# Scalable block-tridiagonal eigensolvers in the context of electronic structure calculations

Alejandro LAMAS DAVIÑA [a], Xavier CARTOIXÀ [b] José E. ROMÁN [a]
[a] *D. Sistemes Informàtics i Computació, Universitat Politècnica de València, 46022 Valencia, Spain*
[b] *D. Enginyeria Electrònica, Universitat Autònoma de Barcelona, 08193 Bellaterra, Spain*

**Abstract.** In computer-based numerical simulations, some methods to determine the electronic and optical properties of semiconductor nanostructures, require computing the energies that correspond to the interior eigenvalues of a Hamiltonian matrix. We study the case in which the Schrödinger equation is expanded into a matrix that has a block-tridiagonal structure. Additionally, the matrix can have two extra non zero blocks in the corners due to periodic boundary conditions. Given that not the whole eigenspectrum is required, we choose to use projection methods to compute the necessary set of eigenvalues. The shift-and-invert Lanczos method requires to solve a linear system at each iteration. We have developed a parallel code that improves the scalability of this step by exploiting the block structure. Results show that, to solve these specific cases, this method offers better scalability when compared to a general-purpose solver such as MUMPS.

**Keywords.** electronic structure calculation, eigenvalue problem, shift-and-invert, cyclic reduction, Schur complement

## 1. Introduction and motivation

There is an increasing interest in methods that can provide an accurate picture of the electronic structure of large systems. This kind of computations allows the design of new devices that are essential for many emerging technologies.

In order to understand the electronic and optical properties of semiconductor hetero and nanostructures, it is imperative to know the allowed energy levels for electrons and their quantum state, completely determined by their corresponding wavefunctions. These quantities are provided by the Schrödinger equation, a linear partial differential equation (PDE) which, in its single particle (an electron from now on) time-independent version, reads

$$\hat{H}\psi(\mathbf{r}) = \left[-\frac{\hbar^2}{2m_0}\nabla^2 + V(\mathbf{r})\right]\psi(\mathbf{r}) = E\psi(\mathbf{r}), \tag{1}$$

where $\hat{H}$ is the Hamiltonian operator, $\hbar$ is the reduced Planck constant, $m_0$ is the free electron mass, $V(\mathbf{r})$ is the microscopic potential energy affecting the electron, and $\psi(\mathbf{r})$ and $E$ are the sought electron wavefunction and energy, respectively. When the unknown $\psi(\mathbf{r})$ is expanded as a linear combination of unknown coefficients $c_i$ times known basis functions $\phi_i(\mathbf{r})$,

$$\psi(\mathbf{r}) = \sum_i c_i \phi_i(\mathbf{r}), \tag{2}$$

the Schrödinger PDE is transformed into an algebraic (maybe generalized) eigenvalue problem.

Commonly used basis functions for the expansion of the Schrödinger equation include plane waves, often encountered in density functional theory codes [17]; atomic-like orbitals, giving rise to the family of empirical tight-binding methods (ETB) [19]; or the solution at the Brillouin zone center for the primitive cell of bulk materials, leading to the development of the $\mathbf{k} \cdot \mathbf{p}$ / Envelope Function Approximation (EFA) methods [15,14,3]. More precisely, the EFA provides a set of coupled PDEs that mimick the Schrödinger equation, but replacing the microscopic potential by the primitive cell average of some quantities. In addition, the EFA method can be mapped into ETB if the system of PDEs obtained in the EFA is discretized in a mesh coincident with the underlying Bravais lattice sites [7,8]. Popular options to solve the EFA equations are the finite difference method (FDM) and the finite element method (FEM). Although the number of basis functions is in principle infinite, in practice it is limited to a low figure (between 1-20) chosen on physical intuition grounds.

In a semiconductor heterostructure, the matrix resulting from the expansion in (2), will have the following characteristics:

- Each atom or primitive cell (in ETB) or discretization node (in EFA) will contribute with $N_b$ basis functions to the wavefunction, and with an $N_b \times N_b$ block to the matrix $A$ representing $\hat{H}$.
- The topology of the non-vanishing interactions between the atomic-like orbitals, in ETB, or the chosen stencil in FDM will determine which blocks in $A$ will have non-zero entries.

If, in addition, the semiconductor heterostructure has translational symmetry along the $x, y$ coordinates and varies along the $z$ axis (i.e., is effectively one-dimensional), the obtained $A$ matrix will be block-tridiagonal, with the possibility that non-zero blocks appear in the upper-left and lower-right corners if periodic boundary conditions are imposed (block cyclic tridiagonal structure).

The above one-dimensional case appears in the study of quantum wells, superlattices, design of chirped superlattices for quantum cascade lasers [10], Anderson localization in non-ideal superlattices, etc. Although a typical dimension of the resulting matrix may be of the order of $10^5 \times 10^5$ for one-dimensional structures hundreds of nanometers long, and a single solution is typically not too much time consuming, efficient solvers for the resulting block cyclic tridiagonal matrices are needed since in a typical application solutions must be sought for a large number of $\mathbf{k}$-points inside the first Brillouin zone, and then an outer loop of Poisson-Schrödinger self-consistency may be required.

As we just need a few energies corresponding to eigenvalues from the interior of the spectrum, close to the Fermi level , we use an iterative eigensolver that, by means of the shift-and-invert technique, solves a linear system in each iteration. These linear systems are solved with a direct method based on matrix factorization, that exploits the nonzero structure of the matrix to improve the scalability. In particular, we use a Schur complement approach combined with a parallel block-cyclic reduction.

The rest of the paper is organized as follows. Section 2 describes the method used for the eigenvalue computation. Section 3 provides details of the linear system solution used. Some computational results are given in section 4. Finally, we wrap up with some conclusions in section 5.

## 2. Shift-and-invert Arnoldi for interior eigenvalues

For a square matrix $A$ of order $n$, the standard eigenvalue problem formulated as

$$Ax = \lambda x, \tag{3}$$

has $n$ solutions $(\lambda, x)$, where $\lambda$ is a scalar (eigenvalue) and $x \neq 0$ an $n$-vector (eigenvector). If the matrix $A$ is Hermitian, then all eigenvalues are real, otherwise eigenvalues are complex in general.

The two major strategies to solve these eigenproblems are based on direct methods and on projection methods. If the matrix is sparse and only a few eigenpairs are required, projecting the eigenproblem on a low-dimensional subspace is usually a better option than the use of direct methods which first generate a condensed form of the matrix and provide all its eigenpairs. Also, when working with large sparse matrices it is desirable to preserve the sparsity of the matrix and the transformation to condensed form of the direct methods produces a fill-in with the consequent cost blow-up.

In this paper, we are considering matrices that have a block cyclic tridiagonal structure, where the blocks are not too large. In turn, these blocks are also sparse, but we will treat them as if they were fully populated. We employ projection methods because only a small amount of eigenvalues are required, and because direct methods are not well suited for the block cyclic tridiagonal structure when having nonzero corner blocks.

One of the projection methods for non-symmetric matrices is the Arnoldi algorithm. Given a matrix $A$ and an initial unit vector $v_1$, after $j$ steps the method computes matrices $V_j$ and $H_j$, where the columns of $V_j$ form an orthogonal basis of the Krylov subspace $\mathcal{K}_j(A, v_1) = \text{span}\{v_1, Av_1, A^2v_1, \ldots, A^{j-1}v_1\}$, and $H_j = V_j^* A V_j$ is the restriction of $A$ to this subspace. The method can be expressed by the recurrence

$$v_{j+1}h_{j+1,j} = w_j = Av_j - \sum_{i=1}^{j} h_{i,j}v_i, \tag{4}$$

where $h_{i,j}$ are the scalar coefficients obtained in the Gram-Schmidt orthogonalization of $Av_j$ with respect to $v_i$, $i = 1, 2, \ldots, j$, and $h_{j+1,j} = \|w_j\|_2$. The upper

Hessenberg matrix $H_j$ provides Ritz approximations to eigenpairs of $A$, $(\tilde{\lambda}_i, V_j y_i)$, being $(\tilde{\lambda}_i, y_i)$ eigenpairs of $H_j$, $i = 1, 2, \ldots, j$.

Since approximate eigenvalues may converge quite slowly, a practical implementation of Arnoldi must incorporate some restarting mechanism, such as the Krylov-Schur [20] method. This method is particularly suitable for computing exterior eigenvalues. However, in the case of interior eigenvalues, convergence is usually prohibitively slow, and a common alternative is the shift-and-invert spectral transformations [9] that consists in applying the Krylov method to matrix $(A - \sigma I)^{-1}$. Then, convergence to the eigenvalues closest to $\sigma$ will be fast, but the drawback is that it requires to implicitly handle a matrix inverse by solving linear systems at each iteration of the eigensolver. Most often direct solvers must be employed to guarantee the robustness of the method.

## 3. Scalable solution of linear systems for block cyclic tridiagonal matrices

SLEPc [12] is a software library for the solution of large-scale eigenvalue problems in parallel computers. Among others, it addresses the standard eigenvalue problem of (3) and can work with either real or complex arithmetic, in single or double precision, and it is not restricted to symmetric (Hermitian) problems. SLEPc has been employed successfully in many different application areas such as nuclear engineering [4] or plasma physics [16].

The library provides a collection of eigensolvers, most of which are based on the subspace projection paradigm described in §2. It includes a robust and efficient parallel implementation of Krylov-Schur, among other methods, and also provides built-in support for the shift-and-invert transformation to compute interior eigenvalues.

It offers the users the possibility to specify many parameters such as the number of eigenpairs to compute, the convergence tolerance or the dimension of the built subspace, both programmatically and in run-time.

SLEPc is built on top of PETSc (Portable, Extensible Toolkit for Scientific Computation, [2]), a parallel framework for the numerical solution of partial differential equations. PETSc uses the MPI paradigm, and encapsulates mathematical algorithms using object-oriented programming techniques in order to be able to manage the complexity of efficient numerical message-passing codes.

The solvers in PETSc and SLEPc have a neutral implementation with respect to the data-structure. They allow to do the computation with different matrix storage formats without needing to modify the code, and it is also possible to use matrices that are not stored explicitly by implementing some user-defined operations like the matrix-vector product.

One interesting characteristic of PETSc is that it offers the possibility to interface with third-party software libraries in a straightforward way in order to provide some functionality such as direct linear solvers and preconditioners, as has been done with MUMPS [1], for example. MUMPS is a very reliable library for factorization-based linear system solves for sparse matrices in parallel. Although MUMPS is quite efficient in general, it often suffers from bad scalability when many MPI processes are used. In this work, we want to demonstrate that for the particular case of matrices with block cyclic tridiagonal structure, it is possible to improve on MUMPS (at least from a certain number of processes) by implementing a customized linear solver that exploits such structure.

## *3.1. Parallel cyclic reduction*

Cyclic reduction is a well-known algorithm for the solution of tridiagonal linear systems that is equivalent to Gaussian elimination without pivoting [6]. Cyclic reduction is stable if $A$ is strictly diagonally dominant or symmetric positive definite, or in those cases where Gaussian elimination is stable with diagonal pivots [11].

The cyclic reduction algorithm is recursive. During the first of its two stages, it progresses reducing the number of rows with which it works. It classifies the rows in even-numbered and odd-numbered and, on each recursive step, it eliminates the even rows of the matrix in terms of the odd rows, halving them until a single row is left and a simple equation with a single unknown is trivially solved. Once the first unknown is solved, the algorithm starts the second stage in which it uses the current recursion level solution(s) to compute the adjacent even rows on the previous level, undoing the recursion.

It is possible to extend the method to the case where the coefficient matrix has a block-tridiagonal structure,

$$T = \begin{bmatrix} D_1 & U_1 & & & & \\ L_2 & D_2 & U_2 & & & \\ & L_3 & D_3 & U_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & L_{m-1} & D_{m-1} & U_{m-1} \\ & & & & L_m & D_m \end{bmatrix}. \qquad (5)$$

Block-cyclic reduction was first studied in [11], and in this case, the algorithm reduces the block-rows in a similar way as the cyclic reduction works with the rows. In order for block-cyclic reduction to be numerically reliable, it must be applied to matrices for which Gaussian elimination without pivoting is applicable, or more precisely, when pivoting is applied only within the diagonal blocks. Since during the first stage it is required to compute the inverse of the diagonal blocks, these blocks need to be non-singular for the algorithm to be able to progress, in the same way that the cyclic reduction cannot progress if a division by zero is hit (when a zero pivot is found).

The nice feature of the cyclic reduction algorithm is that is more amenable to parallelization than classical Gaussian elimination, because the task dependency graph has many independent tasks in each recursion level, that can be assigned to different processes, while classical Gaussian elimination is essentially sequential.

During the first stage, the even-numbered blocks are computed as

$$\begin{aligned}
\hat{D}_{2i} &= D_{2i}^{-1}, \\
\hat{L}_{2i} &= \hat{D}_{2i} L_{2i}, \\
\hat{U}_{2i} &= \hat{D}_{2i} U_{2i}, \\
\hat{b}_{2i} &= \hat{D}_{2i} b_{2i}.
\end{aligned} \tag{6}$$

Then, the off-diagonal blocks $\hat{L}_{2i}$ and $\hat{U}_{2i}$, and $\hat{b}_{2i}$ are sent to the process owning the adjacent odd-numbered row blocks to compute them

$$\begin{aligned}
\hat{D}_{2i-1} &= D_{2i-1} - L_{2i-1}\hat{U}_{2i-2} - U_{2i-1}\hat{L}_{2i}, \\
\hat{L}_{2i-1} &= -L_{2i-1}\hat{L}_{2i-2}, \\
\hat{U}_{2i-1} &= -U_{2i-1}\hat{U}_{2i}, \\
\hat{b}_{2i-1} &= b_{2i-1} - L_{2i-1}\hat{b}_{2i-2} - U_{2i-1}\hat{b}_{2i}.
\end{aligned} \tag{7}$$

The backward substitution stage starts computing the first block of unknowns with $x_1 = \hat{D}_1\hat{b}_1$. Then, $x_i$ is sent to the adjacent even-numbered row blocks and, on each recursive step, the corresponding parts of the solution are obtained with

$$x_{2i} = \hat{b}_{2i} - \hat{L}_{2i}x_{2i-1} - \hat{U}_{2i}x_{2i+1}. \tag{8}$$

The main computational cost of the algorithm lies in the first 3 operations of (6) and (7). Note that, in the case of solving multiple linear systems with the same matrix and different right-hand sides, these operations need not be recomputed. This is the case of the shift-and-invert Arnoldi method, that needs to invoke the linear solver in each iteration of the eigensolver.

### *3.2. Schur complement approach for the case of periodic boundary*

As mentioned in §1, if periodic boundary conditions are imposed, then the coefficient matrix is not purely block-tridiagonal, as in (5), but has additional non-zero blocks on the upper-right and lower-left corners, and hence the block-cyclic reduction method is not directly applicable.

The *Schur complement* [18, ch. 14] is a classical method that can be applied to the resolution of linear systems. Let $M$ be a square matrix of dimension $n$ partitioned as $M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$, and then perform a block Gaussian elimination on it. In our case, the matrix is block cyclic tridiagonal, with $m$ block-rows with blocks of size $k$ ($n = m \cdot k$), so we take $A$, $B$, $C$ and $D$ to be of dimensions $k \times k, k \times p, p \times k$ and $p \times p$, respectively (with $p = n - k$) as illustrated in Figure 1. In our implementation, $A$ and $B$ are stored on the first process, and $C$ and $D$ are partitioned among all the existing processes. Figure 1 also illustrates this data distribution, where we can see how only the non-zero blocks of $B$ are stored while the full size of $C$ is allocated. $D$ is also stored compacted, reducing the unused blocks to two.

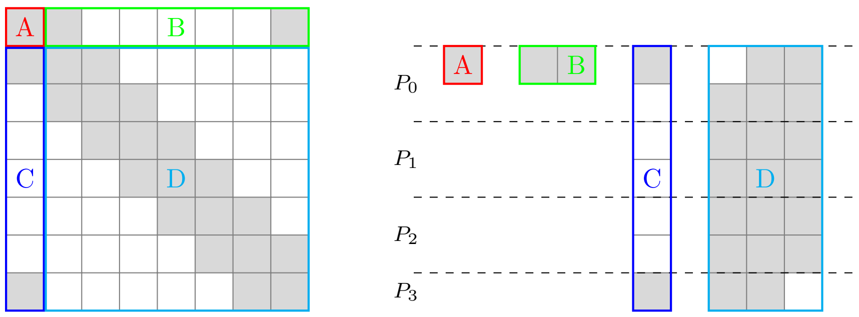

**Figure 1.** Scheme of the partitioning of a square matrix into four smaller matrices to solve a system of linear equations by using the Schur complement (left). Distribution of the four sub-matrices among several processes and its representation in memory (right).

The Schur complement of $M$ relative to $D$ is

$$S = M/D = A - BD^{-1}C. \tag{9}$$

Assuming that $D$ is non-singular, it is possible to solve the system of linear equations

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} e \\ f \end{bmatrix} \tag{10}$$

by solving

$$Sx = e - BD^{-1}f, \tag{11}$$

and once this small system is solved, the rest of the system can be solved in parallel with

$$Dy = f - Cx. \tag{12}$$

In our case, $D$ is a block-tridiagonal matrix that can be factored in parallel with the block-cyclic reduction algorithm. When computing the Schur complement (9), the $D^{-1}C$ operation implies a linear solve with multiple right-hand sides, that can be computed in parallel, and since $C$ has only two non-zero blocks on its edges, the computation involving the null blocks can be avoided during the solve stage of block-cyclic reduction, reducing the processing time. The remaining operations to compute the Schur complement are only done by the first process, as it stores the first block row of $M$ that includes $A$ and $B$. As $B$ also has only two non-zero blocks, the matrix multiplication $B$ times $D^{-1}C$ can be cheaply done by multiplying only these two blocks with the corresponding blocks of $D^{-1}C$; one of them is already stored in this first process and the other must be sent by the process with the highest rank.

In order to obtain $x$, the system (11) is solved in a similar way as (9) by computing $D^{-1}f$ in parallel with block-cyclic reduction. But this time no reduction in the operations is possible as $f$ does not have a zero pattern.

Once the first block of the solution is obtained, it must be sent from the first to the last process for them both to compute $f - Cx$, and after that, the block-cyclic reduction method can be used again to finally solve (12) in parallel.

## 4. Computational results

A set of experiments to measure the performance of the software were conducted in Tirant, a machine consisting of 256 JS20 blade computing nodes, each of them with two 64-bit PowerPC 970+ processors running at 2.2 GHz, and interconnected with a low latency Myrinet network.

The servers run SuSE Linux Enterprise Server 10 as operating system, and our software has been compiled in complex arithmetic and double precision with gcc 4.6.1 using SLEPc 3.7.1, PETSc 3.7.1, MUMPS 5.0.1-p1, and MPICH2 1.0.8p1 as the inter-process communication library.

As the nodes have four computational cores, a limit of four processes per node have been used during the executions.

In the experiments we compare the performance of a state of the art library such as MUMPS with our implementation of the Schur complement that uses the block-cyclic reduction algorithm to solve the block-tridiagonal system that involves $D$ of (9). We should remark that MUMPS stores (and works with) the full matrix (in blocked sparse format) while our software reduces the operations to the non-zero blocks (using dense storage), and except in the case of the sub-matrix $C$, it initially reduces the memory footprint to them.

Two matrices arising from the ETB parametrization [13] of a quantum cascade laser structure [5], and a 4/2 GaAs/AlAs superlattice with fluctuations in the layer widths have been used in the tests. Their dimensions are 82,400 (qcl) and 143,840 (anderson), respectively, and both have a block size of 20. In both cases, we instructed SLEPc to obtain 40 eigenvalues closest to the target value 1.5 eV (lowest energy states in the conduction band) with the shift-and-invert technique and a default tolerance of $10^{-8}$.

The relevant command-line options used in the experiments are:

- MUMPS: `-matload_block_size 20 -eps_nev 40 -eps_target 1.5 -st_type sinvert -st_pc_factor_mat_solver_package mumps`
- Schur complement: `-eps_nev 40 -eps_target 1.5 -st_type shell`

Additionally, for the runs with the anderson matrix, the value of the `-eps_ncv` parameter was increased and set to 128 in order to reduce the number of restarts and achieve a better performance.

Results of the executions can be seen in Figure 2. The plots show that the eigensolver scales linearly (up to 64 MPI processes) when using the custom linear solver based on Schur complement with block-cyclic reduction. In contrast, the scalability of MUMPS is more limited, and performance degrades with 16 processes or more. This can be attributed to the fact that MUMPS is based on a

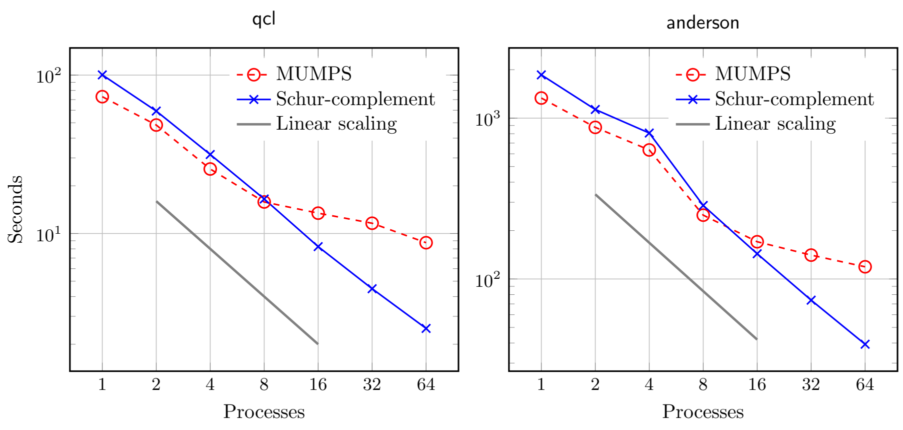

**Figure 2.** Total eigenproblem solve time to obtain 40 eigenvalues closest to 1.5 for the qcl and anderson matrices using 4 processes per node.

classical scheme of factorization followed by triangular solves, where the latter operation is inherently sequential and results in bad scalability. The cyclic reduction scheme rearranges the operations in such a way that there is more opportunity for a larger degree of parallelism.

## 5. Conclusions and future work

We have implemented a parallel linear solver that exploits the block cyclic tridiagonal structure of the coefficient matrix, and have analyzed its scalability when used within an iterative eigensolver in the context of electronic structure calculations. In this type of applications, it is important to obtain the solution very fast, especially when the process involves a self-consistency loop that requires solving an eigenvalue problem in each iteration. Our results illustrate that exploiting the matrix structure in the solver may provide some advantage, such as a better scalability, although it implies a higher development effort compared to using general-purpose numerical libraries.

Currently we treat the blocks of the block cyclic tridiagonal matrices as dense, therefore as a future work it remains to take profit of the sparsity of the blocks. Another possible research direction would be to expand the software to work with tridiagonal blocks that appear in some fast Poisson solvers.

*Acknowledgements* The authors would like to acknowledge the support by Agencia Estatal de Investigación under project TIN2016-75985-P, which includes European Commission ERDF funds, and the Spanish Ministerio de Economía y Competitividad under project No. TEC2015-67462-C2-1-R (MINECO/FEDER). Also, this work has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 696656, and the Department d'Universitats, Recerca i Societat de la Informació of the Generalitat de Catalunya under contract 2014-SGR-384. Alejandro Lamas Daviña was supported by the Spanish Ministry of Education, Culture and Sport through grant FPU13-06655. The authors are also grateful for the resources provided by the Spanish Supercomputing Network.

## References

[1] P. R. Amestoy, I. S. Duff, and J.-Y. L'Excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Comput. Methods Appl. Mech. Eng.*, 184(2–4):501–520, 2000.

[2] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. Gropp, D. Karpeyev, D. Kaushik, M. Knepley, L. Curfman McInnes, K. Rupp, B. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.7, Argonne National Laboratory, 2016.

[3] G. Bastard. *Heterojunctions and Semiconductor Superlattices.* Springer-Verlag New York, New York, Ny, 2nd edition, 1986.

[4] A. Bernal, J.E. Roman, R. Mir, and G. Verd. Assembly discontinuity factors for the neutron diffusion equation discretized with the finite volume method. application to bwr. *Annals of Nuclear Energy*, 97:76–85, 2016.

[5] S. Blaser, M. Rochat, L. Ajili, M. Beck, J. Faist, H. Beere, G. Davies, E. Linfield, and D. Ritchie. Terahertz interminiband emission and magneto-transport measurements from a quantum cascade chirped superlattice. *Physica E*, 13(2–4):854–857, 2002.

[6] B. L. Buzbee, G. H. Golub, and C. W. Nielson. On direct methods for solving Poisson's equations. *SIAM J. Numer. Anal.*, 7(4):627–656, 1970.

[7] Y. C. Chang. Bond-orbital models for superlattices. *Phys. Rev. B*, 37(14):8215–8222, 1988.

[8] G. T. Einevoll and Y. C. Chang. Effective bond-orbital model for acceptor states in semiconductors and quantum dots. *Phys. Rev. B*, 40(14):9683–9697, 1989.

[9] T. Ericsson and A. Ruhe. The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems. *Math. Comp.*, 35(152):1251–1268, 1980.

[10] J. Faist, F. Capasso, D. L. Sivco, C. Sirtori, A. L. Hutchinson, and A. Y. Cho. Quantum cascade laser. *Science*, 264(5158):553–556, 1994.

[11] D. Heller. Some aspects of the cyclic reduction algorithm for block tridiagonal linear systems. *SIAM J. Numer. Anal.*, 13(4):484–496, 1976.

[12] V. Hernandez, J. E. Roman, and V. Vidal. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Trans. Math. Software*, 31(3):351–362, 2005.

[13] J. M. Jancu, R. Scholz, F. Beltram, and F. Bassani. Empirical spds* tight-binding calculation for cubic semiconductors: General method and material parameters. *Phys. Rev. B*, 57(11):6493–6507, 1998.

[14] E. O. Kane. The $\mathbf{k}\cdot\mathbf{p}$ method. In R. K. Willardson and A. C. Beer, editors, *Semiconductors and Semimetals*, volume 1, pages 75–100. Academic, New York, 1966.

[15] J. M. Luttinger and W. Kohn. Motion of electrons and holes in perturbed periodic fields. *Phys. Rev.*, 97:869–883, Feb 1955.

[16] F. Merz, C. Kowitz, E. Romero, J. E. Roman, and F. Jenko. Multi-dimensional gyrokinetic parameter studies based on eigenvalues computations. *Comput. Phys. Commun.*, 183(4):922–930, 2012.

[17] M. C. Payne, M. P. Teter, D. C. Allan, T. A. Arias, and J. D. Joannopoulos. Iterative minimization techniques for *ab initio* total-energy calculations: molecular dynamics and conjugate gradients. *Rev. Mod. Phys.*, 64:1045–1097, Oct 1992.

[18] Y. Saad. *Iterative Methods for Sparse Linear Systems.* SIAM Publications, 2nd edition, 2003.

[19] J. C. Slater and G. F. Koster. Simplified LCAO method for the periodic potential problem. *Phys. Rev.*, 94(6):1498–1524, 1954.

[20] G. W. Stewart. A Krylov–Schur algorithm for large eigenproblems. *SIAM J. Matrix Anal. Appl.*, 23(3):601–614, 2001.