Exact Diagonalization of Large Sparse Matrices: A Challenge for Modern Supercomputers

G. Wellein[*], *G. Hager*[†]
Regionales Rechenzentrum Erlangen, Universität Erlangen, 91058 Erlangen, Germany

A. Basermann[‡]
C&C Research Laboratories, NEC Europe Ltd, 53757 Sankt Augustin, Germany

H. Fehske[§]
Physikalisches Institut, Universität Bayreuth, 95440 Bayreuth, Germany

ABSTRACT: Exact diagonalization of very large sparse matrices is a numerical problem common to various fields in science and engineering. We present an advanced eigenvalue alorithm - the so-called Jacobi-Davidson algorithm - in combination with an efficient parallel matrix-vector multiplication based on the *Jagged Diagonals Storage* (JDS) format. Our JDS implementation allows calculation of several specified eigenvalues with high accuracy both on vector and on RISC processor based supercomputers. Using a 256 processor CRAY T3E-1200 we were able to discuss the fundamental question of the metal-insulator transition in one-dimensional half-filled electron-phonon systems and the properties of the 3/4 filled Peierls-Hubbard Hamiltonian in relation to recent resonant Raman experiments on MX chain [-PtCl-] complexes. In this context we present an extensive performance study on modern supercomputers such as CRAY T3E, SGI Origin3800, NEC SX5e and Hitachi SR8000.

1 Introduction

Many problems in theoretical physics are related to eigenvalue problems involving large sparse matrices. To investigate the low energy physics in the framework of the corresponding microscopic models, iterative subspace methods like the Lanczos algorithm [1] or the Davidson algorithm [2] are commonly used to calculate the ground state and some excited eigenstates. However these methods show a poor convergence and stability if the eigenvalues to be computed are not well separated or even degenerate. In that case, more advanced methods like the Jacobi-Davidson algorithm with preconditioning techniques [3, 4] have to be used, providing both high resolution and rapid convergence.

In general, the computational requirements of these eigenvalue algorithms are determined by a matrix-vector multiplication (MVM) involving the large sparse Hamiltonian matrix. Thus the efficient use of modern supercomputers strongly depends on a parallel, fast and memory saving implementation of the matrix-vector multiplication. Considering the sparsity of the matrix, storage techniques like the *Compressed Row Storage* (CRS) format [5] or the *Jagged Diagonals Storage* (JDS) format [5] are used to store the nonzero elements of the matrix. Using these storage formats, both the memory and cpu-time requirements of the eigenvalue algorithms scale linearly with the matrix dimension.

A typical physical problem that requires the application of high-resolution techniques - such as Jacobi-Davidson - is the calculation of the eigenvalue spectrum for strongly interacting electron/spin-phonon systems. As an example, in section 6 we investigate the 1D Peierls-Hubbard Hamiltonian which serves as a generic model for the intensely studied MX-chain compounds. Most notably, in the strong-coupling regime we find clear signatures of local lattice distortions accompanied by intrinsic multi-phonon localization. Moreover, we are able to reproduce the observed red shift of the overtone resonance Raman spectrum.

The paper is organized as follows: In section 2 the Jacobi-Davidson algorithm is briefly introduced. Section 3 outlines the basic ideas of the MVM implementation using the JDS format. A brief survey of the benchmark platforms as well as the benchmark procedure is given in section 4. The scalability and performance of our MVM implementation is discussed in section 5. Section 6 presents selected results of our MX-chain investigations.

[*] gerhard.wellein@rrze.uni-erlangen.de
[†] georg.hager@rrze.uni-erlangen.de
[‡] baserman@ccrl-nece.de
[§] holger.fehske@uni-bayreuth.de

2 Large sparse eigenproblems

To solve large sparse Hermitian eigenvalue problems numerically, variants of a method proposed by Davidson [6] are frequently applied. These solvers use a succession of subspaces where the update of the subspace exploits approximate inverses of the problem matrix, A. For A, $A = A^H$ or $\bar{A} = A^T$ holds where $\bar{A}$ denotes A with complex conjugate elements and $A^H = \overline{A^T}$ (transposed and complex conjugate).

The basic idea is: Let $\mathbf{V}^k$ be a subspace of $\mathbb{R}^n$ with an orthonormal basis $w_1^k, \ldots, w_m^k$ and W the matrix with columns w_j^k, $S := W^H A W$, $\hat{\lambda}_j^k$ the eigenvalues of S, and T a matrix with the eigenvectors of S as columns. The columns x_j^k of WT are approximations to eigenvectors of A with Ritz values $\hat{\lambda}_j^k = (x_j^k)^H A x_j^k$ that approximate eigenvalues of A. Let us assume that $\hat{\lambda}_{j_s}^k, \ldots, \hat{\lambda}_{j_{s+l-1}}^k \in [\lambda_{\text{lower}}, \lambda_{\text{upper}}]$. For $j \in j_s, \ldots, j_{s+l-1}$ define

$$q_j^k = (A - \hat{\lambda}_j^k I)\, x_j^k\,, \qquad r_j^k = (\hat{A} - \hat{\lambda}_j^k I)^{-1} q_j^k\,, \quad (1)$$

and $\mathbf{V}^{k+1} = \text{span}(\mathbf{V}^k \cup r_{j_s}^k \cup \ldots \cup r_{j_{s+l-1}}^k)$ where $\hat{A}$ is an easy to invert approximation to A ($\hat{A} = \text{diag}(A)$ in [6]). Then $\mathbf{V}^{k+1}$ is an $(m+l)$-dimensional subspace of $\mathbb{R}^n$, and the repetition of the procedure above gives, in general, improved approximations to eigenvalues and -vectors. Restarting may increase efficiency.

For good convergence, $\mathbf{V}^k$ has to contain crude approximations to all eigenvectors of A with eigenvalues smaller than λ_{lower} [6]. The approximate inverse of A must not be too accurate, otherwise the method stagnates. The reason for this was investigated in [3, 7] and leads to the Jacobi-Davidson method with an improved definition of r_j^k:

$$[(I - x_j^k\,(x_j^k)^H)\,(\hat{A} - \hat{\lambda}_j^k I)\,(I - x_j^k\,(x_j^k)^H)]\, r_j^k = q_j^k\,. \quad (2)$$

The projection $(I - x_j^k\,(x_j^k)^H)$ in (2) is not easy to incorporate into the matrix, but there is no need to do so, and solving (2) is only slightly more expensive than solving (1).

For the choice $\hat{A} = I$, the Jacobi-Davidson algorithm expands the subspace in the same way as the (Generalized) Davidson or the Arnoldi method; for $\hat{A} = A$, it corresponds to a Shift-and-Invert method with optimal shift. In the latter case, the Jacobi-Davidson algorithm converges quadratically. Stagnation which can occur for both Davidson and Shift-and-Invert is avoided by the projection [3, 7].

The character of the Jacobi-Davidson method is determined by the approximation $\hat{A}$ to A. Suited algorithms to obtain an approximate solution of the preconditioning system (2) are discussed in [8].

3 Implementation of matrix-vector multiplication

The performance of the Jacobi-Davidson method is mainly determined by a sparse MVM. The extreme sparsity of the matrices used in this work ($\sim$ 10-20 nonzero elements per row, matrix dimension up to 10^8) calls for a solution where only the nonzero elements are stored. In a previous work [10] it was demonstrated that the CRS format which is widely used has serious drawbacks on vector processors due to short vector lengths and provides only a minor performance benefit on RISC processors. To achieve high performance on a wide range of supercomputer architectures, the JDS scheme was implemented.

3.1 The JDS format

The JDS format is a very general storage scheme for sparse matrices making no assumptions about the sparsity structure. To illustrate the principles of the scheme, we introduce an 8×8 matrix A with nonzero elements a_{ij}.

In a compression step, the zero elements are removed, nonzero elements are shifted to the left and the rows are made equal in length by padding them with trailing zeros. To prevent storing theses trailing zeros, the rows are then rearranged according to the number of nonzeros per row:

$$A = \begin{bmatrix} a_{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & a_{36} & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & a_{46} & a_{47} & a_{48} \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} & a_{57} & 0 \\ 0 & 0 & 0 & a_{64} & a_{65} & a_{66} & a_{67} & 0 \\ 0 & 0 & 0 & a_{74} & a_{75} & a_{76} & a_{77} & 0 \\ 0 & 0 & 0 & a_{84} & 0 & 0 & 0 & a_{88} \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} a_{43} & a_{44} & a_{45} & a_{46} & a_{47} & a_{48} \\ a_{74} & a_{75} & a_{76} & a_{77} & 0 & 0 \\ a_{32} & a_{33} & a_{34} & a_{36} & 0 & 0 \\ a_{54} & a_{55} & a_{56} & a_{57} & 0 & 0 \\ a_{64} & a_{65} & a_{66} & a_{67} & 0 & 0 \\ a_{22} & a_{23} & 0 & 0 & 0 & 0 \\ a_{84} & a_{88} & 0 & 0 & 0 & 0 \\ a_{11} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The columns of the compressed and permuted matrix are called *Jagged Diagonals*. Obviously, the number of these *Jagged Diagonals* is equal to the maximum number `max_nz` of nonzeros per row.

3.2 JDS matrix-vector multiplication

The MVM is performed along the *Jagged Diagonals*, providing an inner loop length essentially equal to the matrix dimension (D_{Mat}). To minimize indirect memory accesses, vectors involved in the MVM are permuted once, so that they automatically carry the same permuted order as the matrix rows. The numerical core of the MVM is

```
for j = 1,..,max_nz do
    for i = 1,..,(jd_ptr(j+1) - jd_ptr(j)) do
       y(i) = y(i) + value(jd_ptr(j)+i-1)·
&             x(col_ind(jd_ptr(j)+i-1))
    end for
end for
```

Figure 1: MVM in JDS format: $\mathbf{y} = \mathbf{y} + A\mathbf{x}$. The nonzero elements of A are stored in the linear array `value`, stringing the *Jagged Diagonals* together one by one. `col_ind(:)`, an array of the same length, contains the corresponding column indices. The pointer array `jd_ptr(:)` consists of the array indices belonging to the first elements of each *Jagged Diagonal* in `value(:)`.

given in Fig. 1. The innermost loop requires one store and four load operations (including one indirect load) to perform two floating-point operations (Flop). In other words, the performance of the MVM is clearly determined by the quality of the memory access. Even for present-day vector processors such as NEC SX5e with a peak performance to memory bandwidth ratio of one word per Flop, we do not expect the MVM performance to exceed 20%-25% of the peak performance. The performance on RISC systems is even worse [9]. To reduce the load operations associated with the vector components of `y`, we have splitted the outer `j` loop into several loops over *Jagged Diagonals* with equal length by introducing an outermost loop `k` (cf. Fig. 2), which enables then unrolling of the `j` loop by the compiler. Since there are several groups containing 4-5 *Jagged Diagonals* with the same length in our matrices, this modification gives in general a performance improvement especially for RISC processors.

```
for k = 1,..,dif_jds do
    length=i_length(k)
    for j = j_start(k),..,j_end(k) do
       for i = 1,..,length do
          y(i) = y(i) + value(jd_ptr(j)+i-1)·
&            x(col_ind(jd_ptr(j)+i-1))
       end for
    end for
end for
```

Figure 2: Modified JDS MVM with an outer `k` loop running over `dif_jds` blocks of *Jagged Diagonals* with different loop lengths.

3.3 Parallel implementation

For parallel computers with distributed memory, we use a row-wise distribution of matrix and vector elements among the processes involved in eigenvalue computation (cf. Fig. 3). Matrix elements requiring access to vector data stored on remote processes (*nonlocal* vector elements) cause communication. Since the matrix does not change during eigenvalue computation, a definite communication scheme is constructed once, allowing an efficient way of exchanging data as well as an overlapping of communication and computation in the MVM step.

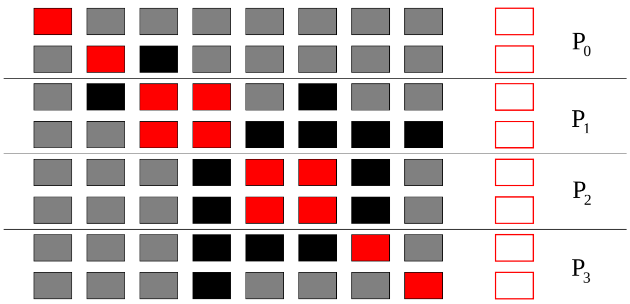

Figure 3: Distribution of matrix and vector elements on 4 processes ($P_0 - P_3$). All nonzero matrix elements are filled. Matrix elements causing communication are marked black and local matrix elements are colored red.

Each process transforms its local rows into the JDS format as described above (see also Ref. [10]). In order to maintain a numerical core similar to that shown in Fig. 1, local and nonlocal vector elements of `x` are stored in the same array. It is apparent that a *Jagged Diagonal* can only be released for computation if the corresponding nonlocal vector elements have successfully been received.

The parallel MVM implementation is based on MPI and uses non-blocking communication routines. After all communication calls have been initialized, we check repeatedly for the completion of any receive call and release *Jagged Diagonals* for computation if possible. This MVM implementation allows overlapping of communication and computation if asynchronous data transfer is supported by the hardware.

4 Benchmark Issues

4.1 Benchmark platforms

The benchmarks runs presented in section 5 have been performed on the platforms given in Table 1. Since the MVM performance is mainly determined by the memory bandwidth a brief summary of the memory hierarchy is given for each processor.

The **CRAY T3E-1200** system is based on the DEC Alpha 21164-600 MHz processor and equipped with 512 MByte per processor. For the benchmark runs we have enabled the stream buffers, which analyze the memory access pattern and prefetch data from main memory to L2 cache at runtime. The MPI implementation is based on the CRAY `mpt.1.4.0.2`.

The **NEC SX5e** vector processor runs at a frequency of 250 MHz using eight-track vector pipelines for arithmetic and memory operations. There is one load and one load/store pipeline which can either work in parallel as two load pipelines or as a single store pipeline. This configuration provides a single processor performance of 1.6 GFlop/s for the vector triad with vectors longer than 1000, which is roughly 2.2 times the performance of a single CRAY T90 vector processor (cf. Ref. [11]).

The **SGI Origin3800** uses the MIPS R12000-400 MHz processor with a total of 56 GByte distributed shared memory and a large L2 cache with a bandwidth of 4.2 GByte/s. In the benchmark runs all processors of a compute node have been used and each MPI process has been fixed to one processor. The benchmark platform ran CPUSETS in combination with LSF.

Platform	#CPUs Memory	PEAK	BW	L1 L2
NEC SX5e HLR Stuttgart	32 80	4.0	32.0	— —
CRAY T3E-1200 NIC Juelich	540 270	1.2	0.6	8 0.096
SGI Origin3800 TU Dresden	128 56	0.8	0.8	32 8
HITACHI SR8000-F1 LRZ Munich	112 × 8 900	1.5	4.0	128 —

Table 1: Specifications of the benchmark platforms. The second column shows the total number of processors and the aggregate memory in GByte. The rightmost columns (3-5) contain single processor specifications, including the performance numbers (PEAK) in GFlop/s, the memory bandwidth (BW) per processor in GByte/s and the sizes of the L1 (in KByte) and L2 cache (in MByte) for the RISC processor based systems. Please note that the HITACHI SR8000 platform is based on eight-way SMP nodes and the NEC SX5e comprises SMP nodes with 16 vector processors each.

The **HITACHI SR8000-F1** processor is basically a IBM PowerPC running at 375 MHz with some important modifications in hardware and instruction set, including a large floating point register set as well as prefetch and preload instructions. In combination with an extensive software pipelining done by the compiler these extensions form the so called Pseudo-Vector-Processing (PVP) feature, which provides a continuous stream of data from the main memory to the processor avoiding the penalties of memory latency. Eight processors form a shared memory (SMP) node with a memory bandwidth of 32 GByte/s which is the same as for a single NEC SX5e processor. Hardware support for fast collective thread operations is provided. Therefore the system can efficiently be used either as a vector-like computer running one process comprising eight threads (**vector-mode** on one node or as an MPP assigning one process to each processor (**MPP-mode**).

4.2 Benchmark procedure

All of the performance data presented in this report have been measured on non-dedicated platforms. With the exception of the NEC SX5e memories as well as processors have been solely used by our benchmark program while the interconnects have been shared with other users. Since we had to share memory and processors on the NEC SX5e the corresponding performance data presented in section 5 give a lower limit only.

For each benchmark run we have executed at least 200 MVM steps and give the average performance number. In the benchmark runs the Hamilton matrix of a microscopic electron-phonon model (cf. section 6) was used and we have fixed the parameters in such a manner that the matrix size can be increased without changing the basic structure of the matrix.

5 Performance analysis

The aim of the performance analysis is to discuss scalability both of the platforms and of the MVM implementation as well as quality of the memory access.

5.1 Sequential performance

Thus, first the sequential performance as a function of the problem size D_{Mat} is discussed (see Fig. 4).

For the SGI Origin we recover the well-known cache effect: At small problem sizes cached data can be reused giving a performance of about 110 MFlop/s. Consistent with the large L2 cache size the performance drops around $D_{Mat} \sim 10^4 - 10^5$ and a performance of about 35 MFlop/s is sustained for **local** memory access. If the matrix size is further increased ($D_{mat} > 2.5 \times 10^6$) **nonlocal** memory on remote compute nodes has to be allocated. Although the network could in principle sustain the same memory-bandwidth for local and remote memory the performance decreases to 25 MFlop/s due to the higher latencies for remote memory accesses.

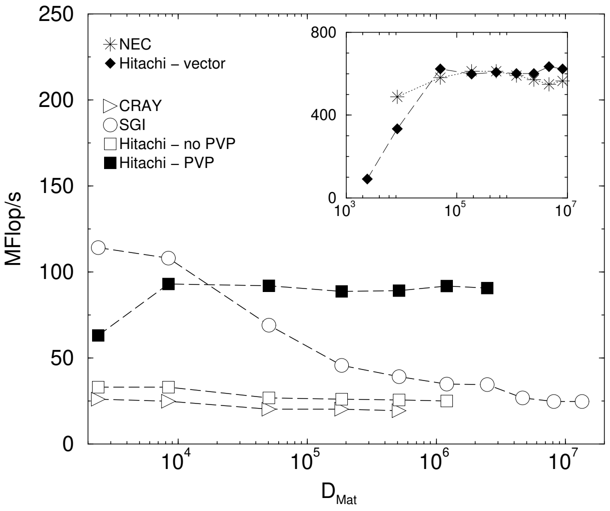

Figure 4: Single processor performance as a function of the matrix dimension D_{Mat}. The inset depicts a comparison of one NEC SX5e **processor** and one HITACHI SR8000 **node** running in vector-mode.

Since the problem sizes in Fig. 4 start at $D_{Mat} \sim 10^3$ there is only a minor cache-effect for CRAY T3E and HITACHI SR8000, due to their small cache sizes. For the HITACHI SR8000 the picture changes completely if we enable the PVP feature. Then the compiler will generate a pseudo-vectorisation of the inner loop, using prefetch streams for the contiguous memory accesses and preload streams for the indirect load. As a consequence a vector-like characterisic is established for the RISC processor: The performance increases with increasing problem size and saturates at long loop lengths. With an asymptotic performance of about 90 MFlop/s, the HITACHI SR8000 is able to maintain a reasonable portion of its high memory-bandwidth and outperforms present-day RISC processors by a factor of 2.5 or even more. Therefore PVP has always been enabled in what follows.

It is straightforward to use the HITACHI SR8000 in the vector-mode by enabling the automatic parallelisation compiler feature which generates a shared-memory parallelisation of the inner loop in Fig. 1. In the inset of Fig. 4 the MVM performance of **one** HITACHI SR8000 SMP **node** (eight processors) is shown together with the measurements for one NEC SX5e vector processor. Again the HITACHI SR8000 realizes a vector-like characteristic and saturates at a performance of about 600 MFlop/s or 75 MFlop/s per processor. The decrease of roughly 15% compared to the single processor performance indicates some memory contention problems in the vector-mode. Nonetheless one SMP node achieves the same performance as the NEC SX5e at large problem sizes. Of course this can be attributed to the memory bandwidth which is the same for both systems. On the other hand, if we reach the small problem size region the vector processor outperforms the SMP node by far because the vector start-up times are 2-3 orders of magnitude shorter than the corresponding collective thread operations on the SMP node.

5.2 Parallel performance

Next we discuss the scalability of code and platforms.

Because of the well balanced ratio between local and remote memory bandwidth the CRAY T3E is widely regarded as a paradigm for scalabiltity. Thus the scalability of our MVM implentation was tested on the CRAY T3E considering the aggregate performance and the parallel efficiency

$$\epsilon_1(N_{proc}) = T(1)/(N_{proc} \times T(N_{proc})) \qquad (3)$$

($T(N_{proc})$ is the time per MVM step on N_{proc} processes) as a function of processors used (see Fig. 5). At fixed matrix

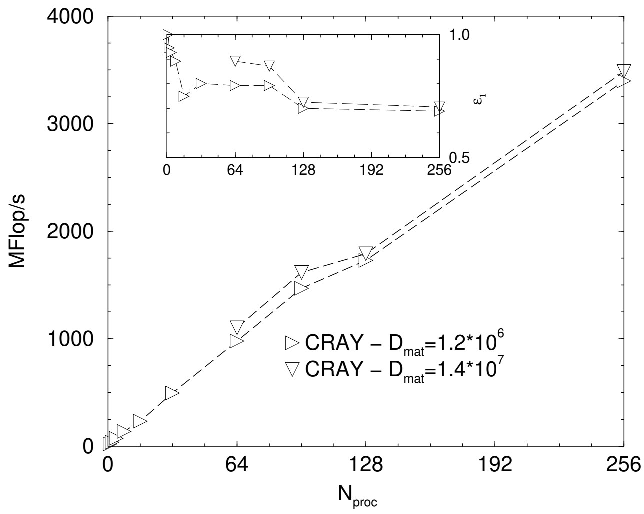

Figure 5: MVM scalability on CRAY T3E for different matrix sizes. The inset shows the corresponding parallel efficiencies ϵ_1. Due to the memory requirements a minimum processor number of 64 had to be used for $D_{mat} = 1.4 \times 10^7$ whereas the asymptotic value for the one processor run in Fig. 4 has been used to calculate the parallel efficiency.

dimension for small to intermediate processor numbers (up to 96) a parallel efficieny of more than 75% can be sustained, which improves with increasing problem size to about 90%. Although a drop in the efficiency is found at 128 processors our MVM basically provides a scalable algorithm because it exceeds an efficiency of 70% even for 256 processors.

In the next step we discuss the MVM performance on SGI Origin3800 when compared to the CRAY T3E (see Fig.

6). At fixed problem size we find that the performance gap between SGI Origin and CRAY T3E widens with increasing processor number, while the parallel efficiency ϵ_1 is lower on the SGI Origin. In that context we have to discuss two

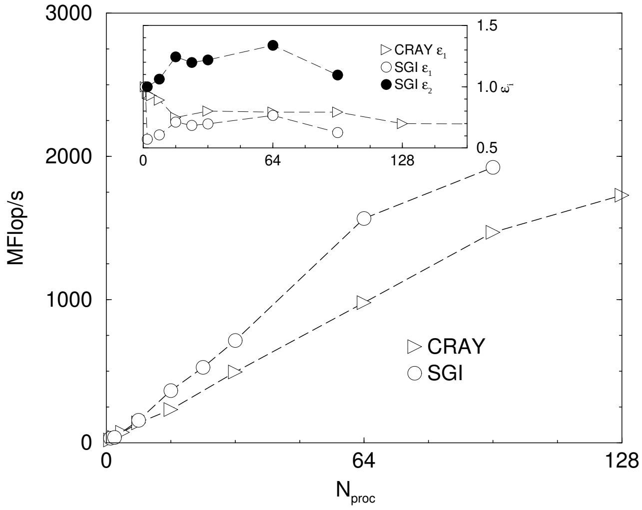

Figure 6: MVM scalability on CRAY T3E and SGI Origin at a fixed matrix size ($D_{Mat} = 1.2 \times 10^6$). The inset shows the corresponding parallel efficiencies ϵ_1 and ϵ_2 (SGI Origin) based on the one and two processor run, respectively.

effects:

1. With increasing processor number the effective problem size per processor is reduced, which is equivalent with a shift to smaller values of D_{Mat} for the one processor performance characteristics given in Fig. 4. Because of the large L2 cache size on the SGI Origin (e.g. the total L2 cache of 16 SGI Origin processors can in principle hold all nonzero elements of the matrix used in Fig. 6) this effect increases the performance gap between both systems.

2. On the SGI Origin two processors have to share one path to the memory. Since our problem is mainly bound by the memory bandwidth it is more reasonable to discuss the scalability of the MVM with respect to the aggregate memory bandwidth. This effect can be taken into account by calculating a modified parallel efficiency ϵ_2 which is determined via equation (3) but now based on the runtime of two processors (see inset of Fig. 6). Supported by the cache effect our MVM implementation achieves superlinear speedup with respect to ϵ_2 on SGI Origin. When increasing the problem size at a fixed processor number, however, the cache effect may reduce the performance gap between CRAY T3E and SGI Origin.

Finally a brief comparison of the CRAY T3E with a vector computer (NEC SX5e) and a SMP based, vector-like system (HITACHI SR8000) is given in Fig. 7. It is

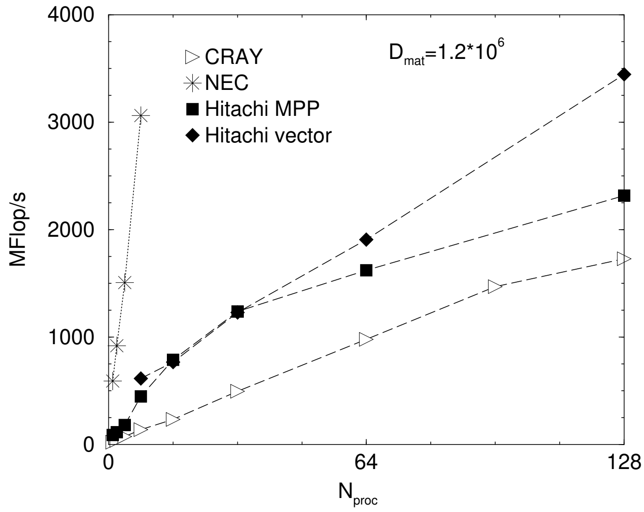

Figure 7: Aggregate MVM performance on CRAY T3E, NEC SX5e and HITACHI SR8000 at a fixed matrix size ($D_{Mat} = 1.2 \times 10^6$). For the HITACHI SR8000 vector-mode one MPI process comprises eight processors, while in all other cases the number of MPI processes is equal to the number of processors.

not surprising that the vector computer provides outstanding single and multiprocessor performance, e.g. 4 NEC SX5e processors achieve approximately the same performance as 96 CRAY T3E processors do. Although the HITACHI SR8000 outperforms the CRAY T3E by a factor of more than four when considering the single processor performance (cf. Fig. 4), the gap decreases dramatically at larger processor numbers using the HITACHI SR8000 MPP-mode (one MPI process is assigned to each processor). Two reasons mainly account for the poor scalability of the HITACHI SR8000 MPP-mode: First the ratio between inter-node communication bandwidth (1 GByte/s) and local memory bandwidth (32 GByte/s) is not as balanced as for the CRAY T3E. Second, the MPI performance suffers a drop if a large number of outstanding messages has to be governed [12]. Due to the eight-way SMP node architecture, the HITACHI SR8000 can also be used with a “hybrid programming model” (vector-mode) which uses both message-passing model (between SMP nodes) and shared-memory model (within SMP node) simultaneously. Evidently the vector-mode scales significantly better than the MPP-mode because the number of MPI processes as well as the number of outstanding MPI messages is reduced by a factor of roughly 8.

In summary the CRAY T3E still offers exceptional scala-

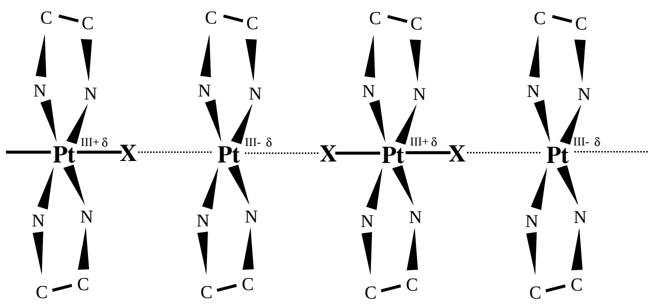

Figure 8: Basic structure of the $[Pt(en)_2][Pt(en)_2X_2] \cdot (ClO_4)_4$ chain material (not shown, for clarity, are the H atoms of the (en) ligands and the ClO_4^- counter-ions.

bility but suffers serious drawbacks when considering memory intensive applications such as sparse MVM. This problem is approached by present-day supercomputers in two different ways: Vector or vector-like systems provide single processors or SMP nodes with large memories, high-peak performance and tremendous memory-bandwidth. On the other hand MPP-like RISC based systems offer scalable and well balanced networks together with large aggregate L2 cache sizes.

6 Application: Localized vibrational modes in one-dimensional charge-density-wave systems

Quasi-one-dimensional MX solids, consisting of chains of transition-metal ions (M= Pt, Pd, Ni) bridged by halogens (X= Cl, Br, I), have been the subject of intense experimental and theoretical study because the various compounds of the family exhibit a remarkable range of strengths of competing electron-electron and electron-lattice forces, and consequent physical properties [13]. A particularly interesting class are the PtX compounds, which are typically Peierls distorted, where the charge disproportionation (alternating valence) of the M sublattice is stabilized by a structural distortion of the X sublattice (see Fig. 8). It has been suggested that in strong-CDW (charge-density-wave) MX materials a dynamical, *intrinsic* localization of vibrational energy in a small segment of the MX chain might take place because both *nonlinearity* (anharmonicity) and *discreteness* are present with sufficient strengths. Indeed, very recently, the experimental observation of such intrinsically localized vibrational modes (ILM's) has been reported in $[Pt(en)_2][Pt(en)_2X_2](ClO_4)_4$ (en= ethylenediamine) [15], subsequently denoted by PtCl. In isotopically pure PtCl the source of the nonlinearity is the strong coupling between electronic and lattice degrees of freedom, and consequently resonance Raman spectroscopy has been used as an ideal experimental technique to measure the energy of the characteristic lattice vibrations associated with the local distortions of these multiphonon gap states. In resonance Raman spectroscopy the material is illuminated with light that is in resonance with a specific electronic transition. The signals from the (fundamental and overtone) vibrational modes that are coupled to the electronic transition are greatly enhanced. Resonant Raman spectra on strong-CDW PtCl exhibiting such an amplification were obtained using Ar^+ laser illumination at 514 nm, which roughly corresponds to the band edge ($\approx$ 2.5 eV) of an intervalence charge transfer (IVCT) transition between Pt^{II} and Pt^{IV} (see Fig. 9). The photo-excited transition into the IVCT band is connected with the excitation of the fundamental Raman active symmetric Cl-Pt-Cl stretch and a progression of many overtones. The ILM's are identified by the strong redshifts they impose upon the overtone resonance Raman spectra.

To discuss the complex interplay of charge, spin and lattice degrees in the PtCl MX-material, we consider a 3/4-filled, two-band tight-binding Hubbard model

$$\mathcal{H}_{\rm el} = \sum_{l\alpha\sigma} \varepsilon_\alpha n_{l\alpha\sigma} - t \sum_{\langle l\alpha, l'\alpha'\rangle\,\sigma} c^\dagger_{l\alpha\sigma} c_{l'\alpha'\sigma} + \sum_{l\alpha} U_\alpha n_{l\alpha\uparrow} n_{l\alpha\downarrow}\,, \quad (4)$$

supplemented by the coupling of the MX electron system to the Raman- and IR-phonon modes

$$\begin{aligned}\mathcal{H}_{\rm el-ph} \;=\; & \lambda_{\rm R}(b_{\rm R} + b^\dagger_{\rm R}) \sum_l (n_{l2} - n_{l4}) \\ & + \lambda_{\rm IR1}(b_{\rm IR1} + b^\dagger_{\rm IR1}) \sum_l (n_{l2} + n_{l4} - n_{l1} - n_{l3})\,, \quad (5)\end{aligned}$$

and the lattice contribution in harmonic approximation

$$\mathcal{H}_{\rm ph} = \hbar\omega_{\rm R} b^\dagger_{\rm R} b_{\rm R} + \hbar\omega_{\rm IR1} b^\dagger_{\rm IR1} b_{\rm IR1}\,. \quad (6)$$

Eqs. (4)-(6) constitute the so-called Peierls-Hubbard model (PHM). Here $c^\dagger_{l\alpha\sigma}$ ($c_{l\alpha\sigma}$) creates (annihilates) an electron

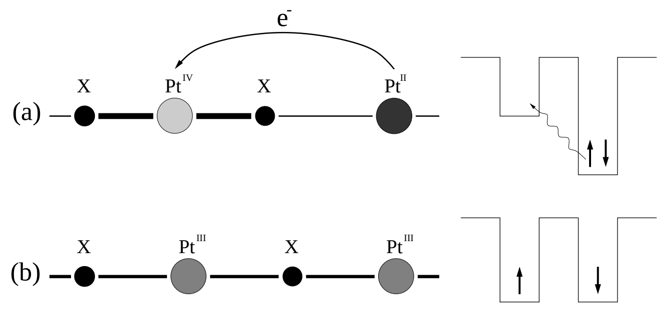

Figure 9: Mixed-valence ground state and IVCT to an excited state without charge disproportionation (a); lattice relaxation and formation of a charge-transfer exciton (b).

with spin projection σ in a Wannier state at site $\{l, \alpha\}$, and $n_{l\alpha\sigma}$ is the corresponding fermion number operator. The phonon destruction (creation) operators are denoted by $b_{\beta}^{(\dagger)}$. Modeling the MX chain materials, l, l' label the unit cells and we use the convention that the M (X) atoms sit on even (odd) sites denoted by the intra-cell index $\alpha, \alpha' = 2, 4$ (1, 3). Their on-site energies ε_{α} can be parameterized by the difference between metal and halogen electron affinities $\Delta = \varepsilon_M - \varepsilon_X$. The other parameters of the PHM are the NN transfer amplitude t, the on-site (Hubbard) electron-electron interactions U_{α}, the electron-phonon couplings ($\lambda_{\rm R}$, $\lambda_{\rm IR1}$), and the bare phonon frequencies ($\tilde{\omega}_{\rm R}$, $\tilde{\omega}_{\rm IR1}$). For the PtCl CDW system, $\Delta = 1.2$, $U_{\rm Pt} = 0.8$, $U_{\rm Cl} = 0$, $\tilde{\omega}_{\rm R} = 0.05$ and $\tilde{\omega}_{\rm IR1} = 0.06$ seem to be appropriate, where the energy scale is given by $t = 1.54$ eV.

In order to understand the evolution of the experimentally observed overtone structure [15] one has to calculate the level shift of the $n^{\rm th}$ excited R-active doublet,

$$r_n = \frac{n\omega_{\rm R}^{(1)} - \omega_{\rm R}^{(n)}}{\omega_{\rm R}^{(1)}} \tag{7}$$

with $\omega_{\rm R}^{(n)} = (E_n - E_0)$, in the framework of the non-adiabatic PHM.

Therefore, in a first step, we determined the low-lying excitations of the $N = 8$ site PHM with periodic boundary conditions at selected electron-phonon couplings, applying the Jacobi-Davidson algorithm outlined in the preceding sections.

Figure 10 shows the energy-level diagram within single-mode approximation ($\lambda_{\rm IR1} = 0$), where the left column displays the spectrum of the decoupled system for comparison. Above each electronic level (solid bars) there is a ladder of overtones (dashed bars) with rungs separated by the bare phonon frequency $\tilde{\omega}_{\rm R}$. Of course, at any finite electron-phonon coupling, the electron and lattice degrees are no longer independent and as a result the excitation spectrum is changed. At weak coupling, however, to a good approximation the ground state is still a zero-phonon state. Then excitations can be obtained simply by adding phonons to the ground state. With increasing electron-phonon interaction strength a strong mixing of electrons and phonons takes place, such that both quantum objects completely lose their individual identity. As a result the ground state is basically a multiphonon state. At this point it seems reasonable to make contact with polaron physics and consider the CDW state of PtCl as built up by ordered bipolarons residing at the $\text{Pt}^{\text{III}-\delta}$ sites (for a more detailed discussion see Ref. [16]). Note that the ground state and, as $\lambda_{\rm R}$ increases, a growing number of excited states show a twofold degeneracy (within numerical accuracy). The reason is the existence of two degenerate CDW ground states in the strong-coupling limit, with large spectral weights of the $|\uparrow\downarrow, \uparrow\downarrow, \uparrow\downarrow, 0\rangle_{\rm el}$ or $|\uparrow\downarrow, 0, \uparrow\downarrow, \uparrow\downarrow\rangle_{\rm el}$ electronic basis states.

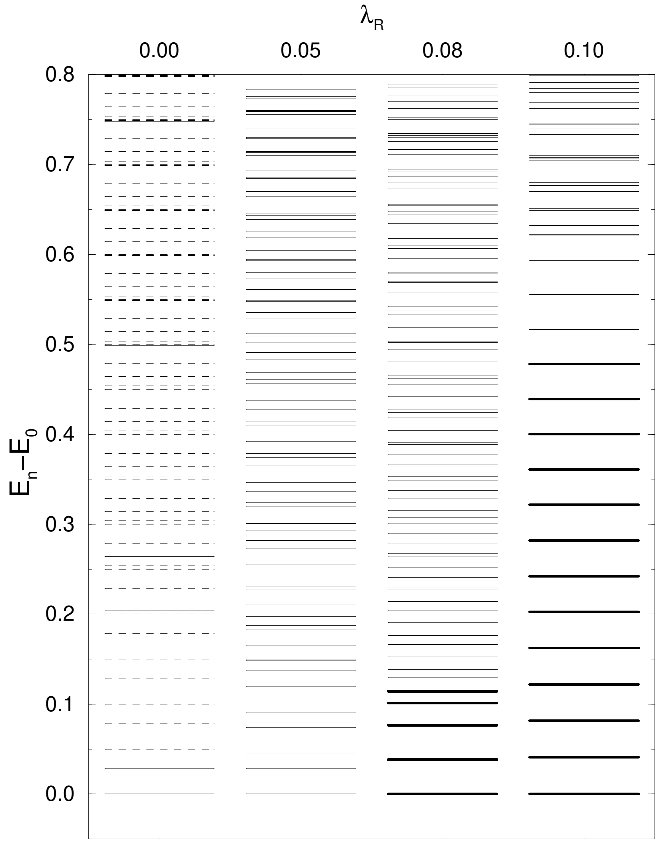

Figure 10: Low-energy part of the eigenvalue spectrum of the Peierls-Hubbard model (single-mode approximation). Twofold degenerate states are marked by bold bars; dashed bars denote the bare phonon overtones of the electronic levels for the $\lambda_{\rm R} = 0$ case.

That means the charge distribution exhibits a CACB modulation.

In the configuration space of the Raman-active normal coordinate this gives rise to the formation of an adiabatic double-well potential. Within one minimum of the double-well potential the low-lying excitations exhibit a large overlap with the displaced oscillator states. As a consequence of the electron-lattice interaction, however, the multiphonon excitations are somewhat shifted from multiples of the bare phonon frequencies $m_{\rm R} \times \omega_{\rm R}$. The important point we would like to emphasize is the weak *anharmonicity* of the double-well potential even at low-energies, provided we consider reasonable coupling strengths. This *nonlinearity*, induced by the coupling to the itinerant *interacting* electron system, is the origin of the redshift of the overtones depicted in Fig. 11. One can state that in the intermediate (but still strong-coupling) region, the Raman-active mode dynamically self-generates a *non-harmonic* lattice potential lead-

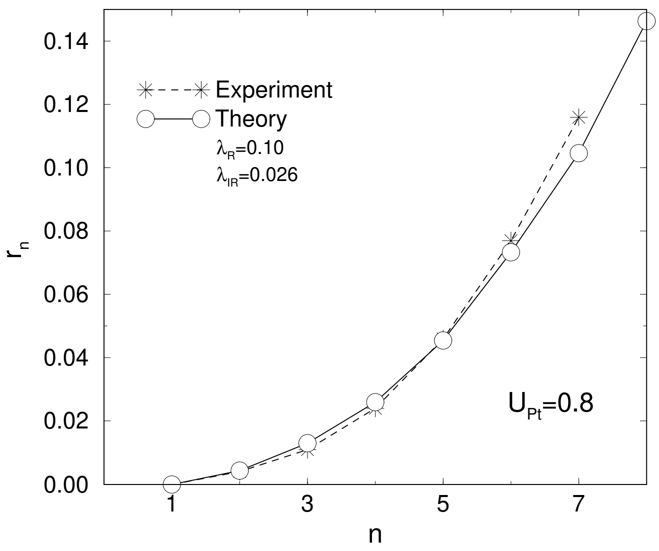

Figure 11: Relative red shift of the lowest-energy peaks, normalized by the fundamental frequency $\omega_R^{(1)}$ given in the inset, as a function of the final quanta of the vibrational energy.

ing to an attractive interaction of Raman phonon quanta located at the same Pt_2Cl_2 unit, with the result that quasi-localized multi-phonon bound states occur in the system. Translational symmetry is restored by quantum mechanical tunneling of those quasiparticles. The nearly perfect quantitative reproduction of the redshift observed in PtCl was obtained using the interaction strengths $\lambda_{\rm R} = 0.1$ and $\lambda_{\rm IR1} = 0.026$.

To summarize, we have shown that a dynamical coupling to the Raman- and infrared-active phonon modes, even in the adiabatic strong-coupling regime, strongly influences the ground-state and spectral properties of the Peierls-Hubbard model. In particular, at appropriate electron-phonon interaction strengths, the effective lattice potential, dynamically self-generated in the process of carrier localization, exhibits a significant nonlinearity, leading to the experimentally observed localization of vibrational energy in PtCl.

Acknowledgements

We are particulary indebted to the C&C Research Laboratories, NEC Europe Ltd., for placing their Jacobi-Davidson program package at our disposal. Special thanks go to the TU Dresden, NIC Jülich, LRZ München and HLR Stuttgart for granting access to their parallel computers. We acknowledge useful discussions with N. Attig, A. R. Bishop, M. Brehm, S. Maletti, R. Rabenseifner, R. Vogelsang and R. Wolff. G.W. and H.F. acknowledge the hospitality at the Theoretical Division of Los Alamos National Laboratory. Part of this work was supported by the *Bavarian Network for High Performance Computing* (KONWIHR).

References

[1] Cullum, J. K., Willoughby, R. A., *Lanczos Algorithms for Large Symmetric Eigenvalue Computations*, volume I & II, Birkhäuser, (Boston 1985).

[2] Davidson, E. R.: J. Comput. Phys., **17**, 87 (1975).

[3] G.L.G. Sleijpen and H.A. van der Vorst, *A Jacobi-Davidson iteration method for linear eigenvalue problems.* SIAM J. Matrix Anal. Appl. 1996; **17**:401–425.

[4] A. Basermann and B. Steffen, *New Preconditioned Solvers for Large Sparse Eigenvalue P roblems on Massively Parallel Computers*, Proceedings of the Eighth SIAM Conference on Par allel Processing for Scientific Computing, CD-ROM, SIAM, (Philadelphia 1997).

[5] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia (1994).

[6] N. Kosugi, *Modifications of the Liu-Davidson method for obtaining one or simultaneously several eigensolutions of a large real symmetric matrix.* Comput. Phys. 1984; **55**:426–436.

[7] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe and H. van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia (2000).

[8] A. Basermann, *Parallel Block ILUT Preconditioning for Sparse Eigenproblems and Sparse Linear Systems.* Numerical Linear Algebra with Applications 2000; **7**:635–648.

[9] J. Dongarra, V. Eijkhout and H.A. van der Vorst, *Iterative Solver Benchmark*, availab le at `http://www.netlib.org/benchmark/sparsebench/`.

[10] M. Kinateder, G. Wellein, A. Basermann and H. Fehske, *Jacobi-Davidson algorithm with Fast Matrix-Vector Multiplication on Massively Paralle l and Vector Supercomputers*, E. Krause and W. Jäger, eds.: High Performance Computing in Science and Engineering 2000 , Springer, Berlin (2001), pp. 188–204.

[11] W. Schönauer, *Architecture and Use of Shared and Distributed Memory Parallel Computers*, eds.: W. S chönauer, ISBN 3-00-005484-7, available at `http://www.rz.uni-karlsruhe.de/Uni/RZ/Pe rsonen/rz03/book`.

[12] M. Brehm, LRZ Munich, privat communication.

[13] A. R. Bishop and B. I. Swanson, *Novel Electronic Materials: The MX Family*, Los Alamos Sciences **21**, 133 (1993).

[14] S. P. Love, S. C. Huckett, L. A. Worl, T. M. Frankcom, S. A. Ekberg, and B. I. Swanson, *Far-infrared Spectroscopy of Halogen-bridged Mixed-valence Platinum-chain Solids: Isotope-substitution Studies*, Phys. Rev. B **47** , 11107 (1993).

[15] B. I. Swanson, J. A. Brozik, S. P. Love, A. P. Shreve, A. R. Bishop, W.-Z. Wang and M. I. Salkola, *Observation of IntrinsicallyLocalized Modes in a Discrete Low-Dimensional Material*, Phys. Rev. Lett. **82**, 3288 (1999).

[16] H. Fehske, M. Kinateder, G. Wellein, and A. R. Bishop, *Quantum lattice effects in mixed-valence transition-metal chain complexes*, Phys. Rev. B **63**, xxx (2001).