subspace construction. From the point of view of software packages for distributed memory architectures, the choice between direct or iterative eigensolvers can lead respectively to the use of traditional parallel libraries such as ScaLAPACK or PARPACK.

In this paper we focus on a specific instance of a plane wave method which splits the basis functions support domain (Muffin-Tin): in a spherical symmetric area around each atom labelled by α, $\psi_{\mathbf{G}}$ receive contributions by augmented radial functions $a^{\alpha,\mathbf{G}}_{lm}u^{\alpha}_{l} + b^{\alpha,\mathbf{G}}_{lm}\dot{u}^{\alpha}_{l}$ times the spherical harmonics Y_{lm}, while plane waves are supported in the interstitial space between atoms.

$$\psi_{\mathbf{G}}(\mathbf{k},\mathbf{r}) = \begin{cases} \frac{1}{\sqrt{\Omega}}e^{i(\mathbf{k}+\mathbf{G})\mathbf{r}} & \text{– Interstitial} \\ \sum\limits_{l,m}\left[a^{\alpha,\mathbf{G}}_{lm}(\mathbf{k})u^{\alpha}_{l}(r) + b^{\alpha,\mathbf{G}}_{lm}(\mathbf{k})\dot{u}^{\alpha}_{l}(r)\right]Y_{lm}(\hat{\mathbf{r}}_{\alpha}) & \text{– Spheres.} \end{cases}$$

At each iteration cycle the radial functions $u^{\alpha}_{l}(r)$ are computed anew by solving auxiliary Schrödinger equations. Moreover a new set of the coefficients $a^{\alpha,\mathbf{G}}_{lm}$ and $b^{\alpha,\mathbf{G}}_{lm}$ is derived by imposing continuity constraints on the surface of the Muffin-Tin spheres. Consequently at each iteration the basis set $\psi_{\mathbf{G}}(\mathbf{k},\mathbf{r})$ changes entirely. This discretization of the KS equations – known as FLAPW – translates in a set of $\mathcal{N}_{\mathbf{k}}$ quite dense GHEVPs

$$\sum_{\mathbf{G}'}(A_{\mathbf{k}})_{\mathbf{G}\mathbf{G}'}\, c^{\mathbf{G}'}_{\mathbf{k},i} = \lambda_{\mathbf{k},i}\sum_{\mathbf{G}'}(B_{\mathbf{k}})_{\mathbf{G}\mathbf{G}'}\, c^{\mathbf{G}'}_{\mathbf{k},i},$$

each one labeled by a value of the plane wave vector $\mathbf{k}$. The role of eigenvectors is played by the n-tuple of coefficients $c_{\mathbf{k},i}$ expressing the orbital wave functions ϕ_i in terms of the basis wave functions $\psi_{\mathbf{G}}$. The entries of each GHEVP matrix are initialized by evaluating numerically a series of expensive multiple integrals involving the $\psi_{\mathbf{G}}$s. Since we are dealing with non-linear eigenvalue problems, one resorts to the self-consistent process described above which now becomes

$$\begin{tikzcd}
\boxed{\begin{array}{c}\text{Initial input}\\ n_{\text{init}}(\mathbf{r})\end{array}} \arrow[r, Rightarrow] & \boxed{\begin{array}{c}\text{Compute KS potential}\\ \mathcal{V}_{\text{eff}}[n(\mathbf{r})]\end{array}} \arrow[r] & \boxed{\begin{array}{c}\text{Solve a set of eigenproblems}\\ P^{(\ell)}_{\mathbf{k}_1}\dots P^{(\ell)}_{\mathbf{k}_N}\end{array}} \arrow[d] \\
\boxed{\begin{array}{c}\text{OUTPUT}\\ \text{Energy, \dots}\end{array}} & \boxed{\begin{array}{c}\text{Converged?}\\ |n^{(\ell)}-n^{(\ell-1)}|<\eta\end{array}} \arrow[l, Rightarrow, "\text{Yes}"'] \arrow[u, "\text{No}"'] & \boxed{\begin{array}{c}\text{Compute new density}\\ n(\mathbf{r})=\sum_{\mathbf{k},i} f_{\mathbf{k},i}|\phi_{\mathbf{k},i}(\mathbf{r})|^2\end{array}} \arrow[l]
\end{tikzcd}$$

There are now $\mathcal{N}_{\mathbf{k}}$ GHEVP for each cycle labeled by ℓ

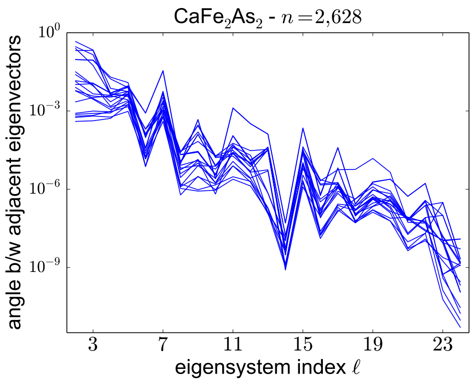
$$P^{(\ell)}_{\mathbf{k}}: \quad A^{(\ell)}_{\mathbf{k}}\, c^{(\ell)}_{\mathbf{k},i} = \lambda^{(\ell)}_{\mathbf{k},i}\, B^{(\ell)}_{\mathbf{k}}\, c^{(\ell)}_{\mathbf{k},i} \qquad (\ell = 1,\dots,N).$$

All along the sequence the solutions of all $P^{(\ell-1)}_{\mathbf{k}}$ are used to initialize the new eigenproblems $P^{(\ell)}_{\mathbf{k}}$. In particular the eigenvectors $c^{(\ell-1)}_{\mathbf{k},i}$ are used to derive the orbital functions $\phi^{(\ell-1)}_{\mathbf{k},i}$ which in turn contribute to the charge density $n^{(\ell-1)}(\mathbf{r})$. At the next cycle $n^{(\ell-1)}(\mathbf{r})$ contributes to modify the potential $\mathcal{V}_{\text{eff}}$ which causes the radial functions defining $\psi^{(\ell)}_{\mathbf{G}}$s to change. This new basis function set directly determines the initialization of the entries of $A^{(\ell)}_{\mathbf{k}}$ and $B^{(\ell)}_{\mathbf{k}}$ and indirectly the new eigenvectors $c^{(\ell)}_{\mathbf{k},i}$. The result is a number $\mathcal{N}_{\mathbf{k}}$ of sequences of eigenproblems $\{P_{\mathbf{k}}\}_N$, one sequence for each fixed $\mathbf{k}$, where the eigenpairs $(\lambda^{(N)}_{\mathbf{k},i}, c^{(N)}_{\mathbf{k},i})$ converged within tolerance to the solution of the original non-linear problem.

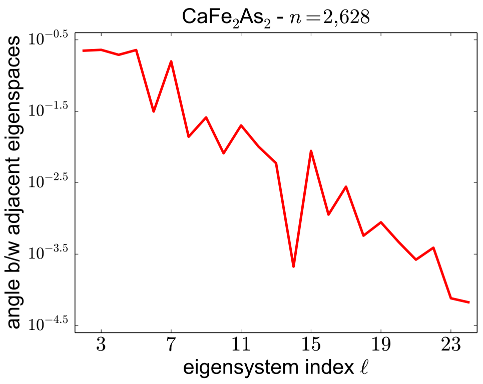
When solving for an eigenvalue problem the first high level choice is between direct and iterative eigensolvers. The first are in general used to solve for a large portion of the eigenspectrum of dense problems. The latter are instead the typical choice for sparse eigenproblems or used to solve for just few eigenpairs of dense ones. In FLAPW the hermitian matrices $A_{\mathbf{k}}$ and $B_{\mathbf{k}}$ are quite dense, have size generally not exceeding 30,000, and each $P^{(\ell)}_{\mathbf{k}}$ is solved for a portion of the lower spectrum not bigger than 20%. Consequently, when each GHEVP is singled out from the rest of the sequence, direct solvers are unquestionably the method of choice. Currently, most of the codes based on FLAPW methods [8, 7, 10] use the algorithms BXINV or MRRR directly out of the ScaLAPACK or ELPA library. If the use of direct solvers is the obvious choice when each $P^{(\ell)}_{\mathbf{k}}$ is solved in isolation, the same conclusion may not be drawn when we look at the entire sequence of $\{P_{\mathbf{k}}\}_N$.

2.2 Harnessing the correlation

As described in the previous section, the chain of computations that goes from $P^{(\ell-1)}_{\mathbf{k}}$ to $P^{(\ell)}_{\mathbf{k}}$ suggests a connection between eigenvectors of successive eigenproblems. The entries of $P^{(\ell)}_{\mathbf{k}}$ are in fact the result of multiple integrals between $\psi_{\mathbf{G}}^{(\ell)}$ and operators depending on the new charge density $n^{(\ell-1)}(\mathbf{r})$. All these quantities are modified by $c_{\mathbf{k},i}^{(\ell-1)}$ collectively in such a complex fashion across each cycle that there is no accessible mathematical formulation which makes explicit the connection between $c_{\mathbf{k},i}^{(\ell-1)}$ and $c_{\mathbf{k},i}^{(\ell)}$. In other words, according to our definitions, it is clear that each set of problems $P_{\mathbf{k}}^{(\ell)} \quad \ell = 1,\ldots,N$ is a sequence but there is no immediate evidence that such sequence is made of correlated eigenproblems.



(a) Evolution of eigenvectors angles.



(b) Evolution of eigenspace angle.

Figure 1: The data in this figure refers to a sequence of eigenproblems relative to the same physical system CaFe$_2$As$_2$. Plot (a) represents the angles between eigenvectors corresponding to the lowest 20 eigenvalues. Plot (b) shows the angle between subspaces spanned by a fraction of the eigenvectors of adjacent eigenproblems.

Evidence of the existence of a correlation between the eigenvectors of successive eigenproblems becomes clear only numerically. As shown in [13] the correlation manifests itself in the evolution of the angles $\theta_{\mathbf{k},i}^{(\ell)} = \left(1 - \langle c_{\mathbf{k},i}^{(\ell-1)}, c_{\mathbf{k},i}^{(\ell)} \rangle\right)$. In particular the $\theta_{\mathbf{k},i}^{(\ell)}$ decrease almost monotonically as a function of cycle index ℓ, going from $\sim 10^{-1}$ down to $\sim 10^{-8}$ towards the end of the sequence (see plot (a) of Fig. 1). Eigenvectors thus become more and more collinear with increasing ℓ. Plot (b) of Fig. 1 clearly shows that even when measuring the full sought after eigenspace of adjacent eigenproblems, the entire subspace changes to a lesser extent as the sequence index grows.

The evolution of the eigenvectors in Fig. 1 suggests that they can be "reused" as approximate solutions, and inputted to the eigensolver at the successive cycle. Unfortunately no direct eigensolver is capable of accepting vectors as approximate solutions. Therefore if we want to exploit the progressive collinearity of vectors as the sequence progresses, we are lead to consider iterative solvers; these solvers by their own nature build approximate eigenspaces by manipulating approximate eigenvectors. When inputting to an iterative eigensolver a vector which already provides a good approximation, it is plausible to expect that the number of iterations of the solver is reduced and the whole process speeds-up.

Since we are provided with as many approximate vectors as the dimension of the sought after eigenspace, the iterative eigensolver of choice should be able to simultaneously accept multiple vectors as input. Such a feature is in part provided by the class of block iterative eigensolvers [9, 16, 19, 29, 31, 32, 28], among which one has to select the one that maximize the number of input vectors and exploit to the maximum extent the approximate guess they provide [12]. Consequently the class of block solvers is dramatically restricted to subspace iteration algorithms. While this class of algorithms has the correct properties it also known to converge at best linearly to an invariant subspace of the eigenspace [14]. By providing the subspace iteration with a polynomial pre-conditioner the convergence can be improved substantially [26]. Among the many choices of polynomial accelerators, we singled out the Chebyshev polynomials for their optimal enhancing of the sought after subspace. The end product is a Chebyshev Filtered Subspace Iteration method (ChFSI) which we are going to describe in the next section.

3 The Chebyshev Filtered Subspace Iteration Method

As mentioned in the previous section, when the search for the solutions to an eigenproblem requires the handling of approximate eigenvectors, iterative eigensolvers are the obvious choice. Such solvers attempt to build an invariant subspace of the eigenspace by repeating in loop-wise fashion a series of steps. At the end of each loop (or iteration) a subspace, which provides a better guess to the invariant eigenspace than the previous one, is constructed. Iterative eigensolvers can be distinguished in two major categories: those for which the dimension of the constructed subspace changes at the end of each iteration, and those for which it is maintained fixed. In the first case the subspace is built out of a single initial vector and the subspace grows at each iteration until some restart mechanism resizes it. Krylov- and Davidson-like eigensolvers are all included in this category. Eigensolvers in the second category have the subspace spanned by a number of vectors equal (or slightly larger) to the size of the required spectrum. Typical examples of solvers in this category are subspace iteration methods.

Block eigensolvers constitute a sort of hybrid group which attempt to combine the robustness of dimension-changing methods with the compactness of dimension-invariant ones. A prototypical example of these methods is a Krylov-like method for which the subspace is augmented with small blocks of mutually orthogonal vectors. Block methods are more versatile since they can accept multiple input vectors to initialize the iterative process. Unfortunately they can also be more unstable due to the very process of subspace building which can lead to rank-deficient subspaces. For this reason the size of the block is never too high and very rarely is larger than 20. Due to these issues standard block solvers are unsuitable to be used when the number of input vectors is already of the order of several dozens. The only alternative is to use subspace iteration methods which, by definition, are maximal-size block methods.

Besides accepting multiple input vectors, block methods, and especially subspace iteration, offer a series of additional advantages when used on dense eigenvalue problems. In the first place they can solve simultaneously for a portion of the sought after eigenspace. In particular subspace iteration avoids stalling when dealing with relatively small clusters of eigenvalues. In the second place, block methods rely on matrix-matrix rather than matrix-vector multiplication kernels. When dealing with dense matrices the obvious choice for these multiplications is the highly optimized `GEMM` routine included in the BLAS 3 library. Especially in the case of subspace iteration the use of `GEMM` permits to reach close-to-peak performance of the processing unit. This is one of the key characteristics that makes ChFSI a very efficient and easy to scale algorithm.

3.1 The sequential algorithm

Subspace iteration is probably one of the earliest iterative algorithms to be used as numerical eigensolver [30, 18]. Subspace Iteration complemented with a Chebyshev polynomial filter is a well known algorithm in the literature [26, 23, 27]. A version of it was recently developed and implemented by Zhou *et al.*[1] for a real space discretization of DFT [34, 35] and included in the PARSEC code [21]. The ChFSI algorithm here presented takes inspiration from this latest implementation and includes some additional features: 1) it introduces an internal loop that iterates over the polynomial filter and the Rayleigh quotient, 2) it adds a locking mechanism to the internal loop, and 3) most importantly, it optimizes the degree of the polynomial filter so as to minimize the number of matrix-vector operations. In addition to these supplementary features, ChFSI is specifically tailored for sequences of correlated eigenproblems. In fact the algorithm can be used to solve an isolated eigenvalue problem but thrives when it is inputted with knowledge extracted from the sequence. The ChFSI pseudocode is illustrated in **Algorithm 1**. Notice that the initial input is not the initial $P^{(\ell)}$ but its reduction to standard form $H^{(\ell)} = L^{-1}A^{(\ell)}L^{-\mathrm{T}}$ where $B^{(\ell)} = LL^{\mathrm{T}}$, and $Y^{(\ell-1)}$ are the eigenvectors of $H^{(\ell-1)}$.

It is a known fact that any implementation based on subspace iteration converges linearly at best. By using a polynomial filter on the initial block of inputted vectors the method experiences a high rate of acceleration. In order to implement an efficient filter, ChFSI uses few Lanczos iterations (`line 3`) so as to estimate the upper limit of the eigenproblem spectrum [33]. This estimate constitutes the upper bound of the interval whose eigenvectors are suppressed by the filter. The lower bound is given by the inputted $\lambda^{(\ell-1)}_{\mathrm{NEV}+1}$ of the previous iteration cycle. These two values are necessary for the correct usage of the filter based on Chebyshev polynomials [27].

Algorithm 1 Chebyshev Filtered Subspace Iteration

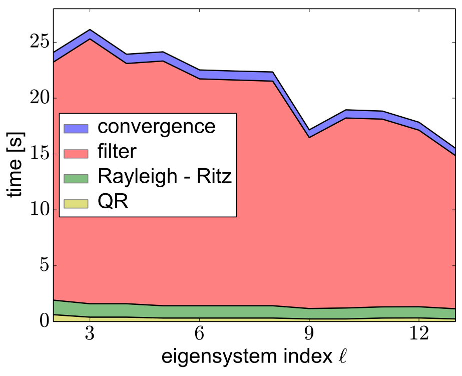
```
Input: Matrix $H^{(\ell)}$, approximate eigenvectors $Y^{(\ell-1)}$ and eigenvalues $\lambda_1^{(\ell-1)}$ and $\lambda^{(\ell-1)}_{\mathrm{NEV}+1}$. Starting
    degree DEG for the filter.
Output: Wanted eigenpairs $\left(\Lambda^{(\ell)}, Y^{(\ell)}\right)$.

 1 Set $\left(\Lambda^{(\ell)}, Y^{(\ell)}\right)$ to be the empty array/matrix.
 2 Set the degrees $(m_1, \dots, m_{\mathrm{NEV}}) = (\mathrm{DEG}, \dots, \mathrm{DEG}) =: m$ for the filter.
 3 Estimate the largest eigenvalue.                                  ▷ LANCZOS
 4 repeat
 5     Filter the vectors $Y^{(\ell-1)} = C_m\left(Y^{(\ell-1)}\right)$.               ▷ CHEBYSHEV FILTER
 6     Orthonormalize $\left[Y^{(\ell)}\, Y^{(\ell-1)}\right]$.                          ▷ QR
 7     Compute Rayleigh quotient $G = Y^{(\ell-1)\dagger} H^{(\ell)} Y^{(\ell-1)}$.    ▷ RAYLEIGH-RITZ (Start)
 8     Solve the reduced problem $GW = W\Lambda^{(\ell-1)}$.
 9     Compute $Y^{(\ell-1)} = Y^{(\ell-1)} W$.                           ▷ RAYLEIGH-RITZ (End)
10     Lock converged eigenpairs into $\left(\Lambda^{(\ell)}, Y^{(\ell)}\right)$.           ▷ LOCKING
11     Compute the degrees $(m_1, \dots, m_k) = m$ for the filter.        ▷ OPTIMIZATION
12 until converged $\geq$ NEV
```

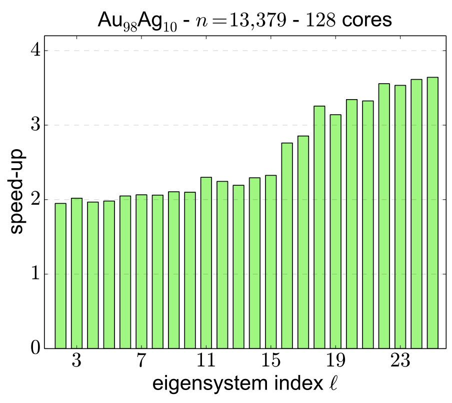
[1]The authors refer to this algorithm as CheFSI not to be confused with our ChFSI algorithm.

After the Chebyshev filter step (`line 5`) the block of vectors spanning the invariant subspace could easily become linearly dependent. In order to avoid such an occurrence each iteration is usually complemented with some orthogonalization procedure which, in ChFSI, is established with a Householder reflectors based QR factorization (`line 6`). The Q vectors of the factorization are then used to compute the Rayleigh-Ritz quotient of the matrix $H^{(\ell)}$ (`line 7`). Such a quotient represents a projection of the eigenproblem onto a subspace approximating the sought after eigenspace. The resulting reduced eigenproblem is then diagonalized and the computed eigenvectors are projected back to the larger problem (`line 9`). At the end of the Rayleigh-Ritz step eigenvector residuals are computed, converged eigenpairs are locked and the non-converged vectors are set to be filtered again. For each non-converged vector, an optimized degree of the polynomial filter is computed using its corresponding residual and approximate eigenvalue (`line 11`).

As for any iterative eigensolver, it is not feasible to predict how many loops ChFSI requires to find all desired eigenpairs. Consequently it is not possible to a priori establish the full computational cost of the eigensolver. In its stead it is feasible to evaluate the cost of each step within a single loop. Let's express with m the degree of the polynomial, n the size of the eigenproblem and k the number of filtered vectors. Then the complexity of the CHEBYSHEV FILTER is $O(mn^2k)$, the QR factorization accounts for $O(nk^2)$, the whole RAYLEIGH-RITZ procedure amounts to $O(n^2k + nk^2 + k^3)$, and the residuals check in the LOCKING step is $O(n^2k)$ [2]. Since $m \gg 1$, the filter makes up for the highest fraction of computing time. Moreover more loops, and so more filtering steps, will be needed to solve for problems at the beginning of the sequence, making the filter complexity even more dominant there. In plot (a) of Fig. 2 we show how this heuristic analysis is confirmed by numerical measurements[3].



(a) EleChFSI steps timing profile.



(b) Random vs Approximate vectors.

Figure 2: EleChFSI behaviour as a function of the iteration (eigenproblem) index ℓ. In plot (a) it is depicted the timing profile for each of the steps. Plot (b) shows the speed-up of EleChFSI when inputted approximate solutions as opposed to random vectors.

Many of the most expensive steps of the algorithms can leverage the `GEMM` kernel of the BLAS 3 library. In the Rayleigh-Ritz, `GEMM` can be invoked for both the computation of the quotient and

[2]The LANCZOS and OPTIMIZATION steps are at most $O(n^2)$ and $O(k)$ respectively and are consequently negligible.
[3]Data in the plot refers to the parallelized version of ChFSI which is discussed in a later section.

the back-transformation of the Ritz vectors. The computation of the vectors residuals can also profit from BLAS 3 calls. As we will see in the next section, the filter is the part of the algorithm which benefits by far the most from the usage of this dense linear algebra kernel.

Algorithm 2 Chebyshev Filter

Input: Matrix $H \in \mathbb{C}^{n\times n}$, vectors $Y_0 \in \mathbb{C}^{n\times k}$ sorted according to the ascending degree specification $m = (m_1, \ldots, m_k) \in \mathbb{N}^k$ and parameters $\lambda_1, c, e \in \mathbb{R}$.

Output: Filtered vectors Y_m, where each vector $Y_{m,j}$ is filtered with a Chebyshev polynomial of degree m_j.

```
H $\leftarrow (H - cI_N)/e$
$\sigma_1 \leftarrow e/(\lambda_1 - c)$
$Y_1 \leftarrow \sigma_1 H Y_0$
s $\leftarrow \mathrm{argmin}_{j=1,\ldots,k}\, m_j \neq 1$
**for** $i = 1, \ldots, m_k - 1$ **do**
    $\sigma_{i+1} \leftarrow 1/(2/\sigma_1 - \sigma_i)$
    $Y_{i+1,1:\mathtt{s}-1} \leftarrow Y_{i,1:\mathtt{s}-1}$
    $Y_{i+1,\mathtt{s}:n} \leftarrow 2\sigma_{i+1} H Y_{i,\mathtt{s}:n} - \sigma_{i+1}\sigma_i Y_{i,\mathtt{s}:n}$
    s $\leftarrow \mathrm{argmin}_{j=\mathtt{s},\ldots,k}\, m_j \neq i+1$
**end for**
```

3.1.1 The Chebyshev filter

Since the Chebyshev filter is heavily based on the use of the homonymous polynomials, let us recall briefly their definition and properties.

Definition 3.1. *The Chebyshev Polynomials $C_m(t)$ of degree m can be defined on the entire real axis as*

$$C_m(t) = \cosh\left(m \cosh^{-1}(t)\right), \quad t \in \mathbb{R}. \tag{3}$$

When the domain of definition is restricted on the interval $[-1, 1]$ *their expression simplifies to*

$$C_m(t) = \cos\left(m \cos^{-1}(t)\right), \quad |t| \le 1.$$

Despite the definition hides their polynomial nature, it can be easily shown that $C_m(t)$ oscillates between the values -1 and 1 in the interval $t \in [-1, 1]$ and diverge monotonically taking values

$$C_m(t) = \frac{|\rho|^m + |\rho|^{-m}}{2}, \qquad |\rho| = \max_{|t|>1} \left| t \pm \sqrt{t^2 - 1} \right|.$$

Additionally, the combination $p_m(t) = \left.\frac{C_m(t)}{C_m(\gamma)}\right|_{|\gamma|\ge 1}$ is shown to satisfy an extremum theorem (See Th. 4.8 in [27]). The theorem together with the asymptotical behaviour of $C_m(t)$ determine the convergence properties of $p_m(A)$ when it applies to a generic vector y. Assume $A \in \mathbb{C}^{n\times n}$ is Hermitian,

(λ_i, w_i) are its eigenpairs in ascending order and that the interval $[-1,1]$ is mapped to an interval $[\alpha,\beta] \ni \{\lambda_2,\dots,\lambda_n\}$, it is then straightforward to show that

$$p_m(A)y = \sum_{i=1}^{n} s_i p_m(\lambda_i) w_i \approx s_1 w_1 + \sum_{i=2}^{n} s_i \frac{1}{|\rho_1|^m} w_i,$$

which converges to w_1 with ratio

$$\tau(\lambda_1) = |\rho_1|^{-1} = \min_{\lambda_1 \notin [\alpha,\beta]} \left\{ \left|\lambda_{1,c,e} + \sqrt{\lambda_{1,c,e}^2 - 1}\right|, \left|\lambda_{1,c,e} - \sqrt{\lambda_{1,c,e}^2 - 1}\right| \right\}, \tag{4}$$

where $c = \frac{\alpha+\beta}{2}$ and $e = \frac{\beta-\alpha}{2}$ are respectively the center and the half-width of the interval $[\alpha,\beta]$ and $\lambda_{1,c,e} = \frac{\lambda_1 - c}{e}$. The further is the eigenvalue λ_1 from the interval, the smaller is $\tau(\lambda_1)$ and so the faster is the convergence to w_1.

Algorithm 2 generalizes the mechanism just described above to the case of an arbitrary number k of vectors Y approximating the eigenvectors corresponding to the values $\{\lambda_1,\dots,\lambda_k\} \notin [\alpha,\beta]$. The values for β and α are respectively estimated by the LANCZOS step of **Algorithm 1** and by $\lambda_{\text{NEV}+1}^{(\ell-1)}$[4], while $\lambda_1^{(\ell-1)}$ gives an estimate for γ. The actual polynomial $p_m(t)$ is not computed explicitly. What is calculated is its action on the vectors Y by exploiting the 3-terms recurrence relation which can be also used to define Chebyshev polynomials

$$C_{m+1}(Y) = 2HC_m(Y) - C_{m-1}(Y), \qquad C_m(Y) \equiv C_m(H)\cdot Y. \tag{5}$$

Eq. (5) generalizes easily to $p_m(Y)$ as shown in `line 8` of **Algorithm 2**.

The choice of polynomial degree deserves a discussion on its own. Although the ratio of convergence for each vector depends on the corresponding eigenvalues as in (4), we can calculate a reasonably accurate estimate of it. This claim is based on two observations. First we start already with an approximate eigenspectrum which is rapidly refined at each additional loop. Second, it is known that the eigenvalues convergence goes as the square of the convergence of their corresponding eigenvectors. These two facts justifies the use of the values $\tau(\hat{\lambda}_i) = |\hat{\rho}_i|^{-1}$, computed with approximate eigenvalues $\hat{\lambda}_i$ after the first loop, as already good estimates for the exact ratios $|\rho_i|^{-1}$. As shown in [11], estimates for the ratios can be used in modeling the behaviour of the filtered vectors residuals

$$\begin{aligned} \text{Res}(Y_{m,1}) &\doteq \frac{\|HY_{m,1} - \hat{\lambda}_1 Y_{m,1}\|}{\|Y_{m,1}\|} \lesssim \frac{|\text{const.}|}{|\hat{\rho}_1|^m} \\ \text{Res}(Y_{m,i}) &\doteq \frac{\|HY_{m,i} - \hat{\lambda}_i Y_{m,i}\|}{\|Y_{m,i}\|} \lesssim \frac{|\text{const.}|}{|\hat{\rho}_i|^m} + |\text{const.}| \frac{|\hat{\rho}_1|^m}{|\hat{\rho}_i|^m} \cdot \varepsilon, \qquad 2 \le i \le k, \end{aligned} \tag{6}$$

where with ε we refer to a value which could be as small as the machine-epsilon.

Eq. (6) portraits two regimes for the eigenpair residuals with $i \ge 2$: a converging regime for small to moderate values of m, and a diverging regime for moderate to large values of m. In the converging

[4]This is only true for the first loop of ChFSI. After that it is given by the approximate value $\hat{\lambda}_{\text{NEV}+1}^{(\ell)}$ given by the previous loop.

regime the first term dominates by monotonically decreasing the residuals to lower values. The diverging regimes kicks in when the ratio $\frac{|\hat{\rho}_1|^m}{|\hat{\rho}_i|^m} \gg 1$ and, counterbalancing the machine precision, contributes to the residual with a value comparable to the first term. At this point the second term becomes dominant and increases monotonically the value of the residuals.

When we are in the converging regime and the initial residual of each input vector Y_i is known, Eq. (6) can be used to compute an upper bound for the degree of the polynomial m_i needed to compute eigenpairs having residual lower than a required tolerance $\text{Res}(Y_{m_i,i}) \leq \text{TOL}$. Reasonable values for $|\hat{\rho}_i|$ can be computed at the end of the first loop with a fairly low polynomial degree $m_0 < 10$. Our experience showed that even with such a low degree the computed ratios were fairly accurate. Then by imposing

$$\text{TOL} \geq \text{Res}(Y_{(m_i+m_0),i}) \approx \text{Res}(Y_{m_0,i}) \frac{|\hat{\rho}_i|^{m_0}}{|\hat{\rho}_i|^{m_i+m_0}}$$

we arrive at the following inequality for the polynomial degree which is used to filter the non-converged vectors at the next loop

$$m_i \gtrsim \frac{\ln \left| \frac{\text{Res}(Y_{m_0,i})}{\text{TOL}} \right|}{\ln \|\hat{\rho}_i\|}. \tag{7}$$

Clearly to be effective the computed m_is have to be within the limits of the converging regime which, in our experience, appears always to be the case for values of m_i lower than 40.

As we have seen at the beginning of this section, the complexity of the filter is linearly proportional to the degree of the polynomial used. Consequently minimizing such a degree boils down to lower the number of matrix-vector operations to a minimum. We refer to such a minimization procedure as *Single Optimization*. Another layer of optimization is accessible when the whole eigenproblem sequence is taken into consideration. We refer to such second layer as *Multiple Optimization*, and direct the reader to [11] for further analysis and discussions on both optimization schemes.

Practically all operations in the filter are carried on using the BLAS 3 subroutine `GEMM`. Since `GEMM` is universally recognized as the most performant routine available (in some cases reaching up to 95% of theoretical peak performance), the filter is not only the most computationally intensive part of the algorithm ((see plot (a) in Fig. 2) but also the most efficient and potentially a very scalable one.

3.2 Parallelizing ChFSI

The algorithm can be efficiently implemented for both shared and distributed memory architectures. Here, we report only the latter. The interested reader can find details on an initial [5] OpenMP based shared memory implementation in [4, 12]. The parallel distributed memory version is implemented, using the Message Passing Interface (MPI), on top of Elemental, a (relatively new) distributed memory dense linear algebra library [25]. The parallel algorithm was consequently called EleChFSI. In the following we first comment on the design of Elemental, and then address its usage for the implementation of EleChFSI.

[5] Lacks the degree optimization.

Elemental — The library is implemented using C++ and provides routines for both sequential and parallel dense linear algebra relying heavily on BLAS and LAPACK. It exploits the object-oriented nature of C++ providing the `Matrix` class, which internally "hides" the details about the matrix datatype (functions and operators are overloaded), size, leading dimension and so on. The net effect is to lift the user from the burden of passing all those attributes to internal routines as it is customary in BLAS and LAPACK libraries. In fact, the provided `blas` and `lapack` namespaces represent an easy-to-use, datatype-independent wrapper over a collection of the most used BLAS and LAPACK routines (others can be called directly on the buffer where the matrix is stored, if needed).

The parallel analogue of the `Matrix` class is the `DistMatrix` class which additionally internally handles distribution details such as alignments and grid information. The essential difference in the design of Elemental compared to ScaLAPACK/PLAPACK is that Elemental performs all computations using element-wise ("elemental" or "torus-wrap") matrix distributions. Among the many different distributions supported, we used the two dimensional cyclic element-wise one. Here, two dimensional means that the MPI processes involved in the computation are logically viewed as a two-dimensional $r \times c$ process grid where $p = r \cdot c$ is the total number of processors. Each processor is labeled with two indices. A matrix $A = [a_{ij}] \in \mathbb{F}^{n \times m}$ is distributed over the grid in such a way that the process (s,t) owns the matrix

$$A_{s,t} = \begin{pmatrix} a_{\gamma,\delta} & a_{\gamma,\delta+c} & \cdots \\ a_{\gamma+r,\delta} & a_{\gamma+r,\delta+c} & \cdots \\ \vdots & \vdots & \end{pmatrix},$$

where $\gamma \equiv (s+\sigma_r) \pmod r$ and $\delta \equiv (t+\sigma_c) \pmod c$, and σ_r and σ_c are arbitrarily chosen alignment parameters. We will hereafter refer to this distribution as the default one.

The library provides `View` and `Partition` mechanisms to allow the user to work with sub-matrices. The routines work purely on a pointer basis and automatically handle the distribution details eliminating the need of allocating additional memory space where to copy the data. The communication for the computations is performed almost entirely in terms of collective communication within rows and columns of the process grid. Elemental is a constantly growing, modern alternative to ScaLAPACK/PLAPACK which not rarely outperforms the two precursors.

EleChFSI — The Hamiltonian, as well as the approximate eigenvectors are distributed over the two dimensional grid in the default manner. For a fixed number $p > 1$ of processors there are several possible choices for r and c forming different grid shapes (r,c). The grid shape can have a significant impact on the overall performance, and careful experiments need to be undertaken in order to determine the best choice. Another parameter which affects performance is the algorithmic block size, which is correlated to the square root of the L2 cache [15]. In practice, the size of the algorithmic block not only depends on the algorithm itself, but it is also affected by the architecture and the used compilers. In Figure 3 we show the performance of EleChFSI with respect to different grid shapes and different algorithmic block sizes. Since we are solving for a small fraction of the eigenspectrum, the matrix of vectors $Y^{(\ell)}$ is tall and skinny and we expect a narrow rectangular grid shape to do a better job than a square and wider one. This is confirmed by the tests. Furthermore, the Figure shows that for EleChFSI a block size of 256 is always recommended independently of the number of cores or grid shape. This effect is imputable to the large number of matrix multiplications

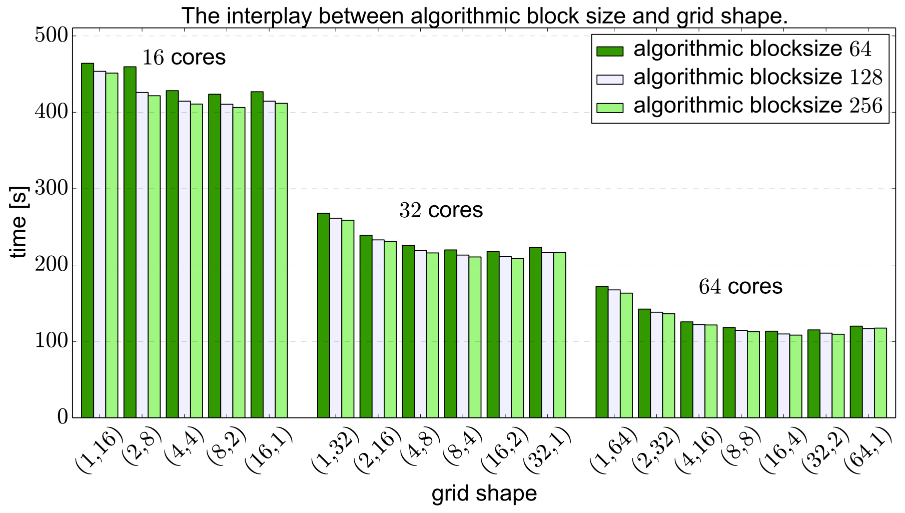


Figure 3: The data in this plot refer to a GHEVP of $\ell = 20$, size $n = 13,379$, and number of sought after eigenpairs $\texttt{NEV} = 972$, corresponding to the physical system $Au_{98}Ag_{10}$. The eigenproblem was repeatedly solved with EleChFSI using 16, 32, and 64 cores, all possible grid shapes (r,c) and three distinct algorithmic block sizes.

carried on by the filter.

The core of EleChFSI is **Algorithm 2**. In the very first iteration $m_1 = m_2 = \ldots = m_{\text{NEV}}$ and the filter reduces to the "conventional" one, where each vector is filtered with the same degree $\texttt{DEG} \equiv m_0$. Due to the introduction of *Single Optimization*, in subsequent iterations these degrees may differ and, in order to exploit `GEMM` maximally, the vectors are sorted according to their increasing polynomial degree. The sorting allows us to invoke `GEMM` m_1 times on the whole block of vectors Y_0, then additional $m_2 - m_1$ times on the right block of Y_{m_1}, excluding the vectors with degree m_1 which have been already filtered. This process continues until we filter only the last vector m_k times, as it is illustrated in **Algorithm 2**.

The reduced eigenproblem in the Rayleigh-Ritz step is solved using a parallel implementation of the MRRR eigensolver – EleMRRR [24] – which is also integrated in Elemental. Since the evaluation of the degrees m_i is based on a heuristic model and some approximate eigenvectors can be better approximations than others, it is not always the case that a smaller eigenpair will converge before a bigger one. Therefore, we bring the converged eigenvectors to the leftmost part of the matrix Y and leave the non-converged in the remaining rightmost part. A `View` of Y then makes sure we work only with the part we are interested in.

4 Numerical tests and discussion

The set of numerical tests presented in this section were performed on eigenproblems extracted from the simulation of four distinct physical systems. The simulations were carried out using the FLEUR code [8] implementing the FLAPW method. Each of the physical systems generated four different eigenproblem sequences of distinct length N and size n. The data of the sequences of eigenproblems is summarized in Table 1.

By including both conductors and insulators, the physical systems represent a heterogeneous sample with distinct physical properties. For example the two TiO_2 compounds are built using respectively $3 \times 3 \times 3$ supercell with 161 atoms, and a $4 \times 4 \times 4$ supercell with 383 atoms, with a single oxygen defect. The valence electrons are formed by the Oxygen $2s$ and $2p$ electrons and the Titanium $3p$, $4s$ and $3d$ electrons resulting in a total valence charge of $1,182$ electrons for the small and $2,816$ electrons for the large system. The functions basis set contained $12,293$ augmented plane waves (APW) and additionally 162 local orbitals (LO), which were increased to $29,144$ APW + 384 LO for the larger system.

Table 1: Simulation data

Material	N	n
$Na_{15}Cl_{14}Li$	13	9,273
TiO_2	30	12,455
$Au_{98}Ag_{10}$	25	13,379
TiO_2	18	29,528

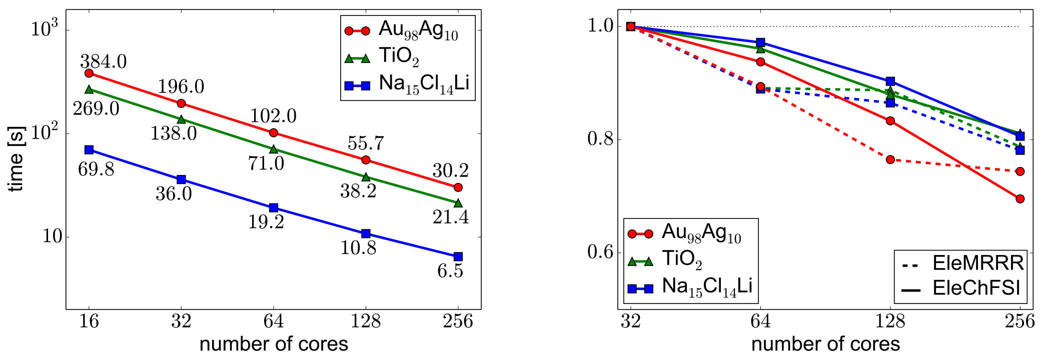
All our numerical tests were performed on JUROPA, a large general purpose cluster where each node is equipped with 2 Intel Xeon X5570 (Nehalem-EP) quad-core processors at 2.93 GHz and 24 GB memory (DDR3, 1066 MHz). The nodes are connected by an Infiniband QDR network with a Fat-tree topology. The tested routines were compiled using the GCC compilers (ver. 4.8.2) and linked to the ParTec's ParaStation MPI library (ver. 5.0.28-1). The Elemental library (ver. 0.84-dev) was used in conjunction with Intel's MKL BLAS and LAPACK (ver 11.0).

Statistically significant measurements were obtained by running each numerical test multiple times and taking the average of the recorded CPU times. In order to probe EleChFSI at different level of accuracy, eigenproblems were solved setting the minimum tolerance TOL for the eigenpair residuals to either 10^{-08} or 10^{-10}. As already mentioned in Sec. 3.2 the choice of processors grid shape and algorithmic block size is crucial to obtain the best performance from the Elemental framework. Figure 3 illustrate that the optimal choices of grid shape and algorithmic block size – for $p = 2^q$ processor on the JUROPA architecture – are respectively $(2^{q-2}, 4)$ and 256. All the tests were carried out using exclusively these values.

4.1 Parallel performance and scalability

In this subsection we momentarily abstract from the eigenproblem sequence and test the scalability and parallel efficiency of the eigensolver for problems at fixed ℓ. Tests were conducted on eigenproblems extracted from sequences arising from the first three chemically different physical systems listed in Table 1.

In our first test we fix the size of the eigenproblem and solve it using an increasing number of cores effectively distributing the computational load over more processes by decreasing the ratio of data per process. Such test measures the strong scalability of the eigensolver and its outcome



(a) Strong scalability.

(b) Parallel efficiency.

Figure 4: EleChFSI behaviour for an increasing number of cores. In plot (a) it is depicted the strong scalability where the size of the eigenproblems are kept fixed while the number of cores is progressively increased. Plot (b) shows the parallel efficiency of EleChFSI. In both plots the green lines refer to the smaller among the TiO_2 systems.

is shown in plot (a) of Fig. 4. Here CPU times are plotted on the y-axis using a logarithmic scale. Each line corresponds to an eigenproblem of specific size n which corresponds to a distinct physical system and need to be solved for a different percentage of the eigenspectrum. In particular $Na_{15}Cl_{14}Li$, TiO_2 and $Au_{98}Ag_{10}$ require respectively $\text{NEV} = 256$ ($\sim 2.8\%$), 648 ($\sim 5.2\%$), and 972 ($\sim 7.2\%$). Despite the eigenproblem properties and spectrum distribution are quite dissimilar they all have very similar scaling ratios showing how the scalability of EleChFSI is not impacted by the nature of the problem. Moreover EleChFSI is extremely efficient even when the ratio of data per process, at the rightmost end of the x-axis, is getting quite low and far from optimal.

In our second test we compute the parallel efficiency of EleChFSI and compare it to the efficiency of a dense linear algebra (direct) eigensolver. Parallel efficiency is a measure of the loss of efficiency of an algorithm as the number of cores increases and is defined as

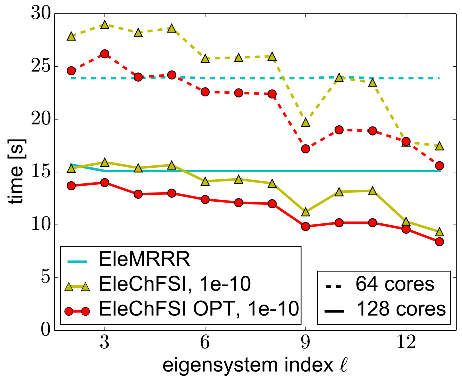
$$\eta_{\text{strong}} \doteq \frac{t_{\text{ref}} \cdot p_{\text{ref}}}{t_p \cdot p},$$

where t and p indicate the CPU time and the number of processors. In plot (b) of Fig. 4 the parallel efficiency η_{strong} of EleChFSI is graphed for the same three physical systems of plot (a) with the addition of three other dashed curves corresponding to the parallel efficiency of the direct solver EleMRRR solving for the same problems. This plot clearly shows that our iterative eigensolver has an overall higher parallel efficiency than the dense counterpart. Such conclusion can be interpreted on the basis of the following observations. While EleChFSI performs more floating point operations (*flops*) per data, it also relies heavily on the parallel BLAS 3 GEMM routine implemented in Elemental which can reach a performance close to the theoretical peak of the CPUs. EleMRRR, as any comparable direct eigensolver, performs a smaller number of flops per data but executes a great portion of the computation – i.e. the reduction to tridiagonal form – by leveraging only BLAS 2 kernels which are quite less performant than GEMM.

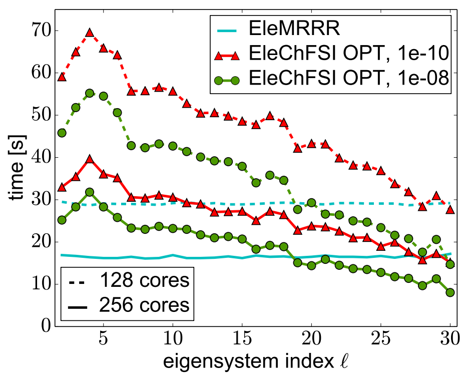
Currently most FLAPW-based codes still rely on the use of ScaLAPACK eigensolvers. As shown in [24] ScaLAPACK PZHEEVX (Bisection and Inverse Iteration) and PZHEEVD (Divide & Conquer) routines are less efficient than EleMRRR for a number of cores higher than respectively 128 and 256 while PZHEEVR (MRRR) efficiency drops starting at 512 cores. These routines are used to solve for $\mathcal{N}_k$ eigenproblems per SCF using few hundreds of cores per problem with residual tolerance close to machine precision. So in the best case scenario each SCF can use efficiently up to $\mathcal{N}_k \times 512$ cores.

The parallel efficiency plot of Fig. 4 makes manifest EleChFSI's potential to scale efficiently on larger number of cores than the ScaLAPACK routines. The slow decrease of the three solid lines ensure that, for eigenproblems of size $n > 10 - 15,000$, EleChFSI can do a better job in exploiting large distributed architectures. In other words EleChFSI has the potential of allowing the users of FLAPW-based codes to generate more complex physical systems made of thousands of atoms resulting in eigenproblems of sizes reaching up to and over 100,000. Additionally the ability of EleChFSI to obtain solutions with a lower accuracy without affecting the convergence of the sequences implies further savings in computational time.

4.2 Performance on sequences of eigenproblems



(a) $Na_{15}Cl_{14}Li$ - $n = 9,273$ - NEV $= 256$.



(b) TiO_2 - $n = 12,455$ - NEV $= 648$.

Figure 5: Comparing EleChFSI with EleMRRR on eigenproblems of increasing self-consistent cycle index ℓ. For the size of eigenproblems here tested the ScaLAPACK implementation of BXINV is comparable with EleMRRR [24]. For this reason a direct comparison with the BXINV solver is not included.

In this subsection we present a set of numerical tests meant to illustrate the performance of EleChFSI as a function of the sequence index ℓ. The foremost objective is to show how the algorithm harnesses the correlation between adjacent eigenproblems. For this purpose we have calculated the speed-up as the fraction of CPU times EleChFSI requires for computing the solutions when it uses initial random vectors as opposed to when it is inputted approximate solutions. Plot (b) of Fig. 2 unequivocally shows EleChFSI obtains a great advantage from the use of the eigenvectors $Y^{(\ell-1)}$ as input in solving the next eigenproblem $H^{(\ell)}$ in the sequence. Already at the beginning of the sequence the algorithm experiences speed-ups higher that 2X and well above 3X towards the end of it. From the analysis of similar plots for other sequences we observed this behaviour is stable with respect to small changes of the sequence correlation and, as such, does not depend on the physical system or spectral properties of the eigenproblems. In other words taking advantage of the sequence correlation is a universal strategy which can be systematically implemented.

Compared to direct solvers, EleChFSI promises to be fairly competitive. Depending on the number of eigenpairs computed, our algorithm is on par or even faster than EleMRRR. In plot (a) of Fig. 5 the non-optimized EleChFSI appears to become competitive with the direct solver only later in the sequence when using just 64 cores. The situation improves substantially with 128 cores, and at the beginning of the sequence both algorithms are on par. This response can be easily explained by observing the higher parallel efficiency of EleChFSI with respect to EleMRRR efficiency in plot (b) of Fig. 4. The second set of lines in plot (a) of Fig. 5 shows the execution time of the single optimized code. EleChFSI OPT saves an extra $15-20\%$ of CPU time and becomes, in absolute terms, the faster algorithm outperforming EleMRRR quite dramatically at the end of the sequence. From this point of view the *Single Optimization* gives EleChFSI that extra edge which makes the out-performance of dense linear algebra solver possible in this particular case.

Clearly the above conclusions are not universal and depend on the fraction of eigenspectrum desired. In plot (b) of Fig. 5 we observe a different situation where even the singly optimized EleChFSI does not outperform as easily the direct eigensolver. In this case one can take advantage of the flexibility of the algorithm and, if the nature of the application allows it, require a looser stopping criterion for the eigenpair residuals. In DFT it is often the case that a threshold of 10^{-08} gives already eigenvectors which are accurate enough for the purpose of the simulation. With this information in hand EleChFSI remains still competitive at least for the second half of the sequence.

In order to illustrate the complex dependence of EleChFSI on the minimum tolerance for the eigenpair residuals, the eigenspectrum fraction, and the sequence index ℓ, we have represented in Fig. 6 its performance for different values of these parameters. From this figure it is evident that depending on the fraction of the spectrum and tolerance desired, EleChFSI can be competitive for the whole sequence or just part of it. Moreover it is also evident how the spectrum distribution plays a significant role. In fact in going from 4.5% to 7% of eigenpairs the EleChFSI curves become less steep. This counterintuitive effect is imputable to the presence of gaps in the spectrum distribution closer to the end of the filtered interval which the Chebyshev filter is capable to exploit effectively. It is important to notice that all the above conclusions are truly independent of the size of the eigenproblem sequence as it is demonstrated by the relatively large eigenproblems ($n \sim 10-30,000$) such as the one used in Fig. 6.

5 Summary and conclusions

In this paper we extend the presentation [5] of the Chebyshev Filtered Subspace Iteration (ChFSI) algorithm especially tailored for sequences of correlated eigenproblems as they arise, for example, in FLAPW-flavoured DFT methods. In particular we have illustrated an optimized version of the algorithm and its parallelization for distributed memory architecture. The paper introduces the concept of sequence of eigenproblems and how the idea of correlation adds critical information to the sequence. We show typical examples of eigenproblem sequences as they arise in materials

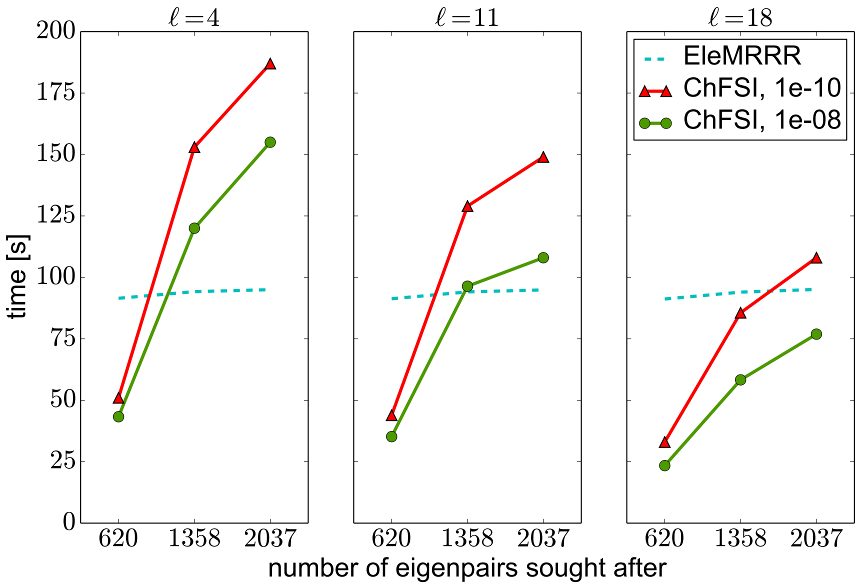


Figure 6: The data in this plot refer to a sequence of eigenproblems at $\ell = 4,\ 11,\ 18$ of the physical system TiO_2, with size $n = 29,528$, and a variable number of sought after eigenpairs $\text{NEV} = 620,\ 1358,\ 2037$, corresponding respectively to $\sim 2\%, \sim 4.5\%$ and $\sim 7\%$ of the spectrum. In this plot the 3 eigenproblems are solved using EleMRRR and EleChFSI for two distinct threshold tolerances, namely 10^{-08} and 10^{-10}.

science and quantum chemistry and illustrate the mathematical model from which they emerge.

We then describe the algorithm in its details with specific reference to the Chebyshev filter. Parallelization is realized through the use of the Elemental library framework which hides the intricacy of the matrix distribution behind a very efficient interface. Numerical tests demonstrate that the parallel implementations of ChFSI takes great advantage from the progressive collinearity of the eigenvectors along the sequence. On large parallel architectures, the algorithm also proves to be quite performant. In the specifics we showed how EleChFSI scales extremely well over a range of cores commensurate to the size of the eigenproblems. Compared to direct eigensolvers, EleChFSI is competitive with routines out of ScaLAPACK and Elemental whenever the percentage of the eigenspectrum sought after is not too high.

The results of this paper paves the road to a new strategical approach when dealing with sequences of dense correlated eigenproblems. Instead of solving each problem of the sequence in isolation, solutions of one problem can be exploited to pre-condition an accelerated subspace iteration eigensolver and speed-up its performance. This strategy, in the specific case of DFT-generated sequences, will enable the final user to access larger physical systems which are currently out of reach.

References

[1] Jose I Aliaga, Paolo Bientinesi, Davor Davidovic, Edoardo Di Napoli, Francisco D Igual, and Enrique S Quintana-Orti. Solving dense generalized eigenproblems on multi-threaded architectures. *Applied Mathematics and Computation*, 218(22):11279–11289, July 2012.

[2] E Anderson, Z Bai, C Bischof, S Blackford, J Demmel, J Dongarra, J Du Croz, A Greenbaum, S Hammerling, A McKenney, and D Sorensen. *LAPACK Users' Guide*. Third Edition. SIAM, January 1999.

[3] T Auckenthaler, V Blum, H J Bungartz, T Huckle, R Johanni, L Krämer, B Lang, H Lederer, and P R Willems. Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations. *Parallel Computing*, 37(12):783–794, December 2011.

[4] Mario Berljafa. A block iterative eigensolver optimized for sequences of correlated eigenproblems. *Proceedings 2012, Guest Student Programme on Scientific Computing*, Mathias Winkel editor, Forschungszentrum Jülich(Technical Report FZJ-JSC-IB-2012-01):95–105, November 2012.

[5] Mario Berljafa and Edoardo Di Napoli. A parallel and scalable iterative solver for sequences of dense eigenproblems arising in FLAPW. *Lecture Notes in Computer Science*, Parallel Processing and Applied Mathematics(8385):395–406, 2014.

[6] L S Blackford, J Choi, A Cleary, E D'Azevedo, J Demmel, I Dhillon, J Dongarra, S Hammarling, G Henry, A Petitet, K Stanley, D Walker, and R C Whaley. *ScaLAPACK Users' Guide*. Society for Industrial and Applied Mathematics, January 1987.

[7] P. Blaha, K. Schwarz, G. Madsen, D. Kvasnicka, and J. Luitz. Wien2k. `http://www.wien2k.at/`.

[8] S. Blügel, G. Bihlmayer, and D. Wortmann. FLEUR. `http://www.flapw.de`.

[9] Jane Cullum and W Donath. A block Lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace of large, sparse, real symmetric matrices. In *1974 IEEE Conference on Decision and Control including the 13th Symposium on Adaptive Processes*, pages 505–509. IEEE, 1974.

[10] K. Dewhurst, S. Sharma, and C. Ambrosch-Draxl. The Exciting Code. `http://exciting-code.org/`.

[11] Edoardo Di Napoli and Mario Berljafa. Optimization schemes for the chebyshev filtered subspace iteration eigensolver. *In preparation*.

[12] Edoardo Di Napoli and Mario Berljafa. Block Iterative Eigensolvers for Sequences of Correlated Eigenvalue Problems. *Computer Physics Communications*, 184(11):2478–2488, June 2013.

[13] Edoardo Di Napoli, Stefan Blügel, and Paolo Bientinesi. Correlations in sequences of generalized eigenproblems arising in Density Functional Theory. *Computer Physics Communications*, 183(8):1674–1682, August 2012.

[14] G H Golub and C F Van Loan. *Matrix Computations*. Johns Hopkins Univ., 2012.

[15] Kazushige Goto and Robert A van de Geijn. Anatomy of high-performance matrix multiplication. *ACM Transactions on Mathematical Software*, 34(3):1–25, May 2008.

[16] Roger G Grimes, John G Lewis, and Horst D Simon. A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems. *SIAM Journal on Matrix Analysis and Applications*, 15(1):228–272, 1994.

[17] P Hohenberg. Inhomogeneous Electron Gas. *Physical Review*, 136(3B):B864–B871, November 1964.

[18] A Jennings and W J Stewart. Simultaneous iteration for partial eigensolution of real matrices. *IMA Journal of Applied Mathematics*, 15(3):351–361, 1975.

[19] A V Knyazev, M E Argentati, I Lashuk, and E E Ovtchinnikov. Block Locally Optimal Preconditioned Eigenvalue Xolvers (BLOPEX) in Hypre and PETSc. *SIAM Journal on Scientific Computing*, 29(5):2224–2239, January 2007.

[20] W Kohn and L J Sham. Self-Consistent Equations Including Exchange and Correlation Effects. *Phys.Rev.*, 140:A1133–A1138, 1965.

[21] Leeor Kronik, Adi Makmal, Murilo L Tiago, M M G Alemany, Manish Jain, Xiangyang Huang, Yousef Saad, and James R Chelikowsky. PARSEC – the pseudopotential algorithm for real-space electronic structure calculations: recent advances and novel applications to nanostructures. *physica status solidi (b)*, 243(5):1063–1079, April 2006.

[22] R B Lehoucq, D C Sorensen, and C Yang. *ARPACK Users' Guide: Solution of Large-scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. Software, Environments, Tools. SIAM, 1998.

[23] T A Manteuffel. The Tchebychev iteration for nonsymmetric linear systems. *Numerische Mathematik*, 28(3):307–327, 1977.

[24] Matthias Petschow, Elmar Peise, and Paolo Bientinesi. High-Performance Solvers for Dense Hermitian Eigenproblems. *SIAM Journal on Scientific Computing*, 35(1):C1–C22, 2013.

[25] Jack Poulson, Bryan Marker, Robert A van de Geijn, Jeff R Hammond, and Nichols A Romero. Elemental: A New Framework for Distributed Memory Dense Matrix Computations. *ACM Transactions on Mathematical Software (TOMS)*, 39(2), February 2013.

[26] H Rutishauser. Computational aspects of FL Bauer's simultaneous iteration method. *Numerische Mathematik*, 13(1):4–13, 1969.

[27] Y Saad. *Numerical methods for large eigenvalue problems*. Siam, 2011.

[28] M Sadkane and R B Sidje. Implementation of a variable block Davidson method with deflation for solving large sparse eigenproblems. *Numerical Algorithms*, 20(2-3):217–240, 1999.

[29] Andreas Stathopoulos and James R McCombs. Nearly Optimal Preconditioned Methods for Hermitian Eigenproblems Under Limited Memory. Part II: Seeking Many Eigenvalues. *SIAM Journal on Scientific Computing*, 29(5):2162–2188, January 2007.

[30] G W Stewart. Accelerating the orthogonal iteration for the eigenvectors of a Hermitian matrix. *Numerische Mathematik*, 13(4):362–376, August 1969.

[31] K S Wu and H Simon. Thick-restart Lanczos method for large symmetric eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 22:602–616.

[32] Yunkai Zhou. A block Chebyshev–Davidson method with inner–outer restart for large eigenvalue problems. *Journal of Computational Physics*, 229(24):9188–9200, December 2010.

[33] Yunkai Zhou and Ren-Cang Li. Bounding the spectrum of large Hermitian matrices. *Linear Algebra and its Applications*, 435(3):480–493, August 2011.

[34] Yunkai Zhou, Yousef Saad, Murilo L Tiago, and James R Chelikowsky. Parallel self-consistent-field calculations via Chebyshev-filtered subspace acceleration. *Physical Review E*, 74(6):066704, 2006.

[35] Yunkai Zhou, Yousef Saad, Murilo L Tiago, and James R Chelikowsky. Self-consistent-field calculations using Chebyshev-filtered subspace iteration. *Journal of Computational Physics*, 219(1):172–184, November 2006.