



# A robust, open-source implementation of the locally optimal block preconditioned conjugate gradient for large eigenvalue problems in quantum chemistry

Tommaso Nottoli<sup>1</sup> · Ivan Gianni<sup>1</sup> · Antoine Levitt<sup>2</sup> · Filippo Lipparini<sup>1</sup>

Received: 28 February 2023 / Accepted: 13 July 2023 / Published online: 30 July 2023  
© The Author(s) 2023

## Abstract

We present two open-source implementations of the locally optimal block preconditioned conjugate gradient (LOBPCG) algorithm to find a few eigenvalues and eigenvectors of large, possibly sparse matrices. We then test LOBPCG for various quantum chemistry problems, encompassing medium to large, dense to sparse, well-behaved to ill-conditioned ones, where the standard method typically used is Davidson's diagonalization. Numerical tests show that while Davidson's method remains the best choice for most applications in quantum chemistry, LOBPCG represents a competitive alternative, especially when memory is an issue, and can even outperform Davidson for ill-conditioned, non-diagonally dominant problems.

**Keywords** Eigenvalues · Davidson · LOBPCG · Full CI · CASSCF

## 1 Introduction

Computing eigenvalues and eigenvectors of matrices is probably the most prominent linear-algebra operation in quantum chemistry. Eigenvalues and eigenvectors are computed in the self-consistent field (SCF) algorithm [1–3], in restricted-step second-order optimizations [4], response theory calculations [5, 6], algebraic diagrammatic construction (ADC) [7, 8], and unitary coupled cluster (UCC) [9–11]. Moreover they are calculated in ground and excited state configuration interaction (CI) calculations [12, 13], including in state-average complete active space self-consistent field (CAS-SCF) [14–16]. Quantum chemistry calculations are typically performed using localized basis sets, most commonly made by Gaussian atomic orbitals (GTOs). Thanks to the compactness of such basis sets, the generalized eigenvalue problem in SCF can be solved using dense linear-algebra techniques.

Despite the cubic scaling, diagonalizing the Fock (or Kohn-Sham) matrix is usually not a significant bottleneck for standard calculations, which are instead dominated by the cost of assembling the Fock matrix. Iterative diagonalization techniques are therefore mostly used for post-Hartree Fock calculations, such as CASSCF and CI, where the combinatorially scaling size of the CI Hamiltonian makes it impossible to build it in memory, let alone diagonalize it, but for the smallest cases. They are also used to compute the direction and step length in second-order SCF and CASSCF strategies and to compute excited states in CI singles and time-dependent SCF. Most of these applications involve computing one, or a small number (up to a few hundreds) of eigenvalues and eigenvectors, and involve symmetric, diagonally dominant matrices.

Since its introduction in 1975, the iterative method proposed by Ernest R. Davidson has been the method of choice [17] and the default strategy used by the majority of quantum chemistry codes. Davidson's diagonalization performs particularly well for diagonally dominant symmetric eigenvalue problems, for which it exhibits fast and robust convergence, and has been generalized to non-symmetric problems and to deal with multiple eigenvalues and eigenvectors [18, 19]. However, the method suffers of a few drawbacks. First, its performance degrades if the matrix is not diagonally dominant. Even though most algorithms in quantum chemistry involve diagonally dominant matrices, this is not always the

---

Tommaso Nottoli and Ivan Gianni have contributed equally to this work.

---

✉ Filippo Lipparini  
filippo.lipparini@unipi.it

<sup>1</sup> Dipartimento di Chimica e Chimica Industriale, Università di Pisa, Via G. Moruzzi 13, Pisa 56124, Italy

<sup>2</sup> Laboratoire de Mathématiques d'Orsay, Université Paris-Saclay, Orsay 91405, France

case, with the parameter Hessian in second-order CASSCF being a prominent example. Finally, as the vector subspace used in Davidson's method to expand the sought eigenvectors can become quite large, especially if many eigenvalues are needed, the cost of diagonalizing the matrix projected in the subspace and of orthogonalizing the new trial vectors can become non-negligible, and memory requirements problematic.

The locally optimal block preconditioned conjugate gradient (LOBPCG) method offers a way out by more aggressively truncating the approximation space, while attempting to preserve the convergence properties of Davidson [20, 21]. Since its introduction, it has been successfully used in several applications, and in particular in approaches to the quantum many-body problem in condensed-matter physics [22] and [23], but to our knowledge it has not permeated quantum chemistry to the same extent.

One important caveat with LOBPCG is its potential numerical instabilities, which can degrade convergence significantly if care is not taken. This point was emphasized in [24], which describes an appropriately stable procedure to build an orthogonal basis of expansion vectors. This relies crucially on a `ortho(X, Y)` primitive, whose goal is to orthogonalize the vectors in  $X$  with respect to those in  $Y$  and among themselves. This turns out to be surprisingly hard to perform reliably in the presence of roundoff errors. The strategy suggested in [24] relies on the `ortho(X, Y)` function in [25], which is based on a modified singular value decomposition (SVD). This however can sometimes fail [26], and the use of the SVD can become expensive. For this reason [26] suggests dropping ill-conditioned directions.

In this paper, we clarify the origins of numerical instabilities, in particular due to the ill-conditioned orthogonalizations and reuse of matrix–vector products. We then present an implementation that uses exclusively Cholesky decompositions for orthogonalizations. These decompositions are known to be efficient but unstable, and for this reason the Cholesky orthogonalization has often been regarded as impractical; however, it was recently realized that appropriate stabilizations based on repeated, potentially shifted factorizations, turn it into a completely reliable orthogonalization algorithm [27]. The resulting LOBPCG algorithm is extremely stable (converging, if desired, to close to machine precision, with no degradation in convergence rate), simple (with no dropping of directions, and with a single parameter controlling the accuracy of the orthogonalizations), and efficient.

In this paper, we present two open-source implementations of the LOBPCG algorithm. One is `DIAGLIB`, an open-source, library written in Fortran 95 that also includes an implementation of Davidson's method and that can be obtained at the following address: <https://github.com/Molecolab-Pisa/diaglib>.

The library has been interfaced with the `CFour` suite of quantum chemistry programs [28, 29], which we have used to test its performance choosing a variety of test cases. The other is included in the `DFTK Julia` software package [30], supports generalized eigenvalue problems and GPUs and is available at <https://github.com/JuliaMolSim/DFTK.jl/>. These implementations are more focused on numerical stability than on sheer speed, but are overall efficient, thanks to the extensive use of highly optimized `BLAS` and `LAPACK` routines.

We then test the LOBPCG implementation on a selection of quantum chemistry problems that include full CI, second-order SCF and CASSCF calculations. For such methods, we compare the performance of LOBPCG to the ones of Davidson's method. Our preliminary results show that while Davidson diagonalization is more efficient for strongly diagonally dominant problems, as the ones encountered in full CI, LOBPCG is a viable alternative, as it exhibits similar performance and, for difficult cases such in CASSCF, can even slightly outperform Davidson.

This paper is organized as follows. In Sect. 2, we present the LOBPCG algorithm. In Sect. 3, we discuss in detail the numerical implementation of LOBPCG, analyzing the possible numerical issues and proposing cost-effective, yet robust, solutions. In Sect. 4, we test our implementation of a few quantum chemistry applications. Finally, some final remarks are given in Sect. 5.

## 2 Theory

In this section, we describe the LOBPCG algorithm in infinite precision arithmetic, without consideration for numerical stability. In this paper, we will focus on real symmetric matrices, but everything of course extends to Hermitian complex matrices. Let  $\mathbf{A} \in \mathbb{R}^{n \times n}$  be a symmetric matrix, for which we seek  $m \ll n$  eigenvalues. Given a set of vectors  $Y = (y_1, \dots, y_p) \in \mathbb{R}^{n \times p}$  with  $p \geq m$ , the Rayleigh-Ritz variational procedure obtains an approximation  $RR(Y) \in \mathbb{R}^{n \times m}$  to the first  $m$  eigenpairs by building an orthonormal basis of  $Y$ , computing the  $p \times p$  matrix representation of  $A$  on that set of vectors, diagonalizing it and taking the first  $m$  eigenvectors.

To use this procedure to obtain an iterative algorithm, a way of constructing  $Y$  must be specified. A standard method for doing so is to use the residuals, defined for a vector  $x$  by  $r(x) = \mathbf{A}x - \rho(x)x$ , where  $\rho(x) = \frac{x^T \mathbf{A} x}{x^T x}$  is the Rayleigh quotient of  $x$ . In ill-conditioned problems, these residuals might not however be a good search direction, and therefore it is useful to precondition them according to

$$w = \mathbf{T}r(x)$$

where  $T$  is a given preconditioner (for instance, when  $\mathbf{A}$  is diagonally dominant,  $\mathbf{T} = (\text{diag}(\mathbf{A}) - \rho(x)\mathbf{I})^{-1}$ ).

This choice of search direction results in the block Davidson algorithm, starting from an initial set of vectors  $X^{[0]} \in \mathbb{R}^{n \times m}$ :

$$X^{[k+1]} = \text{RR}(X^{[1]}, \mathbf{T}^{[1]}R^{[1]} \dots, \mathbf{T}^{[k]}R^{[k]})$$

where  $R^{[k]} = \mathbf{A}X^{[k]} - X^{[k]}\Lambda^{[k]}$  is the residual at the  $k$ -th iteration, and  $\Lambda^{[k]}$  is the diagonal matrix composed of the Ritz values. Since the expansion subspace is only ever enlarged, the implementation is standard: each block of vectors added to the subspace is orthogonalized against the previous vectors and against itself (although see subtleties of this operation in the next section).

Since the Davidson method performs the Rayleigh-Ritz procedure in the full convergence history, its computational requirements can increase quickly. The LOBPCG algorithm instead only keeps the last two iterates:

$$X^{[k+1]} = \text{RR}(X^{[k-1]}, X^{[k]}, T^{[k]}(AX^{[k]} - X^{[k]}\Lambda^{[k]})).$$

The method is locally optimal (LO) because the Rayleigh-Ritz procedure optimizes the Rayleigh quotient in the local expansion subspace. It is a block algorithm (B) and uses a preconditioner (P). Finally, the intuition of keeping only the previous iterate comes from the conjugate gradient (CG) algorithm for solving linear systems. This algorithm can seem like a drastic truncation of the Davidson method. There is however reason to believe that it can converge asymptotically as quickly as the full Davidson algorithm, inspired by the optimality in the Krylov space of the three-terms conjugate gradient algorithm [20].

The convergence properties of this algorithm are sensitive to the gap between eigenvalues  $m$  and  $m + 1$ , which might be small. This is particularly clear in the case of a simplified version of the LOBPCG algorithm, the block gradient descent with fixed step (which might be termed “BG,” since it is obtained by removing the locally optimal,

preconditioning and conjugate features of LOBPCG), where explicit convergence rates can be obtained easily [31]. Accordingly, as is standard, in practice one uses a block size  $m$  which is larger than the number of eigenvalues  $m_{\text{sought}}$  actually sought, and stops the algorithm as soon as the first  $m_{\text{sought}}$  eigenvalues are converged. The convergence rate is then dependent on the gap between eigenvalues  $m_{\text{sought}}$  and  $m + 1$ .

## 3 Implementation

### 3.1 The LOBPCG algorithm

When implementing the above algorithm on a computer, we face the difficulty that the basis  $(X^{[k-1]}, X^{[k]}, T^{[k]}(AX^{[k]} - X^{[k]}\Lambda^{[k]}))$  is extremely badly conditioned. This is because, as the iteration progresses,  $X^{[k]}$  becomes close to  $X^{[k-1]}$ , and the residual becomes small. Therefore, if we try to solve the Rayleigh-Ritz problem as a generalized eigenvalue problem, the results will be inaccurate. Instead, following [24], we construct systematically an orthogonal basis  $(X^{[k]}, W^{[k]}, P^{[k]})$  of the expansion subspace spanned by  $(X^{[k-1]}, X^{[k]}, T^{[k]}(AX^{[k]} - X^{[k]}\Lambda^{[k]}))$ . The  $P^{[k]}$  is implicitly constructed as the orthogonalization of  $X^{[k]}$  against  $X^{[k-1]}$ ; the  $W^{[k]}$  is constructed as the orthogonalization of  $T^{[k]}(AX^{[k]} - X^{[k]}\Lambda^{[k]})$  against  $X^{[k]}$  and  $P^{[k]}$ .

To obtain these orthogonal bases, we introduce the primitive `ortho(X, Y)` which, given a set of orthogonal vectors  $Y$ , returns an orthogonal basis of the projection of the vectors in  $X$  onto the space orthogonal to  $Y$ . In infinite precision arithmetic, this would be given by an orthogonalization of  $X - YY^T X$ ; in finite precision arithmetic, care has to be taken, as we will see in the next section. Given this primitive, the LOBPCG algorithm is given in Algorithm 1.

**Algorithm 1** LOBPCG

---

**Input:** Initial guess  $X^{[0]}$ , operator  $\mathbf{A}$ .  
**Output:**  $X, \lambda$ , the lowest eigenpairs of  $\mathbf{A}$ .  
**Intermediates:** Residuals  $R$ , preconditioned residuals  $W$ , differences  $P$ , unorthogonalized quantities (tildes).

- 1:  $AX^{[0]} = \mathbf{A}X^{[0]}$
- 2:  $a^{[0]} = (X^{[0]})^T AX^{[0]}$  ▷ Initial Rayleigh-Ritz
- 3: solve  $a^{[1]}u^{[1]} = u^{[1]}\lambda^{[1]}$
- 4:  $X^{[1]} = X^{[0]}u^{[1]}$  and  $AX^{[1]} = AX^{[0]}u^{[1]}$
- 5:  $R^{[1]} = AX^{[1]} - X^{[1]}\lambda^{[1]}$
- 6:  $\widetilde{W}^{[1]} = TR^{[1]}$  ▷ Preconditioned residuals...
- 7:  $W^{[1]} = \text{ortho}(\widetilde{W}^{[1]}; X^{[1]})$  ▷ ... orthogonalized
- 8:  $AW^{[1]} = \mathbf{A}W^{[1]}$
- 9:  $k = 0$
- 10: **while**  $k < k_{\max}$  **do**
- 11:    $k = k + 1$
- 12:   **if**  $k = 1$  **then** ▷ Expansion subspace  $V^{[k]}$
- 13:      $V^{[k]} = (X^{[k]}, W^{[k]})$
- 14:   **else**
- 15:      $V^{[k]} = (X^{[k]}, W^{[k]}, P^{[k]})$
- 16:   **end if**
- 17:    $a^{[k]} = (V^{[k]})^T AV^{[k]}$  ▷ Rayleigh-Ritz in  $V^{[k]}$
- 18:   solve  $a^{[k]}u^{[k]} = \lambda^{[k]}u^{[k]}$ , get lowest eigenvectors  $u_x^{[k]}$
- 19:    $X^{[k+1]} = V^{[k]}u_x^{[k]}$  and  $AX^{[k+1]} = AV^{[k]}u_x^{[k]}$
- 20:    $R^{[k]} = AX^{[k]} - X^{[k]}\lambda^{[k]}$
- 21:   Lock converged eigenvectors, exit if done
- 22:   Compute  $\widetilde{u}_p^{[k]}$  from unconverged  $u_x^{[k]}$  ▷ Components of  $P$  in  $V^{[k]}$ ...
- 23:    $u_p^{[k]} = \text{ortho}(\widetilde{u}_p^{[k]}; u_x^{[k]})$  ▷ ... orthogonalized
- 24:    $P^{[k+1]} = V^{[k]}u_p^{[k]}$  and  $AP^{[k+1]} = AV^{[k]}u_p^{[k]}$
- 25:    $\widetilde{W}^{[k+1]} = TR^{[k]}$  ▷ Preconditioned residuals...
- 26:    $W^{[k+1]} = \text{ortho}(\widetilde{W}^{[k+1]}; (X^{[k+1]}, P^{[k+1]}))$  ▷ ... orthogonalized
- 27:    $AW^{[k+1]} = \mathbf{A}W^{[k+1]}$
- 28: **end while**

---

**3.1.1 Basis selection**

In such algorithm,  $X^{[k]}, W^{[k]}, P^{[k]}$  are the  $n \times m$  matrices that contain the  $m$  desired eigenvectors ( $X$ ) and the corresponding preconditioned residuals ( $W$ ) and increments ( $P$ ), and we denote with a  $\sim$  symbol the vectors before orthogonalization. The matrices  $AX^{[k]}, AW^{[k]}, AP^{[k]}$  contain the results of the application of  $\mathbf{A}$  to such vectors. LOBPCG is a *matrix-free* algorithm, i.e., it does not require to assemble and store in memory the matrix  $\mathbf{A}$  but just to be able to perform the

relevant matrix–vector multiplications. More details on the orthogonalization procedure are given in Sect. 3.2. The reduced matrix  $a^{[k]} \in \mathbb{R}^{3m \times 3m}$  is diagonalized using standard dense linear algebra routines (in our implementation, we use LAPACK's DSYEV).

In the block implementation of LOBPCG, computing  $\widetilde{P}^{[k]}$  as  $X^{[k]} - X^{[k-1]}$  can become problematic, as these vectors become smaller and smaller when approaching convergence, which can create numerical instabilities. As a more robust alternative, the  $\widetilde{P}^{[k]}$  vectors are computed in a different way.

Let  $u_x^{[k]} \in \mathbb{R}^{3m \times m}$  be the first  $m$  eigenvectors of the reduced matrix. The eigenvectors have a block structure, that is

$$u^{[k]} = \begin{pmatrix} u_{xx}^{[k]} \\ u_{wx}^{[k]} \\ u_{px}^{[k]} \end{pmatrix} \quad (1)$$

where each block is a  $m \times m$  square matrix. The new eigenvectors are computed as (line 19)

$$X^{[k+1]} = X^{[k]}u_{xx}^{[k]} + W^{[k]}u_{wx}^{[k]} + P^{[k]}u_{px}^{[k]} \quad (2)$$

To compute the  $P^{[k+1]}$  vectors, we first get the expansion coefficients  $\tilde{u}_p^{[k+1]}$  of  $X^{[k+1]} - X^{[k]}$  in  $V^{[k]}$ , which are obtained by subtracting the identity matrix from the unconverged components of  $u_{xx}^{[k]}$ . Then, we orthogonalize them against  $u_x$  and use them to compute the new  $P^{[k+1]}$  vectors. Note that we assemble the  $P$  vectors only corresponding to the active eigenvectors, i.e., the ones that have not yet converged; for this reason, it is important to perform such operation before orthogonalizing  $\tilde{u}_p^{[k+1]}$ .

### 3.1.2 Reuse of applications

Apart from the choice of a basis and its orthogonalization, a numerically sensitive point is the reuse of the applications of  $\mathbf{A}$ . Since this is a potentially costly operation, it is not feasible to recompute for instance  $\mathbf{A}P^{[k]}$  before the Rayleigh-Ritz procedure; instead, we use the fact that  $P^{[k]}$  is built as a linear combination of other vectors, on which we know the application of  $\mathbf{A}$ . If this is done naively, however this can result in a large error. This is because, in general, if  $\mathbf{A}V$  is known to some precision  $\epsilon$ , then  $(\mathbf{A}V)u$  will be an approximation of  $\mathbf{A}(Vu)$  with a precision of the order of  $\|u\|\epsilon$ .

Consider the problem of computing  $\mathbf{A}P^{[k]}$ , line 24 of the algorithm. In exact arithmetic, we could compute  $\tilde{P}^{[k]} = V^{[k]}\tilde{u}_p^{[k]}$ , compute  $\tilde{\mathbf{A}}\tilde{P}^{[k+1]} = (\mathbf{A}V^{[k]})\tilde{u}_p^{[k]}$ , orthogonalize  $\tilde{P}^{[k+1]}$  against  $X^{[k+1]}$  and update  $\tilde{\mathbf{A}}\tilde{P}^{[k+1]}$  accordingly, etc. This however amounts to obtaining  $\mathbf{A}P^{[k+1]}$  by right-multiplying  $\mathbf{A}V^{[k]}$  with a sequence of potentially ill-conditioned (and therefore of large norm) matrices, which incurs a large error on  $\mathbf{A}P^{[k+1]}$ . Instead, we obtain directly the expansion coefficients  $u_p^{[k]}$  of  $P^{[k+1]}$  on  $V^{[k]}$  and obtain  $\mathbf{A}P^{[k+1]}$  as  $(\mathbf{A}V^{[k]})u_p^{[k]}$ . Since both  $P^{[k+1]}$  and  $V^{[k]}$  are orthogonal, so is  $u_p^{[k]}$ , and therefore no precision is lost in the update  $\mathbf{A}P^{[k+1]} = \mathbf{A}V^{[k]}u_p^{[k]}$ . The same is true for the update  $\mathbf{A}X^{[k+1]} = \mathbf{A}V^{[k]}u_x^{[k]}$  (line 19).

### 3.1.3 Locking

Another crucial aspect of an efficient and stable implementation concerns the treatment of converged eigenvectors. In our implementation, we freeze the first  $m_{\text{conv}}$

consecutive eigenvectors, which means that we only compute  $m_{\text{act}} = m - m_{\text{conv}}$  new residuals,  $W$  and  $P$  vectors. The converged eigenvectors are kept into  $X$ , to enforce the orthogonality of the active search subspace. This means that the reduced matrix and the  $V$  subspace dimensions are  $m + 2m_{\text{act}}$ , and that only  $m_{\text{act}}$  matrix–vector multiplications are performed at each iteration, combining thus stability and efficiency.

## 3.2 A robust and stable `ortho(X, Y)` procedure

One of the most crucial steps in LOBPCG is the orthogonalization of a set of vectors against a given set and its subsequent orthonormalization. We first tackle the `ortho(X)` routine, which orthogonalizes a set of vectors.

### 3.2.1 The `ortho(X)` procedure

The gold standard for orthogonalizing a set of vectors is to compute the (thin) singular value decomposition of  $X$  and then take the left singular vectors. A slightly less expensive, yet very stable alternative, is to use the QR decomposition of  $X$ , which in our tests performs equivalently well. Another good option is the modified Gram-Schmidt algorithm. However, these algorithms can all become expensive, especially if a large number of eigenvalues are sought.

An alternative and cheaper strategy is to compute the Cholesky decomposition of the overlap matrix

$$X^T X = M = LL^T \quad (3)$$

The orthogonal vectors can then be obtained by solving the triangular linear system

$$\text{ortho}(X)L^T = X. \quad (4)$$

This is often more efficient, as it allows for greater parallelization and full use of BLAS3 routines.

This procedure works in infinite precision, but has two issues in finite precision. First, even after a first Cholesky orthogonalization, the vectors can fail to be orthogonal. Second, the Cholesky decomposition can fail, because  $M$  may not be positive-definite to machine precision; this happens when the conditioning of  $X$  is larger than the square root of the inverse machine epsilon, about  $10^8$  in double-precision arithmetic.

Fortunately, there is a simple fix to the first problem: orthogonalize twice. This has been established to produce vectors orthogonal to machine precision [32]. For the second problem, following [27], we level shift the metric before its Cholesky decomposition by adding a small constant to its diagonal. Such a constant can be chosen very small (in our implementation, we start from 100 times the norm of  $X$  times the machine precision) and, if the decomposition

still fails, increased until the Cholesky decomposition is successful. In our tests, the shifted decomposition never failed, and therefore, with at most 4 Cholesky orthogonalizations (one failed unshifted, one shifted, then two unshifted) we are

guaranteed to obtain vectors that are orthogonal to machine precision. In practice, often much less than this is needed—failures of the first Cholesky orthogonalization have been observed only exceptionally. A pseudo-code for the `ortho` procedure is given in algorithm 2.

---

**Algorithm 2** `ortho`( $\tilde{X}$ ): Orthonormalize a set of vectors  $\tilde{X}$  using the Cholesky decomposition of the overlap with iterative refinement.

---

**Input:** non orthogonal vectors  $\tilde{X}$ , threshold  $\tau_{\text{ortho}}$ .

**Output:**  $X$ , orthonormal vectors.

```

1:  $X = \tilde{X}$ 
2: while  $\|X^T X - Id\| > \tau_{\text{ort}}$  do
3:    $M = X^T X$ 
4:   Attempt Cholesky factorization  $M = LL^T$ 
5:   if fail then
6:     Add  $\alpha\epsilon\|X\|$  to the diagonal of  $M$  until successful
7:   end if
8:    $X = XL^{-T}$ 
9: end while

```

---

The algorithm as given is somewhat wasteful, as in the common case where only one or two successful Cholesky factorizations are needed it recomputes the overlap to check for termination. This can potentially be alleviated by computing a cheap estimation to the norm of  $L^{-T}$ : if this is moderate, then the new vectors are orthogonal to a good accuracy, and a new round is unnecessary.

### 3.2.2 The `ortho`( $X, Y$ ) procedure

Using the previous orthogonalization algorithm, we could implement the `ortho`( $X, Y$ ) as `ortho`( $X - YY^T X$ ). This is

however numerically unstable: if  $X - YY^T X$  is of order  $\delta$  (because  $X$  was almost in the range of  $Y$ ), then the orthogonalization above will multiply it by a factor of order  $1/\delta$ , meaning that  $Y^T \text{ortho}(X - YY^T X)$  will be of order  $\epsilon/\delta$ , where  $\epsilon$  is the machine epsilon, and the vectors will not be sufficiently orthogonal to  $Y$ . To avoid this, we use a loop: first project out  $Y$ , then orthogonalize, iteratively until convergence. In practice, two steps are usually enough to achieve convergence.

The algorithm for the `ortho`( $X, Y$ ) procedure is given in algorithm 3.

---

**Algorithm 3** `ortho`( $\tilde{X}, Y$ ): given a set of orthonormal vectors  $Y$  and a set of vectors  $\tilde{X}$ , orthogonalize  $\tilde{X}$  to  $Y$  and orthonormalize  $\tilde{X}$ .

---

**Input:** orthonormal vectors  $Y$ , non orthogonal vectors  $\tilde{X}$ , threshold  $\tau_{\text{ortho}}$ .

**Output:**  $X$ , orthonormal vectors also orthogonal to  $Y$ .

```

1:  $X = \tilde{X}$ 
2: while  $\|Y^T X\| > \tau_{\text{ortho}}$  do
3:    $X = X - YY^T X$ 
4:    $X = \text{ortho}(X)$ 
5: end while

```

---

Similar to before, this algorithm is relatively wasteful in the common case where one or two passes are enough, because it recomputes  $Y^T X$  to check for termination. This can be remedied by monitoring the growth factor of `ortho`( $X$ ) (the maximum amplification of errors in  $X$  caused by the `ortho` routine, and therefore a measure of the lack of  $Y$ -orthogonality after one iteration), and exiting the loop when that number is moderate.

### 3.3 Generalized eigenvalue problems

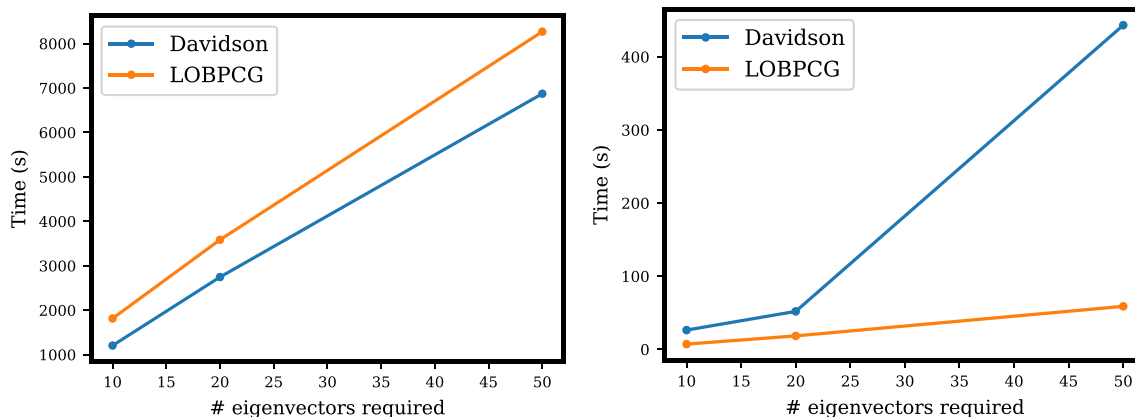
Missing from this algorithm is a discussion of generalized eigenvalue problems, simply because they are not often encountered in quantum chemistry. In the generalized eigenvalue problem, one solves  $\mathbf{A}x = \lambda \mathbf{B}x$ , where  $\mathbf{B}$  is a symmetric positive definite matrix. Eigenvectors are orthogonal with respect to the modified inner product  $\langle x, y \rangle_{\mathbf{B}} = x^T \mathbf{B} y$ . The theoretical LOBPCG algorithm is unchanged except for the fact that residuals are now  $\mathbf{A}x - \lambda \mathbf{B}x$ , and that all

orthogonalizations are with respect to the modified inner product.

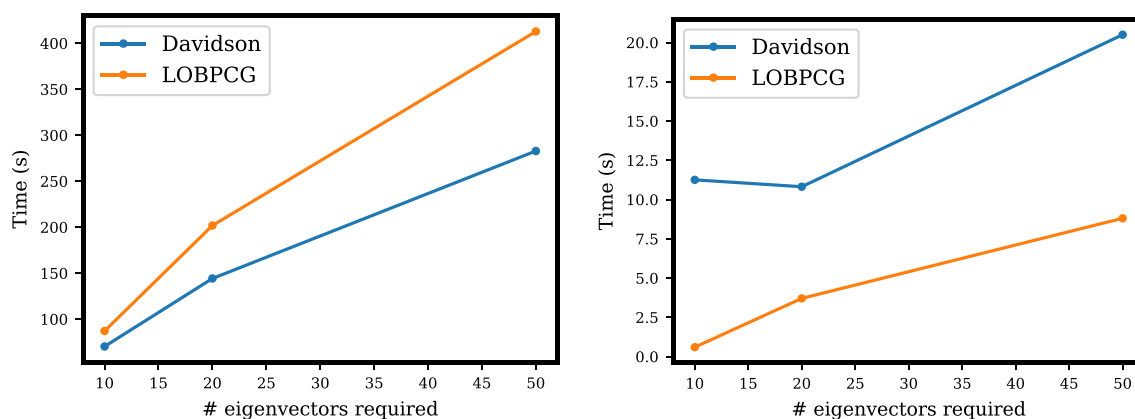
Our practical algorithm has to be modified by keeping a  $\mathbf{B}$ -orthogonal basis  $V$ . This could be done by using the  $\mathbf{B}$  inner product in the orthogonalization of  $W$ , line 26 of the algorithm above, and maintaining the values of  $\mathbf{B}V$  along the iterations. However, a naive implementation of this step requires either multiple applications of  $\mathbf{B}$  (twice per iteration, on  $\tilde{W}^{[k+1]}$  and on  $W^{[k+1]}$ ) or potentially unsafe reuses of applications of  $\mathbf{B}$ . As a compromise, a good option is to use the intermediate quantity

$$\hat{W}^{[k+1]} = \text{ortho}(\tilde{W}^{[k+1]}, (\mathbf{B}X^{[k+1]}, \mathbf{B}P^{[k+1]})),$$

which is  $\mathbf{B}$ -orthogonal to  $X^{[k+1]}$  and  $P^{[k+1]}$ , but whose vectors are only orthogonal (and not  $\mathbf{B}$ -orthogonal) to each other. This set of vectors  $\hat{W}^{[k+1]}$  is however well conditioned (with respect to the  $\mathbf{B}$ -inner product). We can therefore  $\mathbf{B}$ -orthogonalize it to compute  $W^{[k+1]} = \hat{W}^{[k+1]} L^{-T}$  with  $L$  a well-conditioned matrix; it is then safe to re-use the  $\mathbf{B}$  application



**Fig. 1** All-electron full CI calculations for water using Davidson or LOBPCG. Total timings (left panel) and cumulative time for the Rayleigh-Ritz and orthogonalization procedures (right panel)



**Fig. 2** Frozen-core full CI calculations for water using Davidson or LOBPCG. Total timings (left panel) and cumulative time for the Rayleigh-Ritz and orthogonalization procedures (right panel)

as  $\mathbf{B}W^{[k+1]} = (\mathbf{B}\hat{W}^{[k+1]})L^{-T}$ . This appeared to perform very well in our tests, even if  $\mathbf{B}$  itself was not well-conditioned. If more stability is needed, reuses of  $\mathbf{B}$  applications appear necessary.

## 4 Numerical experiments

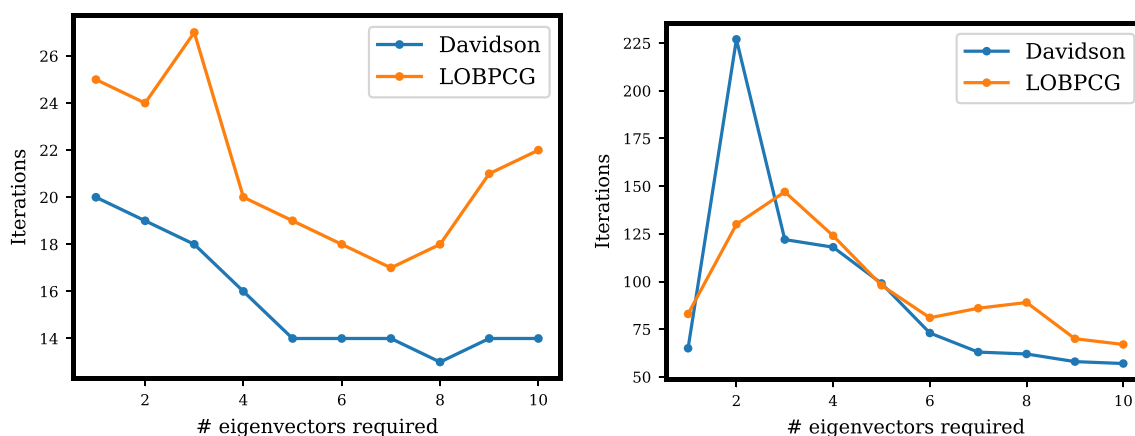
To test our implementation of LOBPCG, and to compare its performance with respect to the block-Davidson method, we interfaced the `DIAGLIB` library with the `CFOUR` quantum chemistry package [28, 29]. As typical problems where an iterative procedure to compute one or a few eigenvectors is required, we selected three different test cases, coming from full CI Hamiltonian, quadratically convergent self-consistent field (SCF) and quadratically convergent complete active space self-consistent field (CASSCF). In all the calculations, we use a threshold of  $10^{-14}$  for the `ortho(X, Y)` and `ortho(X)` procedures. Together with the convergence threshold for the eigenvectors, this is the only parameters that control the LOBPCG calculation. For Davidson, we use a subspace dimension of 25, i.e., we keep in memory up to 25 vectors per eigenvector in the history. For both the algorithms we exploit a locking procedure for the converged eigenvectors.

### 4.1 Full CI calculations

We compute the first few total symmetric electronic states of water at the full CI level of theory, using a determinant CI direct implementation. The full CI Hamiltonian is sparse and diagonally dominant, but extremely large, and thus provides a good test case for well-behaved, large, sparse

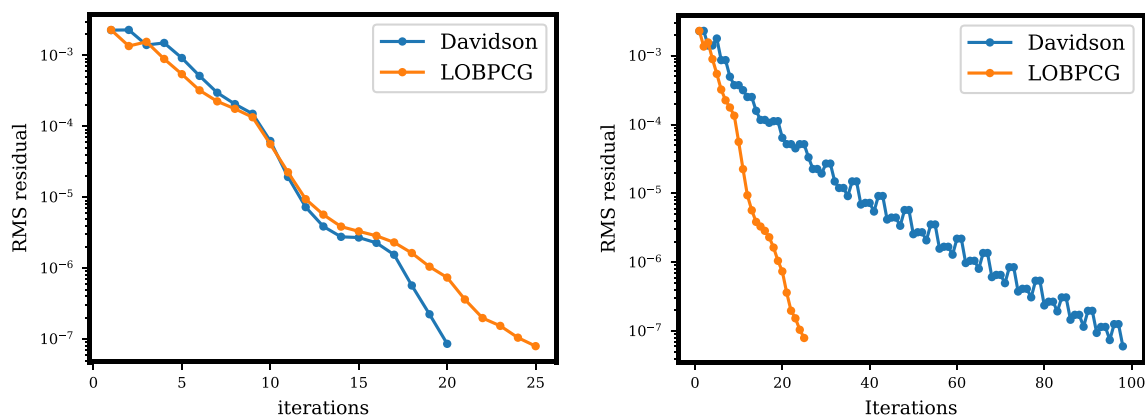
systems. Furthermore, the iterative solution of (Full) CI problems is a quite common task in quantum chemistry, as it is encountered in CASCI/CASSCF and truncated CI (including for excited states at the CI singles level of theory). We use Pople's 6-31 G\* basis set [33] and perform both all-electron and frozen-core calculations, correlating thus 10 electrons in 18 orbitals (18 360 640 determinants) or 8 electrons in 10 orbitals (1 416 732 determinants). We seek 10, 20 or 50 eigenpairs. Convergence is achieved when the root-mean-square norm of the residual is smaller than  $10^{-9}$ , and its maximum absolute value is smaller than  $10^{-8}$ . For LOBPCG, we seek 5 additional eigenpairs, as numerical tests proved that this improves convergence and, despite the additional matrix-vector products required, improves overall performance. Note that we do not check that the additional eigenvalues are converged, as they are only used to increase the expansion subspace. No additional eigenpairs are sought for Davidson, as this choice showed the overall best performance. A brief description of the process that led us to these choices is reported in the Supporting Information.

It comes as no surprise that Davidson outperforms LOBPCG for Full-CI calculations. As the full-CI Hamiltonian is strongly diagonally dominant for closed-shell systems, it is an ideal scenario for Davidson, an algorithm originally conceived for exactly this problem. Nevertheless, the performance of LOBPCG is comparable, the latter algorithm being about 20–30% slower than the former. It is interesting to note that while for LOBPCG the cost of the Rayleigh-Ritz and orthogonalization procedures is overall negligible with respect to the cost of computing matrix-vector multiplications, this is not the case for Davidson. Keeping up to 25 vectors in the history comes with a cost that can be clearly seen in the right panels of Figs. 1, 2. On the other hand, the larger subspace used in Davidson's method allows for faster convergence, which



**Fig. 3** Orbital rotation Hessian diagonalization at SCF convergence (left panel) and with extended Hückel guess orbitals (right panel). The number of iterations required to achieve convergence is reported as a function of the number of sought eigenpairs





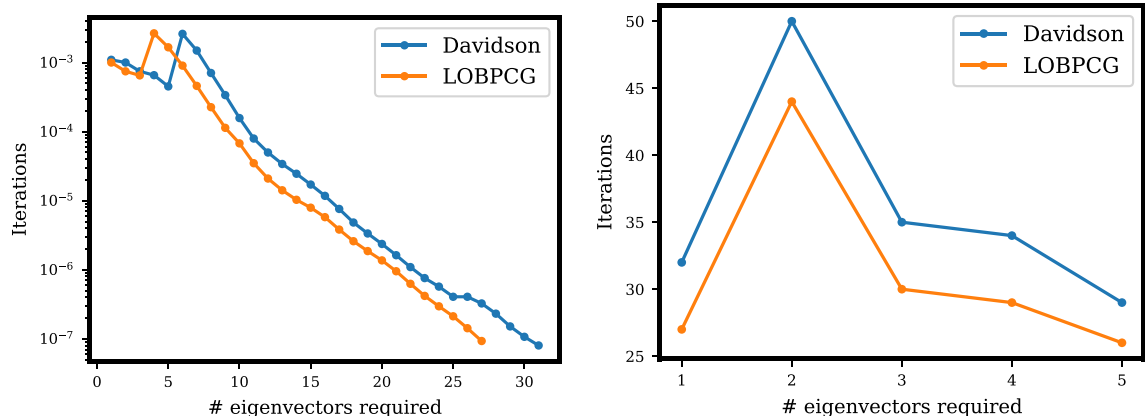
**Fig. 4** Root-mean-square (RMS) of the residual along the iterations for the diagonalization of the orbital rotation Hessian at SCF convergence seeking for one eigenpair. The subspace for the Davidson algorithm is of size 25 (left panel) and 3 (right panel)

is achieved in 28, 37 and 24 iterations for the calculation seeking 10, 20 and 50 eigenpairs, respectively, both for the all-electron and the frozen-core calculations. This has to be compared with 26, 41 and 45 iterations for LOBPCG, again, for both sets of calculations. On the other hand, the long history is also a limitation for Davidson, as the amount of memory required to perform a calculation can become very high. As an example, the largest calculation performed (all-electron, 50 states) required about 356 GB of memory for Davidson, to be compared with 55 for LOBPCG. Using a smaller subspace dimension in Davidson is of course possible, but such a size must be chosen with some care. To better illustrate this point, we repeated the Davidson calculations using a maximum of 10 points in the history: no calculation fully converged within 100 iterations, with 1, 3 and 2 non-converged roots. Therefore, while Davidson is optimal for full CI calculations if enough memory is available to use a large expansion

subspace, LOBPCG can be seen as a competitive alternative when this is not the case.

#### 4.2 Quadratically convergent SCF calculations

In quadratically convergent implementations of the self-consistent field, the Hartree-Fock wavefunction is optimized by using a second-order method. In CFOUR, this is done using an efficient numerical realization of the Levenberg-Marquardt method [34], known as norm-extended optimization, where the step is computed from the lowest eigenpair of the (augmented) orbital-rotation Hessian [35]. The same matrix is used in response calculations and for the analysis of the stability of the Hartree-Fock wavefunction, which requires again to compute a few eigenvalues and eigenvectors of the orbital-rotation Hessian. For closed-shell systems, such a matrix is dense, but typically diagonally dominant. While computing and storing in memory the full Hessian is



**Fig. 5** Root-mean-square (RMS) of the residual along the iterations seeking for one eigenpair (left panel) and number of iterations to achieve convergence as a function of the number of sought eigenpairs

(right panel) for the diagonalization of the CASSCF augmented Hessian at the first macroiteration

possible, such a task is expensive and exhibits a steep scaling in computational cost with respect to the system's size, as assembling it requires a costly partial integral transformation. Direct implementations are therefore usually preferred. To compare Davidson to LOBPCG, we compute the first few eigenpairs (up to 10) of the Hessian after convergence of the SCF procedure for a transition metal complex,  $\text{FeC}(\text{CO})_3$ , using Dunning's *cc*-pVDZ basis set. To provide an example on a somewhat more challenging case, we further repeat the calculations, but at the beginning of the SCF calculations, that is, using orbitals computed with an extended Hückel guess. Far from convergence, the Hessian has many negative eigenvalues and is not guaranteed to be as diagonally dominant as with fully converged orbitals. In the following calculations, convergence is achieved when the root-mean-square norm of the residual is smaller than  $10^{-7}$ , and its maximum absolute value is smaller than  $10^{-6}$ . Such threshold is adequate for stability analysis, but also for the first steps of a second-order optimization procedure. For both Davidson and LOBPCG, to improve convergence, we seek twice the eigenpairs required and stop the calculation when just the required ones are converged. The size of the subspace for Davidson is 25. For the calculations at SCF convergence, where the Hessian is strongly diagonally dominant (Fig. 3, left panel) Davidson outperforms again LOBPCG. On the other hand, the less well-behaved case (Fig. 3, right panel) shows a different picture. LOBPCG and Davidson exhibit a very similar behavior, with LOBPCG even outperforming Davidson in a few cases. As orbital rotation Hessians are hardly very large matrices (for the case reported here, the size is 3849), using large expansion subspaces in Davidson can probably further improve convergence, but for difficult cases, LOBPCG can be a valid alternative.

We report in Fig. 4 (left panel) the root-mean-square residual as a function of the iterations for the computation of the first eigenpair of the orbital rotation Hessian. As expected the behaviors between the two algorithms are analogous.

We would like here to underline the fact that LOBPCG manages to behave similarly to Davidson even despite the very small size of the subspace used for the Rayleigh-Ritz procedure. To show how remarkable this is, we report in Fig. 4 (right panel) a comparison between LOBPCG and Davidson, where for the latter we use a three-dimensional subspace—that is, the same dimension used in LOBPCG. Davidson eventually manages to converge; however, it requires a much larger number of iterations. While this is purely an academic example, as such small expansion subspaces are never used in practice, it testifies to the effectiveness of the LOBPCG 3-terms sequence.

**Table 1** Total number of iterations required to converge the lowest eigenvalue of the CASSCF orbital-rotation Hessian at the first macroiteration using the LOBPCG and Davidson solvers with three different preconditioners

LOBPCG			Davidson		
Diagonal	Tridiagonal	iLU	Diagonal	Tridiagonal	iLU
56	53	47	52	54	50

### 4.3 CASSCF calculations

CASSCF calculations can be very challenging from a numerical point of view, which makes second-order methods particularly attractive [4, 36–38]. In *CFOUR*, the same technique used for Hartree-Fock, namely, the norm-extended optimization algorithm, is used. The (augmented) Hessian in CASSCF is made by a dense, typically quite ill-conditioned, medium-sized block for the orbital optimization and a large, sparse, usually diagonally dominant block that corresponds to the Hamiltonian in the CAS space. Even for well-behaved systems, computing the NEO step, which in turn requires computing the first eigenpair of the augmented Hessian, can be challenging. To illustrate this, we report calculations on niacin (vitamin B3), a small conjugated organic molecule. We correlated all the  $\pi$  electrons, resulting in a CAS(6,6) calculation, and we employ Pople's 6–31 G\* basis set [33]. Symmetry broken unrestricted natural orbitals are used as a guess [39, 40]. The system is very well behaved, and convergence of the wavefunction is achieved in just 3 s-order iterations. Nevertheless, the iterative calculation of the step, i.e., the augmented hessian lowest eigenvector, can still be challenging. We report in the left Fig. 5 the convergence pattern for Davidson and LOBPCG at the first second-order iteration. LOBPCG outperforms Davidson at every second-order iterations, 27, 25 and 23 iterations at the first, second and third second-order step, to be compared with 32, 28 and 31 iterations for Davidson. Also when more than one eigenpair is required, as it is the case for state-specific excited state calculations, LOBPCG keeps being the best performing algorithm, as reported in the right panel of Fig. 5. This behavior is not surprising, as the CASSCF augmented Hessian is not diagonally dominant and is consistent with what observed for second-order SCF calculations starting with extended Hückel orbitals Davidson's performance may be improved by increasing the size of the expansion subspace. This is, however, not the best option, as for large active spaces the size of the Hessian can become very large—comparable with the sizes reported for full CI calculations. Using very large expansion subspaces is therefore very demanding

in terms of memory, and can make the orthogonalization expensive. LOBPCG seems therefore a more robust choice for this specific problem.

#### 4.4 Preconditioning

To improve the convergence of both LOBPCG and Davidson one can devise different types of preconditioners. In most quantum chemistry applications, computing and storing in memory the matrix of which one seeks one or a few eigenpairs is prohibitively expensive, which forces the choice of a Jacobi (diagonal) preconditioner in most cases. This is definitely the case for the full CI and CASSCF examples showed in the previous section. However, for both second-order SCF and CASSCF, we have implemented the explicit construction of the orbital rotation Hessian, mainly as a debug option, which allows us to perform a few numerical experiments. In particular, we compare three possible choices and focus on the CASSCF orbital-rotation Hessian as a test case, as it is notoriously ill-conditioned and therefore we expect that a more advanced preconditioning strategy may be particularly beneficial. For these examples, we only seek to compute one eigenpair. The first preconditioner that we test is, as in the previous sections, approximates the matrix to its diagonal. The second one, which will be here addressed as tridiagonal, improves upon the diagonal approximation by also including the upper and lower diagonal elements. These two options should perform particularly well in diagonally dominant matrices. As a third choice, we propose a sparse approximation  $M$  to the matrix  $A$ , that is

$$M_{ij} = \begin{cases} A_{ij}, & \text{if } |A_{ij}| > \text{tol or } i = j \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where we set tol equal to 0.5 and decreased it to 0.1 as soon as the root-mean-square norm of the residual is close to convergence. For both the second and third options it is necessary to solve a linear system. For the tridiagonal case we simply exploit a LAPACK routine which performs a Gaussian elimination with partial pivoting. Instead, in the case of the sparse  $M$  matrix, the linear system is solved using the incomplete LU (iLU) decomposition [41] as implemented by Saunders et al [42]. In Table 1 we compare the behavior of LOBPCG and Davidson when changing the preconditioner. As expected the preconditioner based on the sparse approximation of  $A$  is the one that performs best. However, such an option may become expensive and requires to store the full matrix in core or at least to have an heuristic procedure to estimate the elements. From this simple-minded experiment, we note that both methods benefit from better preconditioners, with LOBPCG exhibiting slightly more marked improvements. However, assembling such preconditioners is expensive, as it requires to build the matrix, or at least

some approximation to it, in memory, which in many practical cases is far too demanding. Given the relatively small beneficial effect of going beyond a diagonal preconditioner, we believe that the latter is the optimal compromise choice.

## 5 Conclusions and perspectives

In this contribution, we have described an efficient and numerically robust implementation of LOBPCG, available both in the DFTK plane-wave density functional program and in the open-source library DIAGLIB. We have discussed in detail how to avoid numerical problems and error propagation and presented a cost-effective, yet stable strategy to orthogonalize a set of vectors using Cholesky decomposition of the overlap matrix. We have then compared the resulting implementation to Davidson's method for a selection of test cases in quantum chemistry. Davidson's method is the *de facto* standard for solving large eigenvalue problems in quantum chemistry and for good reasons. As many of such problems are characterized by strongly diagonally dominant matrices, Davidson's method always exhibits reliable, fast convergence. This comes, however, at a price. To be efficient, Davidson's method requires a rather large expansion subspace, which can become cumbersome for large-scale calculations, both in terms of memory requirements and computational effort in the orthogonalization step. Furthermore, the method has some difficulties dealing with non-diagonally dominant matrices, as the ones encountered in CASSCF calculations. For all these reasons, LOBPCG represents a valid alternative. Due to its low memory requirements, it can be used to treat systems for which deploying Davidson's method would be too costly. It can also be a backup method in cases where Davidson fails or, for particularly hard cases as in CASSCF, used as a default method. The implementation in DIAGLIB is free and accessible, and can be used under the terms of the LGPL v2.1 license, while the one in DFTK is available under the MIT license. It is our hope that it will provide an useful tool to the developers community in quantum chemistry.

Our numerical experiments highlight the degradation of the convergence rate of the Davidson method as the history size is truncated. On the other hand, LOBPCG is able to maintain a good convergence rate (although slightly inferior to untruncated Davidson) with a subspace of size  $3N$ . It would be an interesting topic of further research to devise a method that is able to interpolate between the two, being able to use a large history size if available, but preserving the good behavior of LOBPCG when used with a smaller history size. This could pave the way toward a fully adaptive method that truncates the history size dynamically based on available information.

**Supplementary Information** The online version contains supplementary material available at <https://doi.org/10.1007/s00214-023-03010-y>.

**Acknowledgements** This paper is dedicated to Maurizio Persico, an invaluable teacher, mentor and colleague, and the most enthusiastic mountaineer. A.L. wishes to thank Michael Herbst for help implementing and stress-testing the algorithm. F.L. T.N. and I.G. acknowledge financial support from ICSC-Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing, funded by the European Union–Next Generation EU–PNRR, Missione 4 Componente 2 Investimento 1.4. F.L. further acknowledges funding from the Italian Ministry of Research under grant 2020HTSXMA\_002 (PSI-MOVIE).

**Author Contributions** AL and FL wrote the main manuscript text. TN, IG, FL wrote the Fortran code used in the paper. AL developed the main numerical strategies presented. TN and IG performed the calculations and gathered the data. All authors reviewed the manuscript.

**Funding** Open access funding provided by Università di Pisa within the CRUI-CARE Agreement.

## Declarations

**Conflict of interest** The authors declare no competing interests.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Roothaan CCJ (1951) New developments in molecular orbital theory. *Rev Mod Phys* 23:69–89
- Roothaan CCJ (1960) Self-consistent field theory for open shells of electronic systems. *Rev Mod Phys* 32:179–185
- Almlöf J, Faegri K Jr, Korsell K (1982) Principles for a direct scf approach to lcao-moab-initio calculations. *J Comp Chem* 3(3):385–399
- Jensen H-JA, Jørgensen P (1984) A direct approach to second-order mscf calculations using a norm extended optimization scheme. *J Chem Phys* 80(3):1204–1214
- Christiansen O, Jørgensen P, Hättig C (1998) Response functions from fourier component variational perturbation theory applied to a time-averaged quasienergy. *Int J Quant Chem* 68(1):1–52
- Casida ME (1995) Time-dependent density functional response theory for molecules. World Scientific, Singapore, pp 155–192
- Schirmer J (1982) Beyond the random-phase approximation: a new approximation scheme for the polarization propagator. *Phys Rev A* 26(5):2395
- Dreuw A, Wormit M (2015) The algebraic diagrammatic construction scheme for the polarization propagator for the calculation of excited states. *WIREs Comp Mol Sci* 5(1):82–95
- Bartlett RJ, Kucharski SA, Noga J (1989) Alternative coupled-cluster Ansätze II. The unitary coupled-cluster method. *Chem Phys Lett* 155(1):133–140
- Taube AG, Bartlett RJ (2006) New perspectives on unitary coupled-cluster theory. *Int J Quant Chem* 106(15):3393–3401
- Liu J, Asthana A, Cheng L, Mukherjee D (2018) Unitary coupled-cluster based self-consistent polarization propagator theory: a third-order formulation and pilot applications. *J Chem Phys* 148(24):244110
- Handy NC (1980) Multi-root configuration interaction calculations. *Chem Phys Lett* 74(2):280–283
- Olsen J, Roos BO, Jørgensen P, Jensen HJA (1988) Determinant based configuration interaction algorithms for complete and restricted configuration interaction spaces. *J Chem Phys* 89(4):2185–2192
- Werner H-J (2009) Matrix-formulated direct multiconfiguration self-consistent field and multiconfiguration reference configuration-interaction methods. *Adv Chem Phys* 69:1–62
- Roos BO (1987) The complete active space self-consistent field method and its applications in electronic structure calculations. *Adv Chem Phys* 69:399–445
- Shepard R et al (1987) The multiconfiguration self-consistent field method. *Adv Chem Phys* 69:63–200
- Davidson ER (1975) The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *J Comp Phys* 17(1):87–94. [https://doi.org/10.1016/0021-9991\(75\)90065-0](https://doi.org/10.1016/0021-9991(75)90065-0)
- Liu B (1978) The simultaneous expansion method for the iterative solution of several of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *Numer Algorithms Chem Algebraic Methods* 49–53
- Zuev D, Vecharynski E, Yang C, Orms N, Krylov AI (2015) New algorithms for iterative matrix-free eigensolvers in quantum chemistry. *J Comp Chem* 36(5):273–284
- Knyazev AV (2001) Toward the optimal preconditioned eigensolver: locally optimal block preconditioned conjugate gradient method. *SIAM J Sci Comput* 23(2):517–541
- Knyazev AV, Argentati ME, Lashuk I, Ovtchinnikov EE (2007) Block Locally Optimal Preconditioned Eigenvalue Solvers (BLOPEX) in Hypre and PETSc. *SIAM J Sci Comput* 29:2224–2234
- Bottin F, Leroux S, Knyazev A, Zérah G (2008) Large-scale ab initio calculations based on three levels of parallelization. *Comput Mater Sci* 42(2):329–336
- Shao M, Aktulga HM, Yang C, Ng EG, Maris P, Vary JP (2018) Accelerating nuclear configuration interaction calculations through a preconditioned block iterative eigensolver. *Comput Phys Commun* 222:1–13
- Hetmaniuk U, Lehoucq R (2006) Basis selection in lobpcg. *J Comput Phys* 218(1):324–332
- Stathopoulos A, Wu K (2002) A block orthogonalization procedure with constant synchronization requirements. *SIAM J Sci Comput* 23(6):2165–2182
- Duersch JA, Shao M, Yang C, Gu M (2018) A robust and efficient implementation of lobpcg. *SIAM J Sci Comput* 40(5):655–676
- Fukaya T, Kannan R, Nakatsukasa Y, Yamamoto Y, Yanagisawa Y (2020) Shifted Cholesky QR for computing the QR factorization of ill-conditioned matrices. *SIAM J Sci Comput* 42(1):477–503
- Stanton JF, Gauss J, Cheng L, Harding ME, Matthews DA, Szalay PG. CFOUR, Coupled-Cluster techniques for Computational Chemistry, a quantum-chemical program package. With contributions from A. Asthana, A.A. Auer, R.J. Bartlett, U. Benedikt, C. Berger, D.E. Bernholdt, S. Blaschke, Y. J. Bomble, S. Burger, O. Christiansen, D. Datta, F. Engel, R. Faber, J. Greiner, M. Heckert,

- O. Heun, M. Hilgenberg, C. Huber, T.-C. Jagau, D. Jonsson, J. Jusélius, T. Kirsch, M.-P. Kitsaras, K. Klein, G.M. Kopper, W.J. Lauderdale, F. Lipparini, J. Liu, T. Metzroth, L.A. Mück, D.P. O'Neill, T. Nottoli, J. Oswald, D.R. Price, E. Prochnow, C. Puzzarini, K. Ruud, F. Schiffmann, W. Schwalbach, C. Simmons, S. Stopkowicz, A. Tajti, J. Vázquez, F. Wang, J.D. Watts, C. Zhang, X. Zheng, and the integral packages MOLECULE (J. Almlöf and P.R. Taylor), PROPS (P.R. Taylor), ABACUS (T. Helgaker, H.J. Aa. Jensen, P. Jørgensen, and J. Olsen), and ECP routines by A. V. Mitin and C. van Wüllen. For the current version, see <http://www.cfour.de>
29. Matthews DA, Cheng L, Harding ME, Lipparini F, Stopkowicz S, Jagau T-C, Szalay PG, Gauss J, Stanton JF (2020) Coupled-cluster techniques for computational chemistry: the CFOUR program package. *J Chem Phys* 152(21):214108
  30. Herbst MF, Levitt A, Cancès E (2021) DFTK: a julian approach for simulating electrons in solids. *Proc JuliaCon Conf* 3:69
  31. Cancès E, Kemlin G, Levitt A (2021) Convergence analysis of direct minimization and self-consistent iterations. *SIAM J Matrix Anal Appl* 42(1):243–274
  32. Yamamoto Y, Nakatsukasa Y, Yanagisawa Y, Fukaya T (2015) Roundoff error analysis of the CholeskyQR2 algorithm. *Electron Trans Numer Anal* 44(01):306–326
  33. Hehre WJ, Ditchfield R, Pople JA (1972) Self-consistent molecular orbital methods. XII. Further extensions of Gaussian-type basis sets for use in molecular orbital studies of organic molecules. *J Chem Phys* 56(5):2257–2261
  34. Fletcher R (2013) *Practical methods of optimization*. Wiley, New Jersey
  35. Nottoli T, Gauss J, Lipparini F (2021) A black-box, general purpose quadratic self-consistent field code with and without cholesky decomposition of the two-electron integrals. *Mol Phys* 119(21–22):1974590
  36. Werner H-J, Knowles PJ (1985) A second order multiconfiguration SCF procedure with optimum convergence. *J Chem Phys* 82(11):5053–5063
  37. Werner H-J, Meyer W (1980) A quadratically convergent multiconfiguration-self-consistent field method with simultaneous optimization of orbitals and ci coefficients. *J Chem Phys* 73(5):2342–2356
  38. Jensen HJA, Ågren H (1986) A direct, restricted-step, second-order mc scf program for large scale ab initio calculations. *Chem Phys* 104(2):229–250
  39. Pulay P, Hamilton TP (1988) UHF natural orbitals for defining and starting MC-SCF calculations. *J Chem Phys* 88(8):4926–4933
  40. Tóth Z, Pulay P (2020) Comparison of methods for active orbital selection in multiconfigurational calculations. *J Chem Theor Comput* 16(12):7328–7341
  41. Gill PE, Murray W, Saunders MA, Wright MH (1987) Maintaining lu factors of a general sparse matrix. *Linear Algebra Appl* 88:239–270
  42. Saunders M LUSOL: Sparse LU for  $Ax=b$ ; freely available at <https://github.com/nwh/lusol>. Accessed 26 Apr 2023

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.