On Large-Scale Diagonalization Techniques for the Anderson Model of Localization[*]

Olaf Schenk[†] Matthias Bollhöfer[‡] Rudolf A. Römer[§]

Abstract

We propose efficient preconditioning algorithms for an eigenvalue problem arising in quantum physics, namely the computation of a few interior eigenvalues and their associated eigenvectors for large-scale sparse real and symmetric indefinite matrices of the Anderson model of localization. We compare the Lanczos algorithm in the 1987 implementation by Cullum and Willoughby with the shift-and-invert techniques in the implicitly restarted Lanczos method and in the JACOBI–DAVIDSON method. Our preconditioning approaches for the shift-and-invert symmetric indefinite linear system are based on maximum weighted matchings and algebraic multilevel incomplete LDL^T factorizations. These techniques can be seen as a complement to the alternative idea of using more complete pivoting techniques for the highly ill-conditioned symmetric indefinite Anderson matrices. We demonstrate the effectiveness and the numerical accuracy of these algorithms. Our numerical examples reveal that recent algebraic multilevel preconditioning solvers can accelerate the computation of a large-scale eigenvalue problem corresponding to the Anderson model of localization by several orders of magnitude.

keywords. Anderson model of localization, large-scale eigenvalue problem, Lanczos algorithm, JACOBI–DAVIDSON algorithm, Cullum–Willoughby implementation, symmetric indefinite matrix, multilevel preconditioning, maximum weighted matching

AMS. 65F15, 65F50, 82B44, 65F10, 65F05, 05C85

1 Introduction

One of the hardest challenges in modern eigenvalue computation is the numerical solution of large-scale eigenvalue problems, in particular those arising from quantum physics such as, e.g., the Anderson model of localization (see section 3 for details). Typically, these problems require the computation of some eigenvalues and eigenvectors for systems which have up to several million unknowns due to their high spatial dimensions. Furthermore, their underlying structure involves random perturbations of matrix elements which invalidates simple preconditioning approaches based on the graph of the matrices. Moreover, one is often interested in finding some eigenvalues and associated eigenvectors in the interior of the spectrum. The classical Lanczos approach [53] has led to eigenvalue algorithms [18, 19] that are, in principle, able to compute these eigenvalues using only a small amount of memory. More recent work on implicitly restarted Lanczos techniques [44] has accelerated these methods significantly, yet to be fast one needs to combine this approach with

[*]Received by the editors August 5, 2005; accepted for publication (in revised form) January 31, 2006; published electronically June 19, 2006. This work was supported by the Swiss Commission for Technology and Innovation (CTI) under grant 7036 ENS-ES and by the EPSRC under grant EP/C007042/1.

[†]Department of Computer Science, University of Basel, Klingelbergstrasse 50, CH-4056 Basel, Switzerland (olaf.schenk@unibas.ch).

[‡]Institute of Computational Mathematics, TU Braunschweig, D-38106 Braunschweig, Germany (m.bollhoefer@tu-bs.de). The work of this author was supported by the DFG research center MATHEON "Mathematics for Key Technologies" in Berlin.

[§]Centre for Scientific Computing and Department of Physics, University of Warwick, Coventry CV4 7AL, UK (r.roemer@warwick.ac.uk).

shift-and-invert techniques; i.e., in every step one has to solve a shifted system of type $A - \sigma I$, where σ is a shift near the desired eigenvalues and $A \in \mathbb{R}^{n,n}$, $A = A^T$ is the associated matrix. In general, shift-and-invert techniques converge rather quickly, which is in line with the theory [53]. Still, a linear solver is required to solve systems $(A - \sigma I)x = b$ efficiently with respect to time and memory. While implicitly restarted Lanczos techniques [44] usually require the solution of the system $(A - \sigma I)x = b$ to maximum precision, and thus are mainly suited for sparse direct solvers, the Jacobi–Davidson method has become an attractive alternative [66], in particular when dealing with preconditioning methods for linear systems.

Until recently, sparse symmetric indefinite direct solvers were not as efficient as symmetric positive definite solvers, and this might have been one major reason why shift-and-invert techniques were not able to compete with traditional Lanczos techniques [29], in particular because of memory constraints. With the invention of fast matchings-based algorithms [51], which improve the diagonal dominance of linear systems, the situation has dramatically changed and the impact on preconditioning methods [7], as well as the benefits for sparse direct solvers [60, 62], has been recognized. Furthermore, these techniques have been successfully transferred to the symmetric case [24, 26], allowing modern state-of-the-art direct solvers [59, 60, 62] to be orders of magnitudes faster and more memory efficient than ever, finally leading to symmetric indefinite sparse direct solvers that are almost as efficient as their symmetric positive definite counterparts. Recently this approach has also been utilized to construct incomplete factorizations [40] with similar dramatic success. For a detailed survey on preconditioning techniques for large symmetric indefinite linear systems, the interested reader should consult [5, 6].

2 Numerical approach for large systems

In the present paper we combine the above mentioned advances with inverse-based preconditioning techniques [10]. This allows us to find interior eigenvalues and eigenvectors for the Anderson problem several orders of magnitudes faster than traditional algorithms [18, 19] while still keeping the amount of memory reasonably small.

Let us briefly outline our strategy. We will consider recent novel approaches in preconditioning methods for symmetric indefinite linear systems and eigenvalue problems and apply them to the Anderson model. Since the Anderson model is a large-scale sparse eigenvalue problem in three spatial dimensions, the eigenvalue solvers we deal with are designed to compute only a few interior eigenvalues and eigenvectors, thus avoiding a complete factorization. In particular we will use two modern eigenvalue solvers, which we will briefly introduce in section 5. The first one is Arpack [44], which is a Lanczos-type method using implicit restarts (cf. section 5.1). We use this algorithm together with a shift-and-invert technique; i.e., eigenvalues and eigenvectors of $(A - \sigma I)^{-1}$ are computed instead of those of A. Arpack is used in conjunction with a direct factorization method and a multilevel incomplete factorization method for the shift-and-invert technique.

First, we use the shift-and-invert technique with the novel symmetric indefinite sparse direct solver that is part of Pardiso [60, 59], and we report extensive numerical results on the performance of this method. Section 6 will give a short overview of the main concepts that form the Pardiso solver. Second, we use Arpack in combination with the multilevel incomplete *LU* factorization package Ilupack [11]. Here we present a new *indefinite* version of this preconditioner that is devoted to symmetric indefinite problems and combines two basic ideas, namely (i) symmetric maximum weighted matchings [24, 26] and (ii) inverse-based decomposition techniques [10]. These will be described in sections 6.2 and 8.

As a second eigenvalue solver we use the symmetric version of the Jacobi–Davidson method, in particular the implementation Jdbsym [34]. This Newton-type method (see section 5.2) is used together with Ilupack [11]. As we will see in several further numerical experiments, the synergy of both approaches will form an extremely efficient preconditioner for the Anderson model that is memory efficient while at the same time accelerates the eigenvalue computations significantly; i.e., system sizes that resulted in weeks of computing time [29] can now be computed within an hour.

3 The Anderson model of localization

The Anderson model of localization is a paradigmatic model describing the electronic transport properties of disordered quantum systems [43, 56]. It has been used successfully in amorphous materials such as alloys [54], semiconductors, and even DNA [55]. Its hallmark is the prediction of a spatial confinement of the electronic motion upon increasing the disorder—the so-called Anderson *localization* [2]. When the model is used in three spatial dimensions, it exhibits a metal-insulator transition in which the disorder strength w mediates a change of transport properties from metallic behavior at small w via critical behavior at the transition w_c to insulating behavior and strong localization at larger w [43, 56]. Mathematically, the quantum problem corresponds to a Hamilton operator in the form of a real symmetric matrix A, with quantum mechanical energy levels given by the eigenvalues $\{\lambda\}$, and the respective wave functions are simply the eigenvectors of A, i.e., vectors x with real entries. With $N = M \times M \times M$ sites, the quantum mechanical (stationary) Schrödinger equation is equivalent to the eigenvalue equation $Ax = \lambda x$, which in site representation reads as

$$x_{i-1;j;k} + x_{i+1;j;k} + x_{i;j-1;k} + x_{i;j+1;k} + x_{i;j;k-1} + x_{i;j;k+1} + \varepsilon_{i;j;k} x_{i;j;k} = \lambda x_{i;j;k}, \tag{1}$$

with $i, j, k = 1, 2, \ldots, M$, denoting the Cartesian coordinates of a site. The disorder enters the matrix on the diagonal, where the entries $\varepsilon_{i;j;k}$ correspond to a spatially varying disorder potential and are selected randomly according to a suitable distribution [42]. Here, we shall use the standard box distribution $\varepsilon_{i;j;k} \in [-w/2, w/2]$ such that the w parameterizes the aforementioned disorder strength. Clearly, the eigenvalues of A then lie within the interval $[-6 - w/2,\ 6 + w/2]$ due to the Gershgorin circle theorem. In most studies of the disorder-induced metal-insulator transition, w ranges from 1 to 30 [56]. But these values also depend on whether generalizations to random off-diagonal elements [28, 68] (the so-called random-hopping problem), anisotropies [46, 49], or other lattice graphs [38, 65] are being considered.

The intrinsic physics of the model is quite rich. For disorders $w \ll 16.5$, the eigenvectors are extended, i.e., $x_{i;j;k}$ is fluctuating from site to site, but the envelope $|x|$ is approximately a nonzero constant. For large disorders $w > 16.5$, all eigenvectors are localized such that the envelope $|x_n|$ of the nth eigenstate may be approximately written as $\exp -|\vec{r} - \vec{r}_n|/l_n(w)$ with $\vec{r} = (i, j, k)^T$ and $l_n(w)$ denoting the *localization length* of the eigenstate. In Figure 1, we show examples of such states. Note that $|x|^2$ and not x corresponds to a physically measurable quantity and is therefore the observable quantity of interest to physicists. Directly at $w = w_\mathrm{c} \approx 16.5$, the extended states at $\lambda = 0$ vanish and no current can flow. The wave function vector x appears simultaneously extended and localized, as shown in Figure 2.

In order to numerically distinguish these three regimes, namely, localized, critical, and extended behaviors, one needs to (i) go to extremely large system sizes of order 10^6 to 10^8 and (ii) average over many different realizations of the disorder, i.e., compute eigenvalues or eigenvectors for many matrices with different diagonals. In the present paper, we concentrate on the computation of a few eigenvalues and corresponding eigenvectors for the physically most interesting case of critical disorder w_c and in the center of $\sigma(A)$, i.e., at $\lambda = 0$, for large system sizes [3, 12, 48, 69]. Since there is a high density of states for $\sigma(A)$ at $\lambda = 0$ in all cases, we have the further numerical challenge of clearly distinguishing the eigenstates in this high density region.

4 The Lanczos algorithm and the Cullum–Willoughby implementation

Since the mid 1980s, the preferred numerical tool for studying the Anderson matrix and computing a selected set of eigenvectors, e.g., as needed for a multifractal analysis at the transition, was the Cullum–Willoughby implementation (Cwi) [18, 19, 20] of the Lanczos algorithm. Since

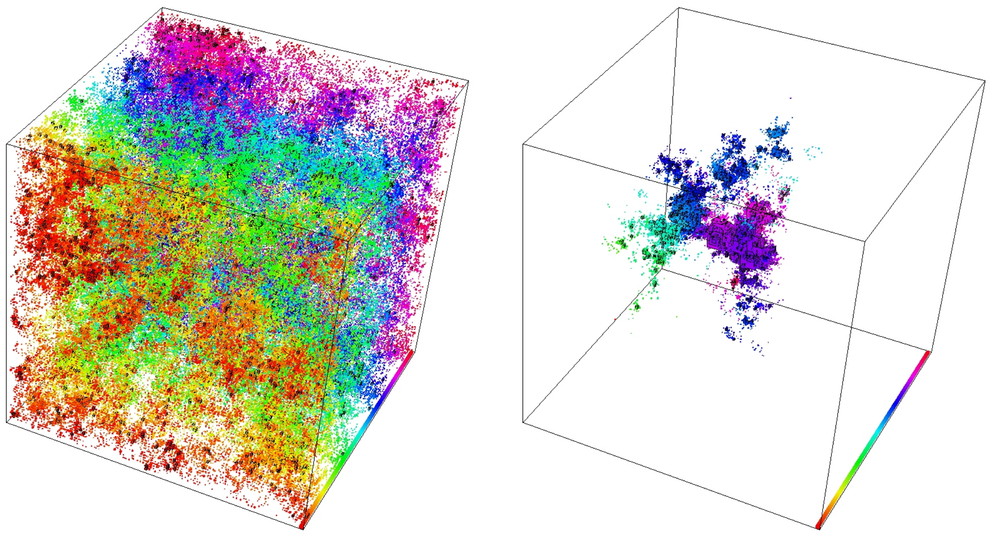


Figure 1: Extended (left) and localized (right) wave function probabilities for the 3D Anderson model with periodic boundary conditions at $\lambda = 0$ with $N = 100^3$ and $w = 12.0$ and 21.0, respectively. Every site with probability $|x_j|^2$ larger than the average $1/N^3$ is shown as a box with volume $|x_j|^2 N$. Boxes with $|x_j|^2 N > \sqrt{1000}$ are plotted with black edges. The color scale distinguishes between different slices of the system along the axis into the page. The eigenstates have been constructed using Arpack in shift-and-invert mode with Pardiso as a direct solver. See section 9 for details.

both Cwi and the algorithm itself are well known, let us here just briefly recall the algorithm's main benefits, mostly to define our notation. The algorithm iteratively generates a sequence of orthogonal vectors v_i, $i = 1, \ldots, K$, such that $V_K^T A V_K = T_K$, with $V = [v_1, v_2, \ldots, v_K]$ and T_K a symmetric tridiagonal $K \times K$ matrix. The recursion $\beta_{i+1} v_{i+1} = A v_i - \alpha_i v_i - \beta_i v_{i-1}$ defines the diagonal and subdiagonal entries of T_K, $\alpha_i = v_i^T A v_i$, and $\beta_{i+1} = v_{i+1} A v_i$, respectively. Its associated (Ritz) eigenvalues and eigenvectors then yield those for the A's.

The Cwi avoids reorthogonalization of the v_i's and hence is very memory efficient. The sparsity of the matrix A can be used to full advantage. However, one needs to construct many Ritz vectors of T_K, which is computationally intensive. Nevertheless, in 1999 Cwi was still significantly faster than more modern iterative schemes [29]. The main reason for this surprising result lies in the indefiniteness of the sparse matrix A, which led to severe difficulties with solvers more accustomed to standard Laplacian-type problems.

5 Modern approaches for solving symmetric indefinite eigenvalueproblems

When dealing with eigenvalues near a given real shift σ, the Lanczos algorithm [53] is usually accelerated when being applied to the shifted inverse $(A - \sigma I)^{-1}$ instead of A directly. This approach relies on the availability of a fast solution method for linear systems of type $(A - \sigma I)x = b$. However, the limited amount of available memory allows only for a small number of solution steps, and sparse direct solvers also need to be memory efficient to turn this approach into a practical method.

The limited number of Lanczos steps has led to modern implicitly restarted methods [44, 67] which ensure that the information about the desired eigenvalues is inherited when being restarted. With an increasing number of preconditioned iterative methods for linear systems [57], Lanczos-

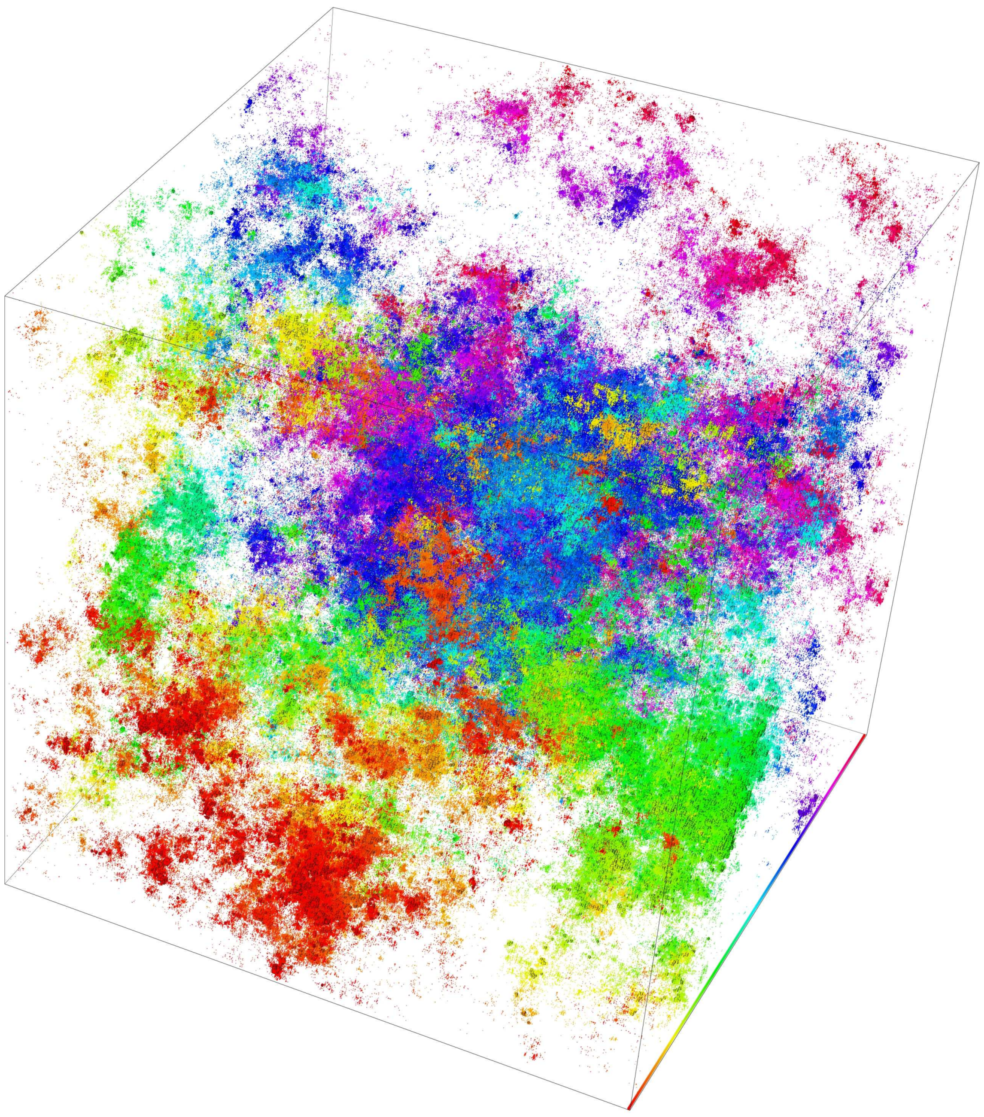


Figure 2: Plot of the electronic eigenstate at the metal-insulator transition with $\lambda = 0$, $w = 16.5$, and $N = 350^3$. The box-and-color scheme is as in Figure 1. Note how the state extends nearly everywhere while at the same time exhibiting certain localized regions of higher $|x_j|^2$ values. The eigenstate has been constructed using Ilupack-based Jacobi–Davidson. See section 9 for details.

type algorithms have become less attractive mainly because in every iteration step the systems of type $(A - \sigma I)x = b$ have to be solved to *full* accuracy in order to avoid false eigenvalues. In contrast to this, Jacobi–Davidson-like methods [66] allow using a crude approximation of the underlying linear system. From the point of view of linear solvers as part of the eigenvalue computation, modern direct and iterative methods need to inherit the symmetric structure $A = A^T$ while maintaining both time and memory efficiency. Symmetric matching algorithms [24, 26, 59] have significantly improved these methods.

5.1 The shift-and-invert mode of the restarted Lanczos method

The Lanczos method for real symmetric matrices A near a shift σ is based on computing successively orthonormal vectors $[v_1, \ldots, v_k, v_{k+1}]$ and a tridiagonal $(k+1) \times k$ matrix

$$\underline{T_k} = \left(\begin{array}{cccc} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \\ \hline & & & \beta_k \end{array}\right) \equiv \left(\begin{array}{c} T_k \\ \hline \beta_k e_k^T \end{array}\right), \tag{2}$$

where e_k is the kth unit vector in $\mathbb{R}^k$, such that

$$(A - \sigma I)^{-1}[v_1, \ldots, v_k] = [v_1, \ldots, v_k, v_{k+1}]\underline{T_k}. \tag{3}$$

Since only a limited number of Lanczos vectors $v_1, \ldots, v_k$ can be stored, and since this Lanczos sequence also consists of redundant information about undesired small eigenvalues, implicitly restarted Lanczos methods have been proposed [67, 44] that use implicitly shifted QR [37], exploiting the small eigenvalues of T_k to remove them from this sequence without ever forming a single matrix vector multiplication with $(A - \sigma I)^{-1}$. The new transformed Lanczos sequence

$$(A - \sigma I)^{-1}[\tilde{v}_1, \ldots, \tilde{v}_l] = [\tilde{v}_1, \ldots, \tilde{v}_l, \tilde{v}_{l+1}]\underline{\tilde{T}_l} \tag{4}$$

with $l \ll k$ then allows one to compute further $k - l$ approximations. This approach is at the heart of the symmetric version of ARPACK [44].

5.2 The symmetric JACOBI–DAVIDSON method

One of the major drawbacks of shift-and-invert Lanczos algorithms is the fact that the multiplication with $(A - \sigma I)^{-1}$ requires solving a linear system to full accuracy. In contrast to this, JACOBI–DAVIDSON-like algorithms [66] are based on a Newton-like approach to solve the eigenvalue problem. Like the Lanczos method, the search space is expanded step by step, solving the correction equation

$$(I - uu^T)(A - \theta I)(I - uu^T)z = -r \quad \text{such that} \quad z = (I - uu^T)z, \tag{5}$$

where (u, θ) is the given approximate eigenpair and $r = Au - \theta u$ is the associated residual. Then the search space based on $V_k = [v_1, \ldots, v_k]$ is expanded by reorthogonalizing z with respect to $[v_1, \ldots, v_k]$, and a new approximate eigenpair is computed from the Ritz approximation $[V_k, z]^T A[V_k, z]$. When computing several right eigenvectors, the projection $I - uu^T$ has to be replaced with $I - [Q, u][Q, u]^T$ using the already computed approximate eigenvectors Q. This ensures that the new approximate eigenpair is orthogonal to those that have already been computed.

The most important part of the JACOBI–DAVIDSON approach is to construct an approximate solution for (5) such that

$$(I - uu^T)K(I - uu^T)c = d \quad \text{with} \quad u^T z = 0 \tag{6}$$

and $K \approx A - \theta I$ that allows for a fast solution of the system $Kx = b$. Here, there is a strong need for robust preconditioning methods that preserve symmetry and efficiently solve sequences of linear systems with K. If K is itself symmetric and indefinite, then the simplified QMR method [31, 32] using the preconditioner $\left(I - \frac{uw^T}{w^T u}\right) K^{-1}$, where $Kw = u$ and the system matrix $\left(I - uu^T\right)(A - \theta I)$, can be used as an iterative method. Note that here the accuracy of the solution of (5) is uncritical until the approximate eigenpair converges [30]. This fact has been exploited in JDBSYM [4, 34]. For an overview on JACOBI–DAVIDSON methods for symmetric matrices see [35].

6 On recent algorithms for solving symmetric indefinite systems of equations

We now report on recent improvements in solving symmetric indefinite systems of linear equations that have significantly changed sparse direct as well as preconditioning methods. One key to the success of these approaches is the use of symmetric matchings, which we review in section 6.2.

6.1 Sparse direct factorization methods

For a long time, dynamic pivoting has been a central tool by which nonsymmetric sparse linear solvers gain stability. Therefore, improvements in speeding up direct factorization methods were limited to the uncertainties that have arisen from using pivoting. Certain techniques, like the column elimination tree [21, 36], have been useful for predicting the sparsity pattern despite pivoting. However, in the symmetric case the situation becomes more complicated since only symmetric reorderings, applied to both columns and rows, are required, and no a priori choice of pivots is given. This makes it almost impossible to predict the elimination tree in a sensible manner, and the use of cache-oriented level-3 BLAS [22, 23] is impossible.

With the introduction of symmetric maximum weighted matchings [24] as an alternative to complete pivoting, it is now possible to treat symmetric indefinite systems similarly to how we treat symmetric positive definite systems. This allows us to predict fill using the elimination tree [33], and thus allows us to set up the data structures that are required to predict dense submatrices (also known as supernodes). This in turn means that one is able to exploit level-3 BLAS applied to the supernodes. Consequently, the classical Bunch–Kaufman pivoting approach [14] needs to be performed only inside the supernodes.

This approach has recently been successfully implemented in the sparse direct solver PARDISO [60, 59]. As a major consequence of this novel approach, the sparse indefinite solver has been improved to become almost as efficient as its symmetric positive analogy. Certainly for the Anderson problem studied here, PARDISO is about two orders of magnitude more efficient than previously used direct solvers [29]. We also note that the idea of symmetric weighted matchings can be carried over to incomplete factorization methods with similar success [40].

6.2 Symmetric weighted matchings as an alternative to complete pivoting techniques

Symmetric weighted matchings [24, 26], which will be explained in detail in section 7.2, can be viewed as a preprocessing step that rescales the original matrix and at the same time improves the block diagonal dominance. By this strategy, all entries are at most one in modulus, and, in addition, the diagonal blocks are either 1×1 scalars a_{ii} such that $|a_{ii}| = 1$ (in exceptional cases we will have $a_{ii} = 0$) or 2×2 blocks

$$\begin{pmatrix} a_{ii} & a_{i,i+1} \\ a_{i+1,i} & a_{i+1,i+1} \end{pmatrix} \text{ such that } |a_{ii}|, |a_{i+1,i+1}| \leqslant 1, \text{ and } |a_{i+1,i}| = |a_{i,i+1}| = 1.$$

Although this strategy does not necessarily ensure that symmetric pivoting, as in [14], is unnecessary, it is nevertheless likely to waive dynamic pivoting during the factorization process. It has been shown in [26] that, based on symmetric weighted matchings, the performance of the sparse symmetric indefinite *multifrontal* direct solver MA57 is improved significantly, although a dynamic pivoting strategy by Duff and Reid [27] was still present. Recent results in [59] have shown that the absence of dynamic pivoting does not harm the method anymore and that, therefore, symmetric weighted matchings can be considered as an alternative to complete pivoting.

7 Symmetric reorderings to improve the results of pivoting on restricted subsets

In this section we will discuss weighted graph matchings as an additional preprocessing step. The motivation for weighted matching approaches is to identify large entries in the coefficient matrix A that, if permuted close to the diagonal, permit the factorization process to identify more acceptable pivots and proceed with fewer pivot perturbations. These methods are based on maximum weighted matchings $\mathcal{M}$ and improve the quality of the factor in a way complementary to the alternative idea of using more complete pivoting techniques. The idea of using a permutation $P_{\mathcal{M}}$ associated with a weighted matching $\mathcal{M}$ as an approximation of the pivoting order for nonsymmetric linear systems was first introduced by Olschowka and Neumaier [51] and extended by Duff and Koster [25] to the sparse case. Permuting the rows $A \leftarrow P_{\mathcal{M}}A$ of the sparse system to ensure a zero-free diagonal or to maximize the product of the absolute values of the diagonal entries are techniques that are now often regularly used for nonsymmetric matrices [7, 47, 60, 64].

7.1 Matching algorithms for nonsymmetric matrices

Let $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ be a general matrix. The nonzero elements of A define a graph with edges $\mathcal{E} = \{(i,j) : a_{ij} \neq 0\}$ of ordered pairs of row and column indices. A subset $\mathcal{M} \subset \mathcal{E}$ is called a matching, or a transversal, if every row index i and every column index j appears at most once in $\mathcal{M}$. A matching $\mathcal{M}$ is called perfect if its cardinality is n. For a nonsingular matrix, at least one perfect matching exists and can be found with well known algorithms. With a perfect matching $\mathcal{M}$, it is possible to define a permutation matrix $P_{\mathcal{M}} = (p_{ij})$ with

$$p_{ij} = \begin{cases} 1 & (j,i) \in \mathcal{M}, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

As a consequence, the permutation matrix $P_{\mathcal{M}}A$ has nonzero elements on its diagonal. This method takes only the nonzero structure of the matrix into account. There are other approaches which maximize the diagonal values in some sense. One possibility is to look for a matrix $P_{\mathcal{M}}$ such that the product of the diagonal values of $P_{\mathcal{M}}A$ is maximal. In other words, a permutation σ has to be found, which maximizes

$$\prod_{i=1}^{n} |a_{\sigma(i)i}|. \tag{8}$$

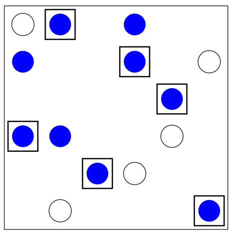
This maximization problem is solved indirectly. It can be reformulated by defining a matrix $C = (c_{ij})$ with

$$c_{ij} = \begin{cases} \log a_i - \log |a_{ij}| & a_{ij} \neq 0, \\ \infty & \text{otherwise,} \end{cases} \tag{9}$$

where $a_i = \max_j |a_{ij}|$, i.e., the maximum element in row i of matrix A. A permutation σ, which minimizes $\sum_{i=1}^{n} c_{\sigma(i)i}$, also maximizes the product (8).

The minimization problem is known as the linear sum assignment problem or the bipartite weighted matching problem in combinatorial optimization. The problem is solved by a sparse variant of the Kuhn–Munkres algorithm. The complexity is $O(n^3)$ for full $n \times n$ matrices and $O(n\tau \log n)$ for sparse matrices with τ entries. For matrices whose associated graph fulfills special requirements, this bound can be reduced further to $O\left(n^{\alpha}(\tau + n \log n)\right)$ with $\alpha < 1$. All graphs arising from finite-difference or finite-element discretizations meet these conditions [39]. As before, we finally get a perfect matching $\mathcal{M}$ that in turn defines a nonsymmetric permutation $P_{\mathcal{M}}$.

The effect of nonsymmetric row permutations using a permutation associated with a matching $\mathcal{M}$ is shown in Figure 3. It is clearly visible that the matrix $P_{\mathcal{M}}A$ is now nonsymmetric, but has the largest nonzeros on the diagonal.

$A =$ 

$$P_{\mathcal{M}}^T = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

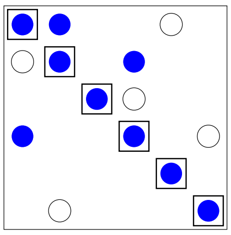
$P_{\mathcal{M}}A =$ 

Figure 3: Illustration of the row permutation. A small numerical value is indicated by $\circ$ and a large numerical value by $\bullet$. The matched entries $\mathcal{M}$ are marked with squares, and $P_{\mathcal{M}} = (e_4; e_1; e_5; e_2; e_3; e_6)$.

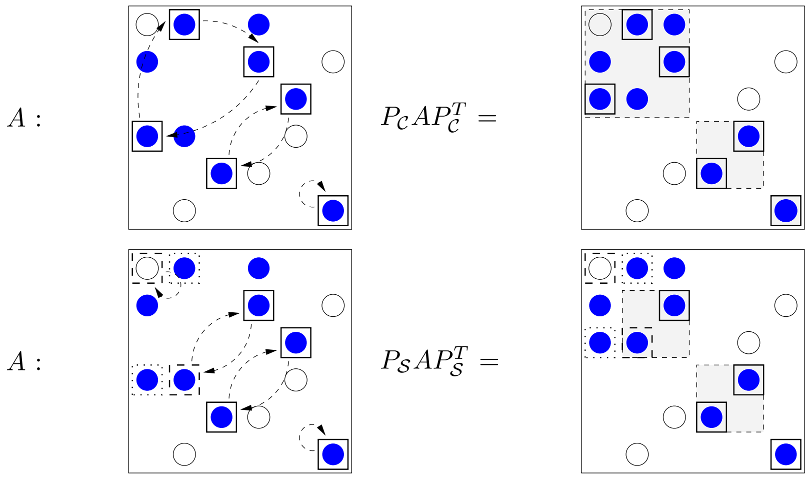

$A:$
$P_{\mathcal{C}}AP_{\mathcal{C}}^T =$
$A:$
$P_{\mathcal{S}}AP_{\mathcal{S}}^T =$

Figure 4: Illustration of a cycle permutation with $P_C = (e_1; e_2; e_4)(e_3; e_5)(e_6)$ and $P_{\mathcal{S}} = (e_1)(e_2; e_4)(e_3; e_5)(e_6)$. The symmetric matching $P_{\mathcal{S}}$ has two additional elements (indicated by dashed boxes), while one element of the original matching fell out (dotted box). The two 2-cycles are permuted into 2×2 diagonal blocks to serve as initial 2×2 pivots.

7.2 Symmetric 1×1 and 2×2 block weighted matchings

In the case of symmetric indefinite matrices, we are interested in symmetrically permuting PAP^T. The problem is that zero or small diagonal elements of A remain on the diagonal when we use a symmetric permutation PAP^T. Alternatively, instead of permuting a large[1] off-diagonal element a_{ij} nonsymmetrically to the diagonal, we can try to devise a permutation $P_{\mathcal{S}}$ such that $P_{\mathcal{S}}AP_{\mathcal{S}}^T$ permutes this element close to the diagonal. As a result, if we form the corresponding 2×2 block to $\left[\begin{smallmatrix} a_{ii} & a_{ij} \\ a_{ij} & a_{jj} \end{smallmatrix}\right]$, we expect the off-diagonal entry a_{ij} to be large, and thus the 2×2 block would form a suitable 2×2 pivot for the supernode Bunch–Kaufman factorization. An observation on how to build $P_{\mathcal{S}}$ from the information given by a weighted matching $\mathcal{M}$ was presented by Duff and Gilbert [24]. They noticed that the cycle structure of the permutation $P_{\mathcal{M}}$ associated with the nonsymmetric matching $\mathcal{M}$ can be exploited to derive such a permutation $P_{\mathcal{S}}$. For example, the permutation $P_{\mathcal{M}}$ from Figure 3 can be written in cycle representation as $P_{\mathcal{C}} = (e_1; e_2; e_4)(e_3; e_5)(e_6)$. This is shown in the upper graphics in Figure 4. The left graphic displays the cycles (1 2 4), (3 5), and (6). If we modify the original permutation $P_{\mathcal{M}} = (e_4; e_1; e_5; e_2; e_3; e_6)$ into this cycle permutation $P_{\mathcal{C}} = (e_1; e_2; e_4)(e_3; e_5)(e_6)$ and permute A symmetrically with $P_{\mathcal{C}}AP_{\mathcal{C}}^T$, it can be observed that the largest elements are permuted to diagonal blocks. These diagonal blocks are shown by filled boxes in the upper right matrix. Unfortunately, a long cycle would result in a large diagonal block, and the fill-in of the factor for $P_{\mathcal{C}}AP_{\mathcal{C}}^T$ may be prohibitively large. Therefore,

[1]Large in the sense of the weighted matching $\mathcal{M}$.

long cycles corresponding to $P_{\mathcal{M}}$ must be broken down into disjoint 2×2 and 1×1 cycles. These smaller cycles are used to define a symmetric permutation $P_{\mathcal{S}} = (c_1, \ldots, c_m)$, where m is the total number of 2×2 and 1×1 cycles.

The rule for choosing the 2×2 and 1×1 cycles from $P_{\mathcal{C}}$ to build $P_{\mathcal{S}}$ is straightforward. One has to distinguish between cycles of even and odd length. It is always possible to break down even cycles into cycles of length 2. For each even cycle, there are two possible ways to break it down. We use a structural metric [26] to decide which one to take. The same metric is also used for cycles of odd length, but the situation is slightly different. Cycles of length $2l + 1$ can be broken down into l cycles of length 2 and one cycle of length 1. There are $2l + 1$ possible ways to do this. The resulting 2×2 blocks will contain the matched elements of $\mathcal{M}$. However, there is no guarantee that the remaining diagonal element corresponding to the cycle of length 1 will be nonzero. Our implementation will randomly select one element as a 1×1 cycle from an odd cycle of length $2l + 1$.

A selection of $P_{\mathcal{S}}$ from a weighted matching $P_{\mathcal{M}}$ is illustrated in Figure 4. The permutation associated with the weighted matching, which is sorted according to the cycles, consists of $P_{\mathcal{C}} = (e_1; e_2; e_4)(e_3; e_5)(e_6)$. We now split the full cycle of odd length 3 into two cycles (1)(24)—resulting in $P_{\mathcal{S}} = (e_1)(e_2; e_4)(e_3; e_5)(e_6)$. If $P_{\mathcal{S}}$ is symmetrically applied to $A \leftarrow P_{\mathcal{S}} A P_{\mathcal{S}}^T$, we see that the large elements from the nonsymmetric weighted matching $\mathcal{M}$ will be permuted close to the diagonal, and these elements will have more chances to form good initial 1×1 and 2×2 pivots for the subsequent (incomplete) factorization.

Good fill-in reducing orderings P_{Fill} are equally important for symmetric indefinite systems. The following section introduces two strategies for combining these reorderings with the symmetric graph matching permutation $P_{\mathcal{S}}$. This will provide good initial pivots for the factorization as well as a good fill-in reduction permutation.

7.3 Combination of orderings P_{Fill} for fill reduction with orderings $P_{\mathcal{S}}$ based on weighted matchings

In order to construct the factorization efficiently, care has to be taken that not too much fill-in is introduced during the elimination process. We now examine two algorithms for the combination of a permutation $P_{\mathcal{S}}$ based on weighted matchings to improve the numerical quality of the coefficient matrix A with a fill-in reordering P_{Fill} based on a nested dissection from Metis [41]. The first method is based on compressed subgraphs and has also been used by Duff and Pralet in [26] in order to find good scalings and orderings for symmetric indefinite systems.

In order to combine the permutation $P_{\mathcal{S}}$ with a fill-in reducing permutation, we compress the graph of the reordered system $P_{\mathcal{S}} A P_{\mathcal{S}}^T$ and apply the fill-in reducing reordering to the compressed graph. In the compression step, the union of the structure of the two rows and columns corresponding to a 2×2 diagonal block is built and used as the structure of a single, compressed row and column representing the original ones.

If $G_A = (V; E)$ is the undirected graph of A and a cycle consists of two vertices $(s, t) \in V$, then graph compression will be done on the 1×1 and 2×2 cycles, which have been found using a weighted matching $\mathcal{M}$ on the graph. The vertices (s, t) are replaced with a single supervertex $u = \{s, t\} \in V_c$ in the compressed graph $G_c = (V_c, E_c)$. An edge $e_c = (s, t) \in E_c$ between two supervertices $s = \{s_1, s_2\} \in V_c$ and $t = \{t_1, t_2\} \in V_c$ exists if at least one of the following edges exists in E: (s_1, t_1), (s_1, t_2), (s_2, t_1), or (s_2, t_2). The fill-in reducing ordering is found by applying Metis on the compressed graph $G_c = (V_c, E_c)$. Expansion of P_{Fill} to the original numbering yields the final permutation. Hence all 2×2 cycles that correspond to a suitable 2×2 pivot block are reordered consecutively in the factor.

8 Symmetric multilevel preconditioning techniques

We now present a new symmetric indefinite approximate multilevel factorization that is mainly based on three parts which are repeated in a multilevel framework in each subsystem. The components consist of (i) reordering of the system, (ii) approximate factorization using inverse-based pivoting, and (iii) recursive application to the system of postponed updates.

8.1 Reordering the given system

The key ingredient for turning this approach into an efficient multilevel solver consists of the symmetric maximum weight matching presented in section 6.2. After the system is reordered into a representation

$$P_s^T DADP_s = \hat{A}, \tag{10}$$

where $D, P_s \in \mathbb{R}^{n,n}$, D is a diagonal matrix, and P_s is a permutation matrix, $\hat{A}$ is expected to have many diagonal blocks of size 1×1 or 2×2 that are well conditioned. Once the diagonal blocks of size 1×1 and 2×2 are built, the associated block graph of $\hat{A}$ is reordered by a symmetric reordering, e.g., AMD [1] or METIS [41], i.e.,

$$\Pi^T P_s^T DADP_s \Pi = \tilde{A}, \tag{11}$$

where $\Pi \in \mathbb{R}^{n,n}$ refers to the associated symmetric block permutation.

8.2 Inverse-based pivoting

Given $\tilde{A}$ we compute an incomplete factorization $LDL^T = \tilde{A} + E$ of $\tilde{A}$. To do this at step k of the algorithm we have

$$\tilde{A} = \begin{pmatrix} B & F^T \\ F & C \end{pmatrix} = \begin{pmatrix} L_B & 0 \\ L_F & I \end{pmatrix} \begin{pmatrix} D_B & 0 \\ 0 & S_C \end{pmatrix} \begin{pmatrix} L_B^T & L_F^T \\ 0 & I \end{pmatrix}, \tag{12}$$

where $L_B \in \mathbb{R}^{k-1,k-1}$ is lower triangular with unit diagonal and $D_B \in \mathbb{R}^{k-1,k-1}$ is block diagonal with diagonal blocks of sizes 1×1 and 2×2. Also, $S_C = C - L_F D_B L_F^T = (s_{ij})_{i,j}$ denotes the approximate Schur complement. To proceed with the incomplete factorization we perform either a 1×1 update or a 2×2 block update. One possible choice could be to use Bunch's algorithm [13]. This approach has been used in [40]. Here we use a simple criterion based on block diagonal dominance of the leading block column. Depending on the values

$$d_1 = \sum_{j>1} \frac{|s_{j1}|}{|s_{11}|}, \quad d_2 = \sum_{j>2} \left\| (s_{j1}, s_{j2}) \begin{pmatrix} s_{11} & s_{12} \\ s_{12} & s_{22} \end{pmatrix}^{-1} \right\|, \tag{13}$$

we perform a 2×2 update only if $d_2 < d_1$. The two leading columns of S_C can be efficiently computed using linked lists [45], and it is not required to have all entries of S_C available.

When applying the (incomplete) factorization LDL^T to $\tilde{A}$ we may still encounter a situation where at step k either $1/|s_{11}|$ or $\|(s_{ij})_{i,j\leqslant 2}^{-1}\|$ is large or even infinite. Since we are dealing with an incomplete factorization we propose to use inverse-based pivoting [10]. Therefore, we require in every step that

$$\left\| \begin{pmatrix} L_B & 0 \\ L_F & I \end{pmatrix}^{-1} \right\| \leqslant \kappa \tag{14}$$

for a prescribed bound κ. If after the update using a 1×1 pivot (or 2×2 pivot) the norm of the inverse lower triangular factor fails to be less than κ, the update is postponed and the leading rows/columns of L_F are permuted to the end of S_C. Otherwise, depending on whether a 1×1 or a 2×2 pivot has been selected, the entries

$$(s_{j1}/s_{11})_{j>1}, \quad \left((s_{j1},\ s_{j2}) \begin{pmatrix} s_{11} & s_{12} \\ s_{12} & s_{22} \end{pmatrix}^{-1} \right)_{j>2} \tag{15}$$

become the next (block) column of L, and we drop these entries whenever their absolute value is less than ε/κ for some threshold ε. For a detailed description see [10]. The norm of the inverse can be cheaply estimated using a refined strategy of [17] and is part of the software package ILUPACK that is now extended to the symmetric indefinite case [11].

8.3 Recursive application

After the inverse-based ILU we have an approximate factorization

$$Q^T \tilde{A} Q = \begin{pmatrix} L_{11} & 0 \\ L_{21} & I \end{pmatrix} \begin{pmatrix} D_{11} & 0 \\ 0 & S_{22} \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{21}^T \\ 0 & I \end{pmatrix}, \tag{16}$$

and it typically does not pay off to continue the factorization for the remaining matrix S_{22} which consists of the previously postponed updates. Thus S_{22} is now explicitly computed and the strategies for reordering, scaling, and factorization are recursively applied to S_{22}, leading to a multilevel factorization.

Note that in order to save memory, L_{21} is not stored but implicitly approximated by $\tilde{A}_{21}(L_{11}D_{11}L_{11}^T)^{-1}$. In addition we use a technique called *aggressive dropping* that sparsifies the triangular factor L a posteriori. To do this observe that when applying a perturbed triangular factor $\tilde{L}^{-1}$ for preconditioning, instead of L^{-1} we have

$$\tilde{L}^{-1} = (I + E_L)L^{-1}, \text{ where } E_L = \tilde{L}^{-1}(L - \tilde{L}).$$

We can expect that $\tilde{L}^{-1}$ serves as a good approximation to L^{-1} as long as $\|E_L\| \ll 1$. If we obtain $\tilde{L}$ from L by dropping some entry, say l_{ij} from L, then we have to ensure that

$$\|\tilde{L}^{-1}e_i\| \cdot |l_{ij}| \leqslant \tau \ll 1$$

for some moderate constant $\tau < 1$, e.g., $\tau = 0.1$. To do this requires having a good estimate for $\nu_i \approx \|\tilde{L}^{-1}e_i\|$ available for any $i = 1, \ldots, n$. In principle it can be computed [10, 17] using $\tilde{L}^T$ instead of $\tilde{L}$. Finally, knowing how many entries exist in column j, we could drop any l_{ij} such that

$$|l_{ij}| \leqslant \tau/(\nu_i \cdot \#\{l_{kj} :\ l_{kj} \neq 0, k = j+1, \ldots, n\}).$$

8.4 Iterative solution

By construction, the computed incomplete multilevel factorization is symmetric but indefinite. For the iterative solution of linear systems using the multilevel factorization, in principle different Krylov subspace solvers could be used, such as general methods that do not explicitly use symmetry (e.g., GMRES [58]) or methods like SYMMLQ [52] which preserve the symmetry of the original matrix but which are devoted only to symmetric positive definite preconditioners. To fully exploit both symmetry and indefiniteness at the same time, here the simplified QMR method [31, 32] is chosen.

9 Numerical experiments

Here we present numerical experiments that show that the previously outlined advances in symmetric indefinite sparse direct solvers as well as in preconditioning methods significantly accelerate modern eigenvalue solvers and allow us to gain orders of magnitude in speed compared to more conventional methods.

9.1 Computing environments and software

All large-scale numerical experiments for the Anderson model of localization were performed on an SGI Altix 3700/BX2 with 56 Intel Itanium2 1.6 GHz processors and 112 GB of memory. If not explicitly stated, we always used only one processor of the system and all algorithms were implemented in either C or Fortran77. All codes were compiled by the Intel V8.1 compiler suite using *ifort* and *icc* with the `−O3` optimization option and linked with basic linear algebra subprograms optimized for Intel architectures. The computations for $M = 250, 350$ and $w = 16.5$ required 64-bit long integers and `−i8` flag for *ifort.* ¿From comparison with smaller examples we observed an overhead of approximately 30% with respect to memory and computation time. For completeness, let us recall the main software packages used:

- Arpack is a collection of Fortran77 subroutines designed to solve large-scale eigenvalue problems. The eigenvalue solver has been developed at the Department of Computational and Applied Mathematics at Rice University. It is available at http://www.caam.rice.edu/software/ARPACK.
- Jdbsym is a C library implementation of the Jacobi–Davidson method optimized for symmetric eigenvalue problems. It solves eigenproblems of the form $Ax = \lambda x$ and $Ax = \lambda Bx$ with or without preconditioning, where A is symmetric and B is symmetric positive definite. It has been developed at the Computer Science Department of the ETH Zürich. It is available at http://people.web.psi.ch/geus/software.html.
- Pardiso is a fast direct solver package, developed at the Computer Science Department of the University of Basel. It is available at http://www.pardiso-project.org.
- Ilupack is an algebraic multilevel preconditioning software package. This iterative solver has been developed at the Mathematics Department of the Technical University of Berlin. It is available at http://www.math.tu-berlin.de/ilupack.

9.2 Cwi compared to shift-and-invert Lanczos with implicit restarts and Pardiso as direct solver

Let us first briefly compare the classical Cwi with the shift-and-invert Lanczos method using implicit restarts. The latter is part of Arpack [44]. For the solution of the symmetric indefinite system $A - \theta I$ we use the most recent version of sparse direct solver Pardiso [60, 59]. This version is based on symmetric weighted matchings and uses Metis as a symmetric reordering strategy. The numerical results deal with the computation of five eigenvalues of the Anderson matrix A near $\lambda = 0$. Here we state the results for the physically most interesting critical disorder strength $w_{\mathrm{c}} = 16.5$. We have measured the CPU times in seconds and memory requirements in GB to compute five eigenvalues closest to $\lambda = 0$ of an Anderson matrix of size $N = M^3 \times M^3$ up to $M = 100$ with Cwi and Arpack–Pardiso. We observe from this initial numerical experiment that the combination of the shift-and-invert Lanczos with Pardiso is faster when compared to the Cwi by about a factor of 10 for systems with $M > 50$. Despite this success, with increasing problem size the amount of memory consumed by the sparse direct solver becomes significant[2] and numerical results with N larger than 1000000 are skipped. Figure 1 shows two different eigenstates computed with the help of Pardiso.

9.3 Using the Ilupack-based preconditioner

We now switch to the Ilupack-based preconditioner that is also based on symmetric weighted matchings and in addition uses inverse-based pivoting. In particular, for our experiments we use $\kappa = 5$ as a bound for the norm $\|L^{-1}\|$ of the inverse triangular factor and Amd for the symmetric reordering. We also tried to use Metis, but for this particular matrix problem we find that Amd is clearly more memory efficient. Next we compare the shift-and-invert Lanczos (Arpack) with Ilupack and the simplified QMR as the inner iterative solver. Here we use $\varepsilon = 1/\sqrt{N}$ with aggressive dropping, and the QMR method is stopped once the norm of residual satisfies $\|Ax - b\| \leqslant 10^{-10}\|b\|$. In order to illustrate the benefits of using symmetric weighted matchings we also tried Ilupack without matching, but the numerical results are disappointing, as can be seen from the †'s in Table 1. We emphasize that the multilevel approach is crucial; a simple

[2]The current standard memory of 2GB RAM for a desktop computer is exceeded for sizes beyond $M > 64$.

Table 1: CPU times in seconds and memory requirements in GB to compute five eigenvalues closest to $\lambda = 0$ of an Anderson matrix of size $M^3 \times M^3$ with Arpack–Pardiso, Arpack–Ilupack, and Arpack–Ilupack–Symmatch. The symbol — indicates that a memory consumption was larger than 25 GB, and † indicates memory problems with respect to the fill-in.

M	W	Arpack					
		Pardiso		Ilupack		Ilupack–Symmatch	
		Time	Mem.	Time	Mem.	Time	Mem.
70	12.0	1359	3.00	5117	1.09	2140	0.95
100	12.0	20639	14.34	39222	5.62	13583	3.20
130	12.0	—	—	†	†	65722	8.20
70	16.5	1305	3.00	504	0.33	477	0.31
100	16.5	20439	14.34	2349	0.95	2177	0.89
130	16.5	—	—	6320	2.09	6530	1.95
160	16.5	—	—	23663	3.95	13863	3.63
70	21.0	1225	3.00	371	0.22	310	0.22
100	21.0	20239	14.34	1513	0.64	1660	0.65
130	21.0	—	—	3725	1.41	3527	1.44
160	21.0	—	—	15302	2.63	20120	2.68

use of incomplete factorization methods without multilevel preconditioning [40] does not give the desired results. Besides the effect of matchings we also compare how the performance of the methods changes when varying the value w from the critical value $w = w_c = 16.5$ to $w = 12.0$ and $w = 21.0$. We find that these changes do not affect the sparse direct solver at all while the multilevel ILU significantly varies in its performance. Up to now our explanation for this effect is the observation that with increasing w the diagonal dominance of the system also increases and the Ilupack preconditioner gains from higher diagonal dominance. As we can see from Table 1, Ilupack still uses significantly less memory than the direct solver Pardiso for all values of w, and it is the only method we were able to use for larger N due to the memory constraints. Also, the computation time is best.

9.4 Using Jacobi–Davidson

When using preconditioning methods inside shift-and-invert Lanczos we usually have to solve the inner linear system for $A - \theta I$ up to machine precision to make sure that the eigenvalues and eigenvectors are sufficiently correct. In contrast to this the Jacobi–Davidson method allows us to solve the associated correction equation less accurately, and only when convergence takes place is a more accurate solution required. In order to show the significant difference between the iterative parts of Arpack and Jacobi–Davidson we state the number of iteration steps in Table 2. If we were to aim for more eigenpairs, we would expect that eventually the Jdbsym would become less efficient and should again be replaced by Arpack. As stopping criteria for the inner iteration inside the Jacobi–Davidson method we use recent results from [50, 70]. Given the eigenvector residual

$$r_{eig} = Au - u\lambda, \text{ where } \|u\| = 1,\ \lambda = u^T Au,$$

and given an approximate solution z of the correction equation (5) with the associated linear system residual

$$r_{lin} = -r_{eig} - (I - uu^T)(A - \theta I)(I - uu^T)z,$$

one could define a new approximate eigenvector via $u_{eignew} = (u + z)/\|u + z\|$. Following [50] the associated new eigenvector residual r_{eignew} can be bounded by

$$\frac{|\|r_{lin}\| - \beta\|z\||}{1 + \|z\|^2} \leqslant \|r_{eignew}\| \leqslant \frac{\sqrt{(\|r_{lin}\| + \beta\|z\|)^2 + \|r_{lin}\|^2\|z\|^2}}{1 + \|z\|^2},$$

Table 2: Number of inner/outer interaction steps inside ARPACK and JACOBI–DAVIDSON. The symbol — indicates that the computations were not performed anymore for ARPACK.

M	W	ILUPACK–SYMMATCH					
		ARPACK			JACOBI–DAVIDSON		
		Outer	Total	Inner average	Outer	Total	Inner average
70	12.0	42	871	20.7	20	218	10.9
100	12.0	43	1101	25.6	17	228	13.4
130	12.0	42	1056	25.1	25	272	10.9
70	16.5	43	611	14.2	18	167	9.3
100	16.5	43	857	19.9	19	193	10.2
130	16.5	42	1058	25.2	19	271	14.3
160	16.5	42	968	23.1	15	223	14.9
190	16.5	—	—	—	19	297	15.6
220	16.5	—	—	—	22	446	20.3
250	16.5	—	—	—	17	463	27.2
350	**16.5**	—	—	—	**16**	**457**	**28.6**
70	21.0	43	585	13.6	18	167	9.3
100	21.0	42	1004	23.9	18	268	14.9
130	21.0	44	914	20.8	16	243	15.2
160	21.0	25	896	35.8	20	398	19.9
190	21.0	—	—	—	21	527	25.1
220	21.0	—	—	—	17	639	37.6
250	21.0	—	—	—	12	502	41.8

where $\beta = |\lambda - \theta + r_{eig}^T z|$. Numerical experiments in [50] indicate that initially

$$\beta\|z\| \ll \|r_{lin}\| \Rightarrow \|r_{eignew}\| \approx \frac{\|r_{lin}\|}{\sqrt{1+\|z\|^2}},$$

while asymptotically we expect r_{lin} to converge to zero leading to

$$\beta\|z\| \gg \|r_{lin}\| \Rightarrow \|r_{eignew}\| \approx \frac{\beta\|z\|}{1+\|z\|^2}.$$

When z is obtained from the simplified QMR algorithm as in our case, it has been shown in [70] that $\|r_{eignew}\|$ and $r_{eig}^T z$ can be cheaply computed as a by-product of the simplified QMR algorithm. In addition, in practice $\|z\|$ need not be recomputed throughout the iteration, since after a few steps, $\|z\|$ typically does not vary too much anymore [50]. This motivates our stopping criterion, where we stop the inner iteration inside QMR whenever

$$\frac{\|r_{lin}\|}{\sqrt{1+\|z\|^2}} \leqslant \min\left\{\|r_{eignew}\|, \frac{\beta\|z\|}{1+\|z\|^2}, \frac{\tau}{2}\right\}.$$

Here τ is the desired tolerance of the eigenvector residual (10^{-10} in our experiments). Note also that $\|r_{lin}\|$ is not explicitly available in the QMR method. Thus we use the quasi residual as an estimate and check only the true residual on exit to safeguard the process.

In what follows we compare the traditional CWI method with the JACOBI–DAVIDSON code JDBSYM [35] using ILUPACK as a preconditioner. Table 3 shows that switching from ARPACK to JACOBI–DAVIDSON in this case improves the total time by another factor of 6 or greater. For this reason JACOBI–DAVIDSON together with ILUPACK will be used as a default solver in the following. The numerical results in Table 3 show a dramatic improvement in the computation time by using ILUPACK-based JACOBI–DAVIDSON. Although this new method slows down for smaller w due to poorer diagonal dominance, a gain by orders of magnitude can still be observed. For $w = 16.5$ and larger, even more than three orders of magnitude in the computation time can be observed. Hence the new method drastically outperforms the CWI method while the memory requirement

Table 3: CPU times in seconds and memory requirements in GB to compute five eigenvalues closest to $\lambda = 0$ with CWI and JACOBI–DAVIDSON using ILUPACK–SYMMATCH for the shift-and-invert technique. ‡ indicates that the convergence of the method was too slow. For CWI and $M = 100$, not all five eigenvalues converged successfully, so the eigenvector reconstruction finished more quickly, leading to variances in the CPU times (*). Computations for $M = 250, 350$, and $w = 16.5$ were computed with 64 bit long integers (^{64}).

M	W	CWI		JACOBI–DAVIDSON ILUPACK–SYMMATCH	
		Time	Mem.	Time	Mem.
70	12.0	20228	0.11	1138	0.9
100	12.0	148843	0.32	7238	3.1
130	12.0	‡	‡	52774	9.0
70	16.5	15100	0.11	161	0.3
100	16.5	255842*	0.32	661	1.0
130	16.5	‡	‡	2000	2.4
160	16.5	‡	‡	3961	4.8
190	16.5	‡	‡	10955	8.1
220	16.5	‡	‡	25669	12.3
250	16.5	‡	‡	57203 ^{64}	26.0 ^{64}
350	**16.5**	‡	‡	**182276** ^{64}	**88.0** ^{64}
70	21.0	14371	0.11	99	0.3
100	21.0	331514*	0.32	484	0.8
130	21.0	‡	‡	1069	1.6
160	21.0	‡	‡	3070	3.2
190	21.0	‡	‡	8564	5.6
220	21.0	‡	‡	17259	8.5
250	21.0	‡	‡	24802	12.6

Table 4: The influence of the inverse bound κ on the amount of memory. For $M = 70$, compare for different thresholds how the fill-in $nnz(LDL^T)/nnz(A)$ varies depending on κ and state the computation time in seconds.

ε	$\kappa = 5$			$\kappa = 10$			$\kappa = 20$		
	Fill	Time LDL^T	Total time	Fill	Time LDL^T	Total time	Fill	Time LDL^T	Total time
0.01	5.4	37	870	8.7	67	500	15.2	160	480
0.005	6.8	54	440	11.0	100	380	19.1	230	500
0.0025	8.6	81	310	13.8	150	360	24.1	340	600
0.001	11.7	130	300	18.0	230	410	32.1	540	780

is still moderate. Figure 2 shows an eigenstate computed within three days with the help of the ILUPACK-based JACOBI–DAVIDSON. The construction of the ILUPACK preconditioner needed 14 hours at a fill-in factor of 18 compared with the fill of the original matrix.

One key to the success of the preconditioner is based on the threshold κ which bounds the growth of L^{-1}. Already, for a small example such as $M = 70$ significant differences can be observed. As we show in Table 4, increasing the bound by a factor of 2 from $\kappa = 5$ up to $\kappa = 10$ and $\kappa = 20$ leads to an enormous increase in fill. Here we measure the fill of the incomplete LDL^T factorization relative to the nonzeros of the original matrix. By varying the drop tolerance ε we also see that the dependence on κ is much more significant than the dependence on ε. Roughly speaking, the ILU decomposition becomes twice as expensive when κ is replaced with 2κ, as does the fill-in. The latter is crucial since memory constraints severely limit the size of the application that can be computed.

Table 5: Difference in performance for our standard problem with periodic boundary conditions, the problem with hard wall conditions, and the inverse problem with random numerical entries in the off-diagonal elements. Memory requirement (in GB) and CPU times (in seconds) to compute at the transition the eigenvectors corresponding to the five eigenvalues closest to $\lambda = 0$ with shift-and-invert Jacobi–Davidson and the Ilupack–Symmatch solver using symmetric weighted matchings.

N	Periodic		Hard wall		Inverse	
	Time	Memory	Time	Memory	Time	Memory
70	161	0.3	169	0.3	251	0.2
100	661	1.0	704	0.9	1055	0.8
130	2000	2.4	1566	2.4	3203	1.8
160	3961	4.8	4078	4.6	11614	3.3
190	10955	8.1	10922	7.8	29455	6.5

9.5 Hard wall boundaries and randomness in the off-diagonal matrix elements

In Table 5 we show how Jdbsym and Ilupack–Symmatch perform when, instead of periodic boundary conditions, we use hard wall boundaries, i.e., $x_{0;j;k} = x_{i;0;k} = x_{i;j;0} = x_{M+1;j;k} = x_{i;M+1;k} = x_{i;j;M+1} = 0$ for all i, j, k. This is sometimes of interest in the Anderson problem, and, generally, it is expected that for large M, the difference in eigenvalues and eigenvectors becomes small when compared to the standard periodic boundaries. In addition, we also show results for the so-called off-diagonal Anderson problem [16]. Here, we shift the diagonal to a constant $\sigma = 1.28$ and incorporate the randomness by setting the off-diagonal elements of A to be uniformly distributed in $[-1/2, 1/2]$. The graph of the matrix A remains the same. These values correspond—similarly to $w_c = 16.5$ used before for purely diagonal randomness—to the physically most interesting localization transition in this model [16]. We note that using hard wall boundary conditions instead of periodic boundary conditions leads to slightly less fill but increases the number of iteration steps, as can be seen in Table 5. This conclusions carries over to the off-diagonal Anderson problem, where the memory consumption is less but the iterative part takes even longer. In principle our results could be improved if we were to switch to a smaller threshold ε than the uniformly applied $\varepsilon = 1/\sqrt{N}$ here.

10 Conclusion

We have shown that modern approaches to preconditioning based on symmetric matchings and multilevel preconditioning methods lead to an astonishing increase in performance and available system sizes for the Anderson model of localization. This approach is not only several orders of magnitudes faster than the traditional Cwi approach, but it also consumes only a moderate amount of memory, thus allowing us to study the Anderson eigenproblem for significantly larger scales than ever before.

Let us briefly recall the main ingredients necessary for this progress. At the heart of the new approach lies the use of symmetric matchings [40] in the preconditioning stage of the inverse-based incomplete factorization preconditioning iterative method [11]. Furthermore, the preconditioning itself is of a multilevel type, complementary to the often used full-pivoting strategies. Next, the inverse-based approach is also of paramount importance to keep the fill-in at a manageable level (see Table 4). Finally, we emphasize that these results, of course, reflect our selected problem class: to compute a few of the interior eigenvalues and associated eigenvectors for a highly indefinite symmetric matrix defined by the Anderson model of localization.

The performance increase by several orders of magnitude (see Table 3) is solely due to our use of new and improved algorithms. Combined with advances in the performance-to-cost ratio of computing hardware during the past six years, current preconditions methods make it possible to solve those problems quickly and easily, which have been considered by far too large until recently [15]. Even for $N \times N$ matrices as large as $N = 64 \cdot 10^6$, it is now possible to compute within a few days the interior eigenstates of the Anderson problem.

Improved software packages and applications in other research areas Two efficient multi-level iterative solver packages, Ilupack [10] and Pardiso/Ml [61] for symmetric indefinite systems based on the described algorithmic strategy are now available. An integrated version of the Jacobi–Davidson approach together with Ilupack has been implemented in the package Jadamilu and can be found in Refs. [8, 9].

The success of the method indicates that it can also be successfully applied to other large-scale problems. The algebraic multilevel solver Pardiso/Ml has been used e.g. in Ref. [63] for the solution of large-scale nonconvex nonlinear programming problems that arise in biomedical cancer simulations. These biomedical nonconvex 3D PDE-constrained optimization problems can have over hundred of millions nonzeros in the Jacobian and the total NLP problem can be solved efficiently with Pardiso/Ml.

Acknowledgments

We gratefully acknowledge discussions with and implementation help from A. Croy and C. Sohrmann in the initial stages of the project. We thankfully acknowledge support and encouragement by V. Mehrmann and M. Schreiber who also got us interested in the problem.

References

[1] P. R. Amestoy, T. A. Davis, and I. S. Duff, *An approximate minimum degree ordering algorithm*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 886–905.

[2] P. W. Anderson, *Absence of diffusion in certain random lattices*, Phys. Rev., 109 (1958), pp. 1492–1505.

[3] H. Aoki, *Fractal dimensionality of wave functions at the mobility edge: Quantum fractal in the Landau levels*, Phys. Rev. B, 33 (1986), pp. 7310–7313.

[4] P. Arbenz and R. Geus, *Multilevel preconditioned iterative eigensolvers for Maxwell eigenvalue problems*, Appl. Numer. Math., 54 (2005), pp. 107–121.

[5] M. Benzi, *Preconditioning techniques for large linear systems: A survey*, J. Comput. Phys., 182 (2002), pp. 418–477.

[6] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[7] M. Benzi, J. C. Haws, and M. Tůma, *Preconditioning highly indefinite and nonsymmetric matrices*, SIAM J. Sci. Comput., 22 (2000), pp. 1333–1353.

[8] M. Bollhöfer and Y. Notay, *JADAMILU: a software code for computing selected eigenvalues of large sparse symmetric matrices*, Comput. Phys. Commun., 177 (12), pp. 951-964.

[9] M. Bollhöfer and Y. Notay, *JADAMILU, JAcobi-DAvidson method with Multilevel ILU preconditioning*, ULB, Brussels, Belgium, October 2006. Available online at http://homepages.ulb.ac.be/~jadamilu/

[10] M. Bollhöfer and Y. Saad, *Multilevel preconditioners constructed from inverse-based ILUs*, SIAM J. Sci. Comput., 27 (2006), pp. 1627–1650.

[11] M. Bollhöfer, Y. Saad and O. Schenk, *ILUPACK Volume* 2.1*—Preconditioning Software Package*, http://www.math.tu-berlin.de/ilupack/ (2006).

[12] T. Brandes, B. Huckestein, and L. Schweitzer, *Critical dynamics and multifractal exponents at the Anderson transition in* 3*d disordered systems*, Ann. Phys. (Leipzig), 5 (1996), pp. 633–651.

[13] J. R. Bunch, *Partial pivoting strategies for symmetric matrices*, SIAM J. Numer. Anal., 11 (1974), pp. 521–528.

[14] J. R. Bunch and L. Kaufman, *Some stable methods for calculating inertia and solving symmetric linear systems*, Math. Comp., 31 (1977), pp. 163–179.

[15] P. Cain, F. Milde, R. A. Römer, and M. Schreiber, *Use of cluster computing for the Anderson model of localization*, Comput. Phys. Comm., 147 (2002), pp. 246–250.

[16] P. Cain, R. A. Römer, and M. Schreiber, *Phase diagram of the three-dimensional Anderson model of localization with random hopping*, Ann. Phys. (Leipzig), 8 (1999), pp. SI33–SI38.

[17] A. K. Cline, C. B. Moler, G. W. Stewart, and J. H. Wilkinson, *An estimate for the condition number of a matrix*, SIAM J. Numer. Anal., 16 (1979), pp. 368–375.

[18] J. Cullum and R. A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Volume* 1*: Theory*, Birkhäuser Boston, Boston, MA, 1985.

[19] J. Cullum and R. A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Volume* 2*: Programs*, Birkhäuser Boston, Boston, MA, 1985. Available online at http://www.netlib.org/lanczos/.

[20] P. Dayal, M. Troyer, and R. Villiger, *The Iterative Eigensolver Template Library*, ETH Zürich, Zürich, Switzerland, 2004. Available online at http://www.comp-phys.org:16080/ software/ietl/.

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

[22] D. Dodson and J. G. Lewis, *Issues relating to extension of the basic linear algebra subprograms*, ACM SIGNUM Newslett., 20 (1985), pp. 19–22.

[23] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, *A proposal for an extended set of Fortran basic linear algebra subprograms*, ACM SIGNUM Newslett., 20 (1985), pp. 2–18.

[24] I. S. Duff and J. R. Gilbert, *Symmetric weighted matching for indefinite systems*, talk presented at the Householder Symposium XV, 2002.

[25] I. S. Duff and J. Koster, *The design and use of algorithms for permuting large entries to the diagonal of sparse matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 889–901.

[26] I. S. Duff and S. Pralet, *Strategies for Scaling and Pivoting for Sparse Symmetric Indefinite Problems*, Technical Report TR/PA/04/59, CERFACS, Toulouse, France, 2004.

[27] I. S. Duff and J. K. Reid, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Trans. Math. Software, 9 (1983), pp. 302–325.

[28] A. Eilmes, R. A. Römer, and M. Schreiber, *The two-dimensional Anderson model of localization with random hopping*, Eur. Phys. J. B, 1 (1998), pp. 29–38.

[29] U. Elsner, V. Mehrmann, F. Milde, R. A. Römer, and M. Schreiber, *The Anderson model of localization: A challenge for modern eigenvalue methods*, SIAM J. Sci. Comput., 20 (1999), pp. 2089–2102.

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

[31] R. Freund and F. Jarre, *A QMR-based interior-point algorithm for solving linear programs*, Math. Programming Ser. B, 76 (1997), pp. 183–210.

[32] R. Freund and N. Nachtigal, *Software for simplified Lanczos and QMR algorithms*, Appl. Numer. Math., 19 (1995), pp. 319–341.

[33] A. George and E. Ng, *An implementation of Gaussian elimination with partial pivoting for sparse systems*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 390–409.

[34] R. Geus, *JDBSYM Version* 0.14, http://www.inf.ethz.ch/personal/geus/software.html (2006).

[35] R. Geus, *The Jacobi–Davidson Algorithm for Solving Large Sparse Symmetric Eigenvalue Problems with Application to the Design of Accelerator Cavities*, Ph.D. thesis, Computer Science Department, ETH Zürich, Zürich, Switzerland, 2002. Available online at http://www.inf.ethz.ch/personal/geus/publications/diss-online.pdf.

[36] J. R. Gilbert and E. Ng, *Predicting structure in nonsymmetric sparse matrix factorizations*, in Graph Theory and Sparse Matrix Computation, J. A. George, J. R. Gilbert, and J. W. H. Liu, eds., Springer, New York, 1993, pp. 107–139.

[37] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[38] U. Grimm, R. A. Römer, and G. Schliecker, *Electronic states in topologically disordered systems*, Ann. Phys. (Leipzig), 7 (1998), pp. 389–393.

[39] A. Gupta and L. Ying, *A Fast Maximum-Weight-Bipartite-Matching Algorithm for Reducing Pivoting in Sparse Gaussian Elimination*, Tech. Report RC 21576 (97320), IBM T. J. Watson Research Center, Yorktown Heights, NY, 1999.

[40] M. Hagemann and O. Schenk, *Weighted matchings for preconditioning symmetric indefinite linear systems*, SIAM J. Sci. Comput., 28 (2006), pp. 403–420.

[41] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392.

[42] B. Kramer, A. Broderix, A. MacKinnon, and M. Schreiber, *The Anderson transition: New numerical results for the critical exponents*, Phys. A, 167 (1990), pp. 163–174.

[43] B. Kramer and A. MacKinnon, *Localization: Theory and experiment*, Rep. Progr. Phys., 56 (1993), pp. 1469–1564.

[44] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998. Available online at http://www.caam.rice.edu/software/ARPACK/.

[45] N. Li, Y. Saad, and E. Chow, *Crout versions of ILU for general sparse matrices*, SIAM J. Sci. Comput., 25 (2003), pp. 716–728.

[46] Q. Li, S. Katsoprinakis, E. N. Economou, and C. M. Soukoulis, *Scaling properties in highly anisotropic systems*, Phys. Rev. B, 56 (1997), pp. R4297–R4300,

[47] X. S. Li and J. W. Demmel, *SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems*, ACM Trans. Math. Software, 29 (2003), pp. 110–140.

[48] F. Milde, R. A. Römer, and M. Schreiber, *Multifractal analysis of the metal-insulator transition in anisotropic systems*, Phys. Rev. B, 55 (1997), pp. 9463–9469.

[49] F. Milde, R. A. Römer, and M. Schreiber, *Energy-level statistics at the metal-insulator transition in anisotropic systems*, Phys. Rev. B, 61 (2000), pp. 6028–6035.

[50] Y. Notay, *Inner Iterations in Eigenvalue Solvers*, Technical Report GANMN 05-01, Université Libre de Bruxelles, Brussels, Belgium, 2005.

[51] M. Olschowka and A. Neumaier, *A new pivoting strategy for Gaussian elimination*, Linear Algebra Appl., 240 (1996), pp. 131–151.

[52] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[53] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[54] I. Plyushchay, R. A. Römer, and M. Schreiber, *Three-dimensional Anderson model of localization with binary random potential*, Phys. Rev. B, 68 (2003), 064201.

[55] D. Porath, G. Cuniberti, and R. Di Felice, *Charge transport in DNA-based devices*, in Long-Range Charge Transfer in DNA II, Topics in Current Chemistry 237, G. B. Schuster, ed., Springer, New York, 2004, p. 183.

[56] R. A. Römer and M. Schreiber, *Numerical investigations of scaling at the Anderson transition*, in The Anderson Transition and Its Ramifications—Localisation, Quantum Interference, and Interactions, T. Brandes and S. Kettemann, eds., Springer, Berlin, 2003, pp. 3–19.

[57] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

[58] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[59] O. Schenk and K. Gärtner, *On fast factorization pivoting methods for symmetric indefinite systems*, Elec. Trans. Numer. Anal., 23 (2006), pp. 158–179.

[60] O. Schenk and K. Gärtner, *Solving unsymmetric sparse systems of linear equations with PARDISO*, J. Future Generation Computer Systems, 20 (2004), pp. 475–487.

[61] O. Schenk and K. Gärtner, *PARDISO and PARDISO/ML — parallel sparse direct and algebraic multi-level solver packages for sparse linear matrices,* July 2007, http://www.pardiso-project.org.

[62] O. Schenk, A. Wächter, and M. Hagemann, *Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization*, J. of Comp. Optim. and Appl., 36 (2007), pp. 475–487.

[63] O. Schenk, A. Wächter, and M. Weiser, *Inertia revealing preconditioning for large-scale nonconvex constrained optimization*, Technical Report CS-2007-12 (2007), Computer Science Department, University of Basel, Switzerland, submitted.

[64] O. Schenk, S. Röllin, and A. Gupta, *The effects of unsymmetric matrix permutations and scalings in semiconductor device and circuit simulation*, IEEE Trans. Computer-Aided Design Integrated Circuits Systems, 23 (2004), pp. 400–411.

[65] M. Schreiber and M. Ottomeier, *Localization of electronic states in 2D disordered systems*, J. Phys.: Condens. Matter, 4 (1992), pp. 1959–1971.

[66] G. L. G. Sleijpen and H. A. Van der Vorst, *A Jacobi–Davidson iteration for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.

[67] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

[68] C. M. Soukoulis and E. N. Economou, *Off-diagonal disorder in one-dimensional systems*, Phys. Rev. B, 24 (1981), pp. 5698–5702.

[69] C. M. Soukoulis and E. N. Economou, *Fractal character of eigenstates in disordered systems*, Phys. Rev. Lett., 52 (1984), pp. 565–568.

[70] A. Stathopoulos, *Nearly Optimal Preconditioned Methods for Hermitian Eigenproblems under Limited Memory. Part* I: *Seeking One Eigenvalue*, Technical Report WM-CS-2005-03, College of William & Mary, Williamsburg, VA, 2005.