Zuse Institute Berlin

Takustrasse 7
D-14195 Berlin-Dahlem
Germany

PHILIPP GUTSCHE, MATTHIAS LÄUTER AND FRANK SCHMIDT

Parameter-dependent Parallel Block Sparse Arnoldi and Döhler Algorithms on Distributed Systems

This research was carried out at Zuse Institute Berlin within the joint bridge project *Parameter-dependent Parallel Block Sparse Arnoldi and Döhler Algorithms on Distributed Systems* of the research groups *Supercomputer Algorithms and Consulting* and *Computational Nano Optics*.

ZIB Report 16-15 (January 2016)

Zuse Institute Berlin
Takustrasse 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

e-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Parameter-dependent Parallel Block Sparse Arnoldi and Döhler Algorithms on Distributed Systems

Philipp Gutsche*, Matthias Läuter and Frank Schmidt

January 2, 2016

Abstract

We summarize the basics and first results of the analyses within our ZIB Bridge Project and give an outlook on further studies broadening the usage of hardware acceleration within the Finite Element Method (FEM) based solution of Maxwell's equations.

1 Solving the Generalized Eigenvalue Problem

The main focus of our project is on the solution of the eigenvalue problem originating from Maxwell's equations in nano-optics. The basic example of such problems are so-called resonance problems [18]. These yield generalized eigenvalue problems, of which the formulation and different solution approaches are discussed in the following. We start by stating the problem and introducing a typical example, namely nanoholes in photonic crystals [1]. Furthermore, we describe three different algorithms to solve the generalized eigenvalue problem: the standard Arnoldi iteration, the recently introduced [16] and extended [8] FEAST algorithm and a Döhler algorithm [2].

1.1 Problem Formulation

In the FEM formulation [14] for Maxwell's equations [4], we face the problem of solving the generalized eigenvalue problem:

$$Au = \lambda Bu \tag{1}$$

with the matrices $A, B \in \mathbb{C}^{N\times N}$ originating from the stiffness and mass matrices, the solution vector $u \in \mathbb{C}^N$ and the eigenvalue $\lambda \in \mathbb{C}$. The square root of the latter corresponds to the so-called eigenfrequencies, i.e. the frequencies of the physically relevant eigenmodes of the specific setup, for resonance problems in electrodynamics [18].

Moreover, the eigenvalue problem is one major building block in domain decomposition approaches [19] extended to mixed FEM and Fourier Modal Method (FMM) [10] computations. Here, a large computational domain is split into non-trivially structured and simpler domains which are coupled to one another. The FMM relies on the represenation of modes in Fourier basis and exhibits lower computational effort for rather simple structures whereas the accuracy and flexibility of the FEM is required for complex geometries.

The at first very simple eigenvalue problem can be numerically challenging as already stated by J. H. Wilkinson 50 years ago [20]:

> *"The solution of the algebraic eigenvalue problem has for long had a particular fascination for me because it illustrates so well the difference between what might be termed classical mathematics and practical numerical analysis. The eigenvalue problem has a deceptively simple formulation and the background theory has been known for many years; yet the determination of accurate solutions presents a wide variety of challenging problems."*

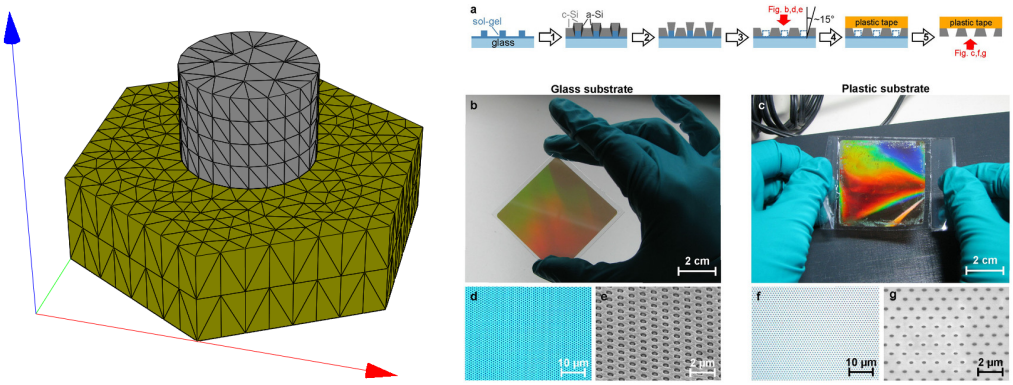
An example of the wide variety of problems is the aforementioned FEM for highly accurate simulations of nanophotonic devices such as a photonic crystal described in the following section.

*gutsche@zib.de

1.2 Example: Photonic Crystal Nanohole

As an state-of-the-art example for nanophotonical FEM simulation, we use a slab consisting of periodically arranged holes. This yields a so-called photonic crystal with interesting properties such as optical band gaps [6]. In order to study the abilities of this device and to tailor its properties, accurate and fast numerical simulations are required.

For photonic crystals, band structure analysis is a common tool. These band structures are obtained from solving the aforementioned resonance problem of Maxwell's equations with adequate Bloch-periodic conditions. Here, we are more interested in the numerical analysis of this nano-optical device and restrict ourselves of the most basic setup depicted in Fig. 1 (a). In recent experimental studies using imprint lithography, photonic crystal slabs were fabricated on a large scale [1].



(a) mesh (b) fabrication (from [1])

Figure 1: Discretized mesh for the FEM simulations (a) of a photonic crystal nanohole slab. The device is based on a hexagonal unit cell and composed of a slab with refractive index $n = 3.55$ surrounded by glass with $n = 1.5$. Recent experiments are able to fabricate these structures at high accuracy (b) using imprint lithography [1].

1.3 Arnoldi Iteration

The standard approach for solving the generalized eigenvalue problem (1) is the Arnoldi iteration. This algorithm is commonly implemented in e.g. Intel's Math Kernel Library (MKL) [5] through the ARnoldi PACKage (ARPACK) [12].

The implicitly restarted Arnoldi method (IRAM) is based on Krylov subspaces which are spanned by the power iteration of $A - \sigma B$, where σ is a shift of the eigenvalue spectrum. The basis vectors of the n-th Krylov subspace are an approximation of the n largest eigenvalues. With the help of the shift σ these are shifted to the desired range of eigenvalues. Accordingly, the IRAM computes the n closest eigenvalues to σ.

The basic algorithm is outlined in Fig. 2 [11]. ARPACK provides a so-called reverse communication interface. We use the MKL, specifically the PARallel sparse DIrect SOlver (PARDISO) [9], to solve the underlying large sparse systems of linear equations.

- *Start*: Build a length m Arnoldi factorization $MV_m = V_m H_m + f_m e_m^T$ with a random starting vector v_1, the shifted matrix $M = A - \sigma B$, the residual vector f_m, $V_m^H V_m = I_m$ and an $m \times m$ upper Hessenberg matrix H_m.
- *Iteration*: Until convergence
  - Compute the eigenvalues $\{\lambda_j : j = 1, 2, ..., m\}$ of H_m. Sort these eigenvalues according to the user selection criterion into a wanted set $\{\lambda_j : j = 1, 2, ..., k\}$ and an unwanted set $\{\lambda_j : j = k+1, k+2, ..., m\}$.
  - Perform $m - k = p$ steps of the QR iteration with the unwanted eigenvalues as shifts to obtain $H_m Q_m = Q_m H_m^+$.
  - *Restart*: Postmultiply the length m Arnoldi factorization with the matrix Q_k consisting of the leading k columns of Q_m to obtain the length k Arnoldi factorization $AV_m Q_k = V_m Q_k H_k^+ + f_k^+ e_k^T$, where H_k^+ is the leading principal submatrix of order k for H_m^+. Set $V_k \leftarrow V_m Q_k$.
  - Extend the length k Arnoldi factorization to a length m factorization.

Figure 2: Summary of the Implicitly Restarted Arnoldi Method in ARPACK (adapted from [11]).

1.4 FEAST Eigenvalue Solver

The recently introduced FEAST algorithm for the solution of a generalized eigenvalue problem is based on contour integration and density matrices in quantum mechanics [16]. Its basic formulation is summarized in Fig. 3: Instead of computing the closest eigenvalues to a given starting value as the Arnoldi iteration, FEAST determines all eigenvalues in a given contour in the complex plane [Fig. 4 (a)]. The first formulation of this solver has been extended to non-Hermitian problems [8], enabling its application in general FEM-based solutions of Maxwell's equations.

FEAST is designed in such a way that it enables three levels of parallelization:

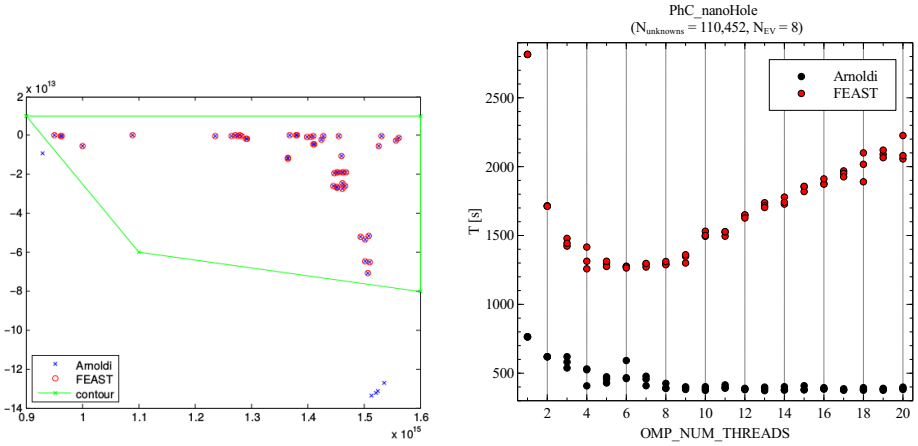
- Linear systems with multiple right hand sides can be solved in parallel.
- Linear systems of contour points are independent and thus can be computed in parallel.
- Division of the contour into smaller parts and computing these parts in parallel.

In order to analyze the applicability of the FEAST algorithm to the numerical solution of the eigenvalue problem given by Maxwell's resonance problem, we implement the first level of parallelization [Fig. 4 (b)]. We find that FEAST is well suited for solving Maxwell's equations, especially when detailed control of the search interval in the complex plane is required, e.g. in high-Q devices where the ratio of the real and imaginary part of the eigenvalues is high [13] or in photonic crystals when discrete eigenvalues close to the continuum above the light line are searched for [6].

However, in order to achieve reasonable performance our results show that all three level, especially the inpendence of the quadrature nodes on the contour, have to be taken into account. With the Arnoldi iteration only an overlapping heuristical separation of the contour could be computed in parallel by choosing more or less randomly distributed starting points. This makes the FEAST algorithm beneficial for setups in which a high number of eigenvalues in an exactly known search interval are required.

- *Start*: Select random subspace $Y_{m_0} = \{y_1, y_2, ...y_{m_0}\}_{n \times m_0} (n >> m_0 \geq m)$ with $\dim(A) = n$, an estimate of the number of eigenvalues in the contour m_0 and the desired number of eigenvalues m
- *Iteration*: Until convergence
  - Compute $Q_{m_0} = \rho(B^{-1}A)Y_{m_0}$
  - Orthogonalize Q_{m_0}
  - Compute $A_Q = Q_{m_0}^H A Q_{m_0}$ and $B_Q = Q_{m_0}^H B Q_{m_0}$
  - Solve $A_Q W = B_Q W \Lambda_Q$ with $W^H B_Q W = I_{m_0 \times m_0}$
  - Compute $Y_{m_0} = Q_{m_0} W$
  - Check convergence of Y_{m_0} and $\Lambda_{Q_{m_0}}$ for the m wanted eigenvalues

Figure 3: Summary of the FEAST algorithm (adapted from [8]).



(a) complex eigenvalues (b) run time

Figure 4: Comparison of the Arnoldi iteration and FEAST algorithms for the example of a photonic crystal nanohole slab. The complex eigenvalues (a) are found by the Arnoldi iteration (blue crosses) as closest neighbours of a specified starting value. The FEAST algorithm (red circles) computes the eigenvalues in a specified contour (green line) given by contour points (green crosses). The run time (b) on a standard CPU with low-level parallelization using threads (OpenMP) shows that all levels of parallelization of FEAST have to be taken into account: In this study only the lowest level, namely the LU-factorizations for the contour points and multiple right hand sides, is parallelized. The second level of parallelization, parallelizing the solution process for the contour points, would result in a similar run time as the Arnoldi iteration since, here, four contour points are used and FEAST is roughly four times slower than the Arnoldi iteration. Further speed-up could be achieved by dividing the contour into smaller parts and compute them in parallel.

1.5 Döhler Algorithm

A third possiblity to solve the generalized eigenvalue problem is the so-called Döhler algorithm which can be used in multigrid methods [2]. Since the basis of Schur vectors has in general a better condition than the basis of eigenvectors, the Döhler algorithm aims at solving a projected Schur problem which is small and dense rather than the solution of the large sparse matrix. The basic formulation is given in Fig. 5.

As a starting point an approximation of the Schur vectors is required. In the case of the resonance problem of Maxwell's equations this basis might be given by a previously computed solution with slightly different parameters. This makes the Döhler algorithm suitable for small perturbations of geometrical or material parameters as in e.g. reduced basis methods [17].

- *Start*: Choose $U \in \mathbb{C}^{N\times q}$ with $U^H BU = I$ and set $P = R = -C^{-1}(AU - BUT(U)) \in \mathbb{C}^{N\times q}$ with the desired number of eigenvalues q
- *Iteration*: Until convergence
  - Set $W = (UP)$
  - Compute $Q \in \mathbb{C}^{2q\times 2q}$ and $T \in \mathbb{C}^{2q\times 2q}$ from $(W^H AW)Q = (W^H BW)QT$ with $Q^H(W^H BW)Q = I$. T is an upper triangular matrix and its diagonal elements λ_j are sorted with respect to $\mathrm{Re}\,(\lambda_1) \leq ... \leq \mathrm{Re}\,(\lambda_{2q})$
  - Set $(US) = WQ$
  - Compute $R = -C^{-1}(AU - BUT(U))$
  - Solve for $X \in \mathbb{C}^{q\times q}$ in $T(S)X - XT(U) = -S^H(AR - BRT * U))$
  - Set $P = R + SX$

Figure 5: Summary of the Döhler algorithm (adapted from [2]).

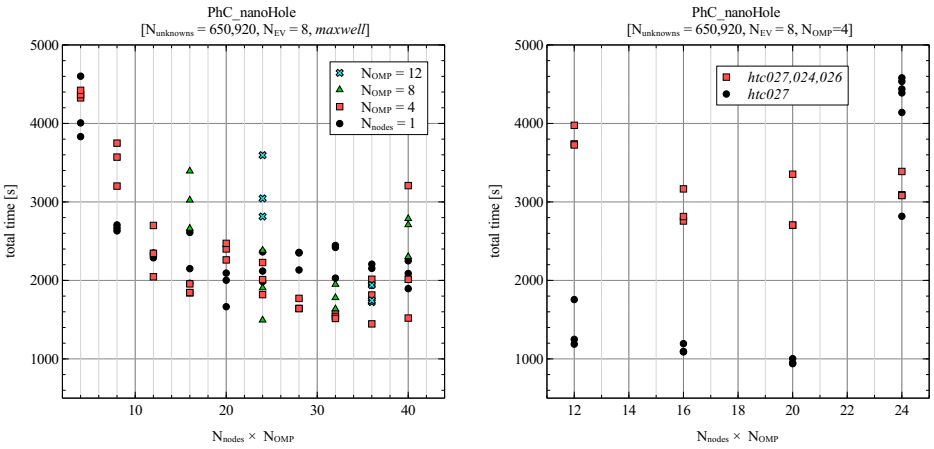
2 Using Hardware Acceleration

In this section, we present two different approaches to parallelize the solution of the generalized eigenvalue problem introduced in the previous section. Specifically, we focus on the LU factorization of the large sparse matrices given by a FEM-based formulation of the resonance problem of Maxwell's equations. First, we study the basic performance of the recently introduced Cluster Pardiso solver of Intel's MKL [7]. Second, we use Many Integrated Core (MIC) processors [5] with high numbers of possible OpenMP threads in offload and native mode.

2.1 Cluster Pardiso

Instead of aiming at threading performance on a single machine for the solution of an eigenvalue problem, one might want to make use of a large cluster of various computation nodes. This can be done in the framework of the Message Passing Interface (MPI) where communication between different nodes is standardized. Typically, these nodes possess rather small memory ressources whereas FEM matrices might be quite large. Accordingly, a distributed assembling and storage process is preferable. The recently introduced parallel direct solver for clusters [7] offers the possibility to deal with distributed matrices.

Here, we analyse the basic properties of this implementation of the solution of the generalized eigenvalue problem. We focus on its use within the standard Arnoldi iteration and show that performance on both a single machine [Fig. 6 (a)] and on a cluster [Fig. 6 (b)] is promising. We analyse different numbers of MPI nodes N_{nodes} where each node uses N_{OMP} OpenMP threads. Our findings show that both the threaded parallelized as well as MPI parallelized implementation of PARDISO show similar perfomance on a single machine. Furthermore, the solution of non-distributed matrices are slower on a cluster as expected due to data transfer between the nodes. However, this transfer could be reduced by assembling and storing the desired matrix distributively.



(a) single machine (b) cluster

Figure 6: Performance of the parallel direct solver for clusters [7] on a single machine (a) and a cluster with a non-distributed matrix (b). Thread-based and MPI-node-based parallelization of PARDISO shows similar performance (a) making it a promising candidate for further parallelization using distributively stored matrices. The slow down of the analyzed cluster (b) is due to time of data transfer between the nodes.

2.2 Many Integrated Core Processors

An easier accessible alternative to Graphical Processing Units (GPUs) which require sophisticated expertise in the hardware specific programming language are so-called Many Integrated Core (MIC) coprocessors [5]. These are built out of typically dozens of cores. Including hyper threading this results in e.g. 240 accesible OpenMP threads on Intel Xeon Phi platforms which can be used with previously existing CPU-based code.

The usage of Intel's MKL on these architectures is based on three different possibilities [3]:

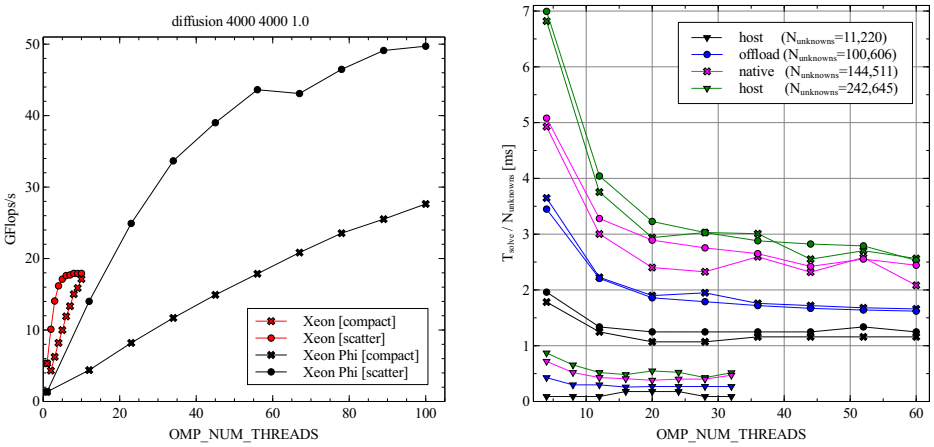
- *automatic offload*: no changes are required and the MKL decides heuristically which portions of the code are offloaded to the MIC.
- *compiler-assisted offload*: only those portions of the code are offloaded to the MIC which are specified by the user.
- *native mode*: the whole program runs natively on the MIC.

Although automatic offload is easily to use it is only implemented for dense matrices resulting in no performance speed-up for the computation involving large sparse matrices as in the solution of Maxwell's equations with the FEM. On the other hand, compiler-assisted offload is much more flexible since it allows to offload as much computation as decided by the user. Nevertheless, its draw-back is that only full calls to Intel's MKL functions can be offloaded. There is no possibility to offload only certain portions of the code as it is done in the automatic offload mode. Lastly, it is possible to use the MIC as a standard Linux-based machine and run a program in native mode after certain requirements in the compilation process are fulfilled.

As an illustrative example we show the capabilities of a MIC coprocessor with the help of the solution of a diffusion problem [Fig. 7 (a)]. It can be clearly seen, that the MIC exhibits bad performance for a small number of threads. However, due to the high number of possible threads the standard CPU (host) is outperformed for highly parallel computations. In this example, up to 50 GFlops/s are reached for 100 threads which is much larger than the maximal number of operations of 18 GFlops/s on the host.

Analyzing our example of a photonic crystal nanohole with the Arnoldi iteration and for various problem sizes [Fig. 7 (b)] reveals a draw-back of the MIC: here, only problems of dimensions of up to 250 thousand unknowns can be offloaded and natively run on the MIC since only 16 GB of memory are accessible. That is why, the solution requires much more time in both compiler-assisted offload and native mode even for large numbers of threads. This is due to early saturation of the speed-up of PARDISO at around 20 threads which is not enough for the MIC to outperform its host CPU. Nevertheless, we show

that the offload mode performs similar to a fully native execution due to a fast data transfer from host to mic.



(a) diffusion problem (b) photonic crystal nanohole slab

Figure 7: Capabilities of a MIC coprocessor in comparison with its host CPU. Although MIC coprocessors are slower for equal number of threads on host and MIC, they outperform CPUs for large number of threads due to their high abilities for parallelization. This is illustrated with the help of a diffusion problem (a) and two different KMP_AFFINITY options to distribute the threads on the machine, namely scatter and compact. When using a MIC for the solution of the resonance problem of a photonic crystal nanohole slab with the Arnoldi iteration (b), it cannot outperform its host since the implementation of PARDISO within Intel's MKL does not speed-up significantly for more than 20 threads. As shown in (a) this would be the regime from which on speed-up due to the high parallelization on the MIC is expected.

3 Conclusion

In conclusion, we analyzed the generalized eigenvalue problem for the solution of Maxwell's equations with the FEM and the determination of eigenmodes in its according resonance problem. With the help of an state-of-the-art example of a photonic crystal nanohole slab [1], we studied different algorithms to solve the eigenvalue equation. These are the standard Arnoldi iteration [12], the recently introduced FEAST [8] and a Döhler algorithm [2].

We investigated the performance of the Arnoldi iteration algorithm when using the recently released parallel direct solver for clusters [7] and found it to be promising when extended to matrices distributively stored on a cluster. Furthermore, we studied a MIC platform and compared run times on the host CPU, the coupled system of CPU and MIC in compiler-assisted offload mode and execution times in native mode running completely on the MIC.

Our results show that data offloading times to the MIC are neglible compared to the solution time itself when compared to the native execution. However, the relatively small memory resources on the MIC limit its applicability for large sparse matrices as given in the FEM for Maxwell's equations. Accordingly, the usage of a MIC coprocessor as several nodes within the parallel direct solver for clusters might be promising and could be based on unified programming models for intra- and inter-node offloading [15] on clusters consisting of both CPUs and MICs.

References

[1] C. Becker, P. Wyss, D. Eisenhauer, J. Probst, V. Preidel, M. Hammerschmidt, and S. Burger. 5×5 cm^2 Silicon Photonic Crystal Slabs on Glass and Plastic Foil Exhibiting Broadband Absorption and High-intensity Near-fields. *Sci. Rep.*, 4, 2014.

[2] T. Friese. *Eine Mehrgitter-Methode zur Lösung des Eigenwertproblems der komplexen Helmholtzgleichung.* PhD thesis, Freie Universität Berlin, 1998.

[3] Intel. *Intel Math Kernel Library for Linux OS - User's Guide.* MKL 11.3, Revision: 046, Chapter 10.

[4] J. D. Jackson. *Classical Electrodynamics.* John Wiley and Sons, 3rd edition, 1998.

[5] J. Jeffers and J. Reinders. *Intel Xeon Phi Coprocessor High-Performance Programming.* Elsevier Science, 2013.

[6] J. Joannopoulos, S. Johnson, J. Winn, and R. Meade. *Photonic Crystals: Molding the Flow of Light (Second Edition).* Princeton University Press, 2011.

[7] A. Kalinkin and K. Arturov. Asynchronous Approach to Memory Management in Sparse Multifrontal Methods on Multiprocessors. *Applied Mathematics*, 2013, 2013.

[8] J. Kestyn, E. Polizzi, and P. T. P. Tang. FEAST Eigensolver for non-Hermitian Problems. *arXiv preprint arXiv:1506.04463*, 2015.

[9] A. Kuzmin, M. Luisier, and O. Schenk. Fast Methods for Computing Selected Elements of the Greens Function in Massively Parallel Nanoelectronic Device Simulations. In F. Wolf, B. Mohr, and D. Mey, editors, *Euro-Par 2013 Parallel Processing*, volume 8097 of *Lecture Notes in Computer Science*, pages 533–544. Springer Berlin Heidelberg, 2013.

[10] A. Lavrinenko, J. Lægsgaard, N. Gregersen, F. Schmidt, and T. Søndergaard. *Numerical Methods in Photonics*, chapter The Modal Method. Optical Sciences and Applications of Light. Taylor & Francis, 2014.

[11] R. Lehoucq, D. Sorensen, and C. Yang. ARPACK Users' Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods. *Software Environ. Tools*, 6, 1997.

[12] R. B. Lehoucq. Implicitly restarted Arnoldi methods and subspace iteration. *SIAM Journal on Matrix Analysis and Applications*, 23(2):551–562, 2001.

[13] B. Maes, J. Petráček, S. Burger, P. Kwiecien, J. Luksch, and I. Richter. Simulations of high-Q optical nanocavities with a gradual 1D bandgap. *Optics express*, 21(6):6794–6806, 2013.

[14] P. Monk. *Finite Element Methods for Maxwell's Equations.* Numerical Mathematics and Scientific Computation. Clarendon Press, 2003.

[15] M. Noack, F. Wende, T. Steinke, and F. Cordes. A unified programming model for intra-and inter-node offloading on Xeon Phi clusters. In *High Performance Computing, Networking, Storage and Analysis, SC14: International Conference for*, pages 203–214. IEEE, 2014.

[16] E. Polizzi. Density-matrix-based algorithm for solving eigenvalue problems. *Physical Review B*, 79(11):115112, 2009.

[17] J. Pomplun. *Reduced basis method for electromagnetic scattering problems.* PhD thesis, Freie Universität Berlin, Germany, 2010.

[18] J. Pomplun, S. Burger, L. Zschiedrich, and F. Schmidt. Adaptive finite element method for simulation of optical nano structures. *physica status solidi (b)*, 244(10):3419–3434, 2007.

[19] A. Schädle, L. Zschiedrich, S. Burger, R. Klose, and F. Schmidt. Domain decomposition method for Maxwells equations: scattering off periodic structures. *Journal of Computational Physics*, 226(1):477–493, 2007.

[20] J. H. Wilkinson, J. H. Wilkinson, and J. H. Wilkinson. *The algebraic eigenvalue problem*, volume 87. Clarendon Press Oxford, 1965.