A parallel sparse linear solver for nearest-neighbor tight-binding problems

Mathieu Luisier[1], Andreas Schenk[1], Wolfgang Fichtner[1], Timothy B. Boykin[2], and Gerhard Klimeck[3]

[1] Integrated Systems Laboratory, ETH Zurich, CH-8092 Zurich, Switzerland
[2] Department of Electrical and Computer Engineering, The University of Alabama in Huntsville, Huntsville, Alabama 35899 USA
[3] Network for Computational Nanotechnology, Purdue University, West Lafayette, Indiana 47907 USA

Abstract. This paper describes an efficient sparse linear solver for block tri-diagonal systems arising from atomistic device simulation based on the nearest-neighbor tight-binding method. The algorithm is a parallel Gaussian elimination of blocks corresponding to atomic layers instead of single elements. It is known in the physics community as the renormalization method introduced in 1989 by Grosso et al, [Phys. Rev. B **40** 12328 (1989)]. Here, we describe in details the functionality of the algorithm and we show that it is faster than direct sparse linear packages like MUMPS or SuperLU_DIST and that it scales well up to 512 processors.

1 Introduction and Motivation

The simulation of nanoelectronic devices such as ultra-thin-body or nanowire field-effect transistors requires to abandon classical concepts such as drift-diffusion models and to use quantum-mechanical approaches. Furthermore, the strong quantization effects present in aggressively scaled nanostructures can only be captured by models describing the entire bandstructure of a crystal and not only its behavior around some high-symmetry points. The tight-binding approach fulfills these requirements and has become more and more popular among the technology computer aided design (TCAD) community. Each atom of the simulation domain as well as the connection between them are taken into account. They are represented by square blocks whose size depends on the number of atomic orbitals that are kept in the model. For example, in the $sp^3d^5s^*$ nearest-neighbor tight-binding model[1] that is used in this paper, the blocks have a size N_{tb}=10 when spin-orbit coupling is not included.

The goal of device simulation is to obtain observable data, such as current characteristics, that can be compared to experimental data. For that purpose the transport properties of electrons and holes must be investigated. Hence, the tight-binding bandstructure model is incorporated into a transport simulator. This is often achieved in the non-equilibrium Green's function formalism[2] which is computationally very intensive. An alternative is to work in the wave function formalism[3] in which sparse linear systems $\mathbf{AC}=\mathbf{S}$ have to be solved. The matrix

A is block tri-diagonal and of size $(N_A \cdot N_{tb}) \times (N_A \cdot N_{tb})$ where N_A is the number of atoms that the nanostructure contains. The vector **S** describes the injection of states into the device and **C** the resulting wave function.

The size of the matrix **A** as well as the low memory per processor that is available on high performance machines oblige us to solve the sparse system **AC=S** in parallel. To accomplish this task we have developed an algorithm based on the Gaussian elimination of all the diagonal blocks of **A** till only its first and last blocks are connected[4]. It is based on an already existing method known in the physics community as "renormalization method"[5]. When it was introduced in 1989 it was dedicated to very small and one- or two-dimensional problems and it was thought as a sequential algorithm. We have improved it to enable the simulation of realistic three-dimensional nanostructures like nanowire field-effect transistors and to work on massively parallel computers.

The renormalization algorithm is presented in Section 2. The mathematical structure of the matrix **A** and its advantages are discussed in details. In Section 3 the performances of the algorithm are shown on 1 to 512 processors. It is also compared to other direct sparse linear solvers like SuperLU_DIST 2.0[6] and MUMPS 4.6.3[7] on 1 to 16 processors. Apart from a speed-up factor of about 2 or more our renormalization algorithm exhibits better scaling properties than SuperLU and MUMPS.

2 Renormalization algorithm

The matrix **A** corresponds to the Hamiltonian coming from the Schrödinger equation expressed in the tight-binding basis. It is block tri-diagonal and of size $n \times n$ where $n = N_A \cdot N_{tb}$, the number of atoms in the device times the number of orbitals that are kept in the tight-binding model. A silicon nanowire example is given in Fig. 1 (a). On the left the atoms (black dots) and their connections (four per atom, gray lines) are depicted. The resulting matrix **A** is shown on the right. The sparsity pattern corresponding to the zone delimited by the dashed lines is plotted (layers 1, 2, 3, 4, and 5). Note that each black point is in fact a $N_{tb} \times N_{tb}$ matrix. Hence, the total system of equations takes the following form

$$\begin{pmatrix} A_{11} & A_{12} & 0 & \cdots & 0 \\ A_{12}^{\dagger} & A_{22} & A_{23} & 0 & \cdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \cdots & \ddots & \ddots & A_{N_B-1N_B} \\ 0 & \cdots & 0 & A_{N_B-1N_B}^{\dagger} & A_{N_BN_B} \end{pmatrix} \cdot \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_{N_B-1} \\ C_{N_B} \end{pmatrix} = \begin{pmatrix} S_1 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{pmatrix}. \qquad (1)$$

The matrix **A** counts N_B diagonal blocks, each of them refers to a specific atomic layer (all the atoms with the same x-coordinate). The diagonal blocks A_{mm} represent the on-site energy of the atoms situated in the m^{th} layer of the nanostructure as well as the nearest-neighbor connections of these atoms

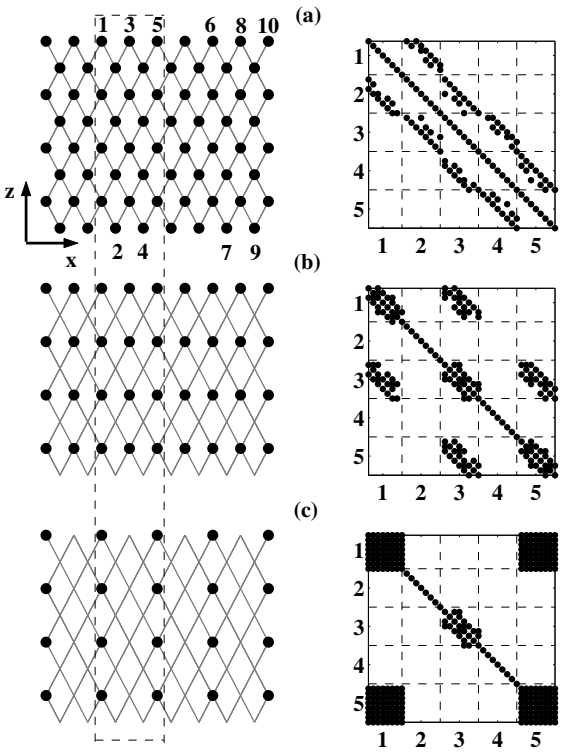

Fig. 1. Nanowire atomic arrangement (left) and corresponding block tri-diagonal matrix (right). Only the $x-z$ face of the nanowire is shown. The black dots depict atoms, the gray lines atomic bonds. The transport direction x is aligned with [100]. The matrices on the right represent the five atomic layers (labeled 1, 2, 3, 4, and 5) surrounded by the dashed lines. Three phases of the renormalization algorithm are represented: (a) initial situation, (b) after the first stage of Gaussian elimination (or layer decoupling), and (c) after the second stage of Gaussian elimination.

within the m^{th} layer. The off-diagonal blocks A_{mm+1} (A_{mm-1}) describe the connections to the atoms situated in the next (previous) atomic layer. All the blocks are real, sparse, and of size $(n_A \cdot N_{tb}) \times (n_A \cdot N_{tb})$, where n_A is the number of atoms per layer. We have the following properties, (1) $A_{mm-1} = A^{\dagger}_{m-1m}$, (2) $A_{mm} = A^{\dagger}_{mm}$, except A_{11} and $A_{N_B N_B}$ which are complex and full and do not have any symmetry since they include the open boundary conditions[3]. The vector S_1 represents the incident states.

As it can be observed in the right part of Fig. 1 (a) the diagonal blocks A_{mm} do not contain connections between atoms and are themselves diagonal. Consequently, their inversion is straight forward and the Gaussian elimination of the atomic layers becomes computationally very efficient. This property is verified for all the nanostructures whose transport direction x is aligned with

the [100] or [111] crystal axis. However, the renormalization algorithm works well as long as one atomic layer is connected only to its adjacent neighbors. This is the case for x=[110] so that all the important crystal orientations are covered.

The basic process of the renormalization algorithm is the decoupling of an atomic layer from its neighbors which is equivalent to the Gaussian elimination of one atomic layer. To decouple one diagonal block m different from 1 and N_B the matrices $\mathbf{M_{L,m}}$ and $\mathbf{M_{R,m}}$ are introduced. They have the same size as $\mathbf{A}$ and are defined as

$$\mathbf{A} = \mathbf{M_{L,m}^{-1}} \cdot \tilde{\mathbf{A}} \cdot \mathbf{M_{R,m}^{-1}}, \tag{2}$$

$$\mathbf{A} = \begin{pmatrix} \ddots & \ddots & \ddots & \cdots & \cdots \\ \ddots & A_{m-1m-1} & A_{m-1m} & 0 & \cdots \\ 0 & A^{\dagger}_{m-1m} & A_{mm} & A_{mm+1} & \ddots \\ \vdots & \ddots & A^{\dagger}_{mm+1} & A_{m+1m+1} & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}, \tag{3}$$

$$\tilde{\mathbf{A}} = \begin{pmatrix} \ddots & \ddots & \ddots & \cdots & \cdots \\ \ddots & \tilde{A}_{m-1m-1} & 0 & \tilde{A}_{m-1m+1} & \cdots \\ 0 & 0 & A_{mm} & 0 & \ddots \\ \vdots & \tilde{A}^{\dagger}_{m-1m+1} & 0 & \tilde{A}_{m+1m+1} & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}, \tag{4}$$

$$\mathbf{M_{L,m}} = \begin{pmatrix} 1 & 0 & \ddots & \cdots & \cdots \\ \ddots & 1 & X^{\dagger}_m & 0 & \ddots \\ \ddots & 0 & 1 & 0 & \ddots \\ \vdots & 0 & Y^{\dagger}_m & 1 & \ddots \\ \vdots & \ddots & \ddots & 0 & 1 \end{pmatrix}, \quad \mathbf{M_{L,m}^{-1}} = \begin{pmatrix} 1 & 0 & \ddots & \cdots & \cdots \\ \ddots & 1 & -X^{\dagger}_m & 0 & \ddots \\ \ddots & 0 & 1 & 0 & \ddots \\ \vdots & 0 & -Y^{\dagger}_m & 1 & \ddots \\ \vdots & \ddots & \ddots & 0 & 1 \end{pmatrix}, \tag{5}$$

$$X_m = A^{-1}_{mm} \cdot A^{\dagger}_{m-1m}, \quad Y_m = A^{-1}_{mm} \cdot A_{mm+1}, \tag{6}$$

$$\tilde{A}_{m-1m+1} = -A_{m-1m} \cdot Y_m, \quad \tilde{A}_{m-1m-1} = A_{m-1m-1} - A_{m-1m} \cdot X_m, \tag{7}$$

$$\tilde{A}_{m+1m+1} = A_{m+1m+1} - A^{\dagger}_{mm+1} \cdot Y_m. \tag{8}$$

The variables carrying a "tilde" are renormalized so that at least one block is decoupled from the others as in Eq. (4). It can be easily proved that $\mathbf{M_{R,m}} = \mathbf{M^{\dagger}_{L,m}}$ and the same relation holds for $\mathbf{M^{-1}_{R,m}}$. Furthermore, by applying the transformation in Eq. (2) only the blocks shown above get modified the other blocks remain unchanged. One solves Eq. (1) by repeatedly decoupling planes until the first layer of $\tilde{\mathbf{A}}$ is connected to its last layer and the rest of the matrix is

block diagonal. The last block $\tilde{A}_{N_B N_B}$ is further decoupled by using a modified matrix $\mathbf{M}^{-1}_{\mathbf{L},\mathbf{N_B}}$ and $\mathbf{M}^{-1}_{\mathbf{R},\mathbf{N_B}}$ where Y_{N_B} is equal to zero. Finally one has

$$\mathbf{A} = \left(\mathbf{M}^{-1}_{\mathbf{L},\mathbf{N_B}} \cdot \mathbf{M}^{-1}_{\mathbf{L},\mathbf{p}} \cdots \mathbf{M}^{-1}_{\mathbf{L},\mathbf{q}}\right) \cdot \hat{\mathbf{A}} \cdot \left(\mathbf{M}^{-1}_{\mathbf{R},\mathbf{q}} \cdots \mathbf{M}^{-1}_{\mathbf{R},\mathbf{p}} \cdot \mathbf{M}^{-1}_{\mathbf{R},\mathbf{N_B}}\right) \tag{9}$$

where $\hat{\mathbf{A}}$ is block diagonal.

The key point of the renormalization algorithm resides in the ordering of the decoupling process. If one starts by decoupling the second block A_{22} no problem is encountered to compute Eqs. (6) to (8) since all the involved matrices are sparse. For example, inverting the diagonal block A_{22} is obvious. However, the first and the third block are modified by this operation. In particular $\tilde{A_{33}}$ is no more diagonal, but still sparse. If now the third block $\tilde{A_{33}}$ is decoupled, $\tilde{A_{11}}$ and $\tilde{A_{44}}$ becomes renormalized. This is computationally not efficient despite the fact that inverting $\tilde{A_{33}}$ can be achieved by a direct sparse linear solver. In effect $\tilde{A_{44}}$ is now a full matrix and the decoupling of the remaining blocks 4 to N_{B-1} is only possible by inverting full diagonal blocks.

A better approach consists in decoupling alternate interior atomic planes to preserve the sparse nature of the matrix $\mathbf{A}$ as long as possible. In the first renormalization stage the blocks 2, 4, 6, $\cdots$ are eliminated. This can be done by inverting diagonal and multiplying sparse matrices. About $(N_B - 2)/2$ blocks are concerned. In the second stage the blocks 3, 7, 11, $\cdots$ are decoupled requiring the inversion of sparse matrices and the multiplication of full and sparse matrices ($\approx N_B/4$ blocks). Finally, only the remaining $\approx N_B/4$ planes require the inversion and the multiplication of full matrices to be decoupled.

This ordering of the decoupling process is illustrated in Fig. 1 (b) and (c). In the first renormalization stage (b) the blocks 2 and 4 are decoupled and the blocks 1, 3, and 5 are renormalized, but keep a sparse pattern. In the second stage (c) the block 3 is decoupled yielding full matrices for the blocks 1 and 5. It is important to notice that the decoupling of all the blocks is accomplished in real arithmetic. Only the last step involving the first and the last block is done in complex arithmetic.

Once the matrix $\mathbf{A}$ is fully block-diagonalized by the procedure described in Eq. (9) the vector $\mathbf{C}$ in Eq. (1) is obtained by a simple recursion involving the X_m and Y_m blocks only[4]

$$C_1 = \tilde{A}^{-1}_{11} \cdot S_1, \quad C_{N_B} = -X_{N_B} \cdot C_1 \tag{10}$$

$$C_m = -X_m \cdot C_{m-p} - Y_m \cdot C_{m+q}. \tag{11}$$

The recursion proceeds by reverse decoupling order and prior to decoupling the block m was connected to the blocks p and q.

The parallelization of the renormalization algorithm is achieved by simultaneously decoupling alternating planes independently. This is similar to a domain decomposition approach. In Fig. 1 it is clear that while a processor is decoupling the blocks (layers) 1 to 5, another processor can proceed to the blocks 6 to 10. The two area are completely independent. Thus, the following parallelization scheme is used if P_{CPU} processors with distributed memory are available. The

matrix $\mathbf{A}$ is divided into P_{CPU} sets of $\approx N_B/P_{CPU}$ subsequent blocks. One and only one set is assigned to each processor P_p. Then the renormalization algorithm is applied until the first block of the processor P_p is only connected to the first block of the processors P_{p-1} and P_{p+1}. To reach this stage no inter-processor communication is required and it remains to decouple P_{CPU} blocks. They are decoupled in $\log_2(P_{CPU})$ steps. In the first step the first blocks of the processors P_2, P_4, P_6, $\cdots$ are decoupled and data are sent via MPI to the processors P_1, P_3, P_5, $\cdots$. In the second step, the first blocks of the processors P_3, P_7, P_{11}, $\cdots$ are decoupled, and so on. At the end the matrix $\mathbf{A}$ is block-diagonalized.

If we assume that the time to decouple a block is t_0 and is the same for all the blocks, are they diagonal, sparse, or full, we obtain the following factorization time $T(p)$ on p processors and speed-up factor $\lambda = T(1)/T(P_{CPU})$

$$T(p) = [(N_B - 1 - p)/p + \log_2(p) + 1] \times t_0, \tag{12}$$

$$\lambda = \frac{N_B - 1}{(N_B - 1 - P_{CPU})/P_{CPU} + \log_2(P_{CPU}) + 1}. \tag{13}$$

Equation (13) is a theoretical value for the speed-up factor since the time to decouple the first half of the planes is much shorter than the time to decouple the last quarter as explained above. Furthermore, in order for λ to reach its ideal value of $\lambda = P_{CPU}$ the number of blocks N_B should be as large as possible.

3 Results

The performances of the renormalization algorithm are analyzed in three tests run on two machines. The first one is a CRAY XT3 with AMD Opteron CPU running at 2.6 GHz and 2 GB of RAM. It is classified at position #84 in the Top 500 list of supercomputers. The second machine is composed of 82 nodes which are Intel Xeon 5140 with 2.33 GHz CPUs and 16 GB of RAM. They are connected with gigabit Ethernet.

3.1 Test 1

The first test is conducted on the CRAY machine. It involves two matrices arising from nanowire field-effect transistor simulation with a length of 600 nm and 1200 nm and a cross section of 2.1×2.1 nm^2. Equation (1) is solved on 16 to 512 processors. The 2GB of RAM per node (1GB per processor) do not allow to consider less than 16 processors. The results of the test 1.a and 1.b as well as the characteristics of the matrices are shown in Fig. 2. On the left the factorization time is reproduced, on the right the solve time. The dashed lines represent the ideal linear scaling. The time (in seconds) to factorize and solve the system of equations (values before the "/") and the speed improvement over 16 processors (values after the "/" and followed by a $\times$) are also reported.

The renormalization algorithm scales well up to 512 processors, where the ideal speed improvement λ over 16 processors should be 32. We have obtained

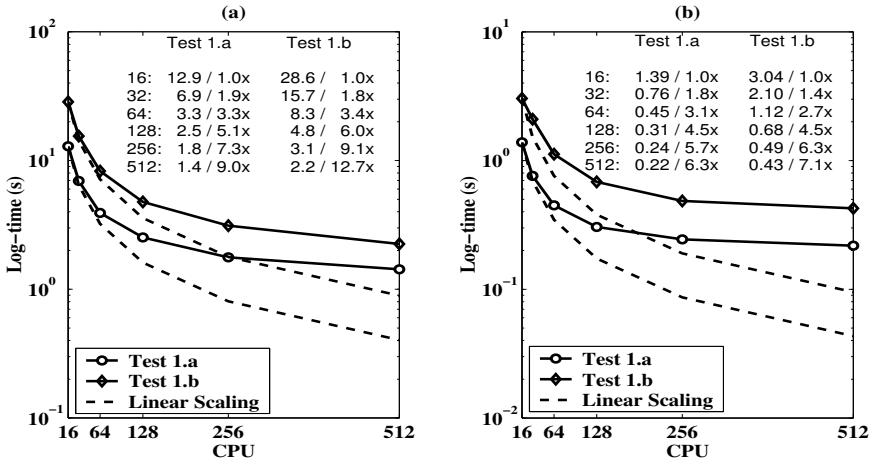

Fig. 2. Scalability results of the renormalization algorithm for (a) the factorization and (b) the solve stage on 16 to 512 CPUs. The matrix in the test 1.a is block tri-diagonal with N_B=3684 diagonal blocks of size 320×320 (total size n=1178880), a band b=720, and nnz=37347328 non-zero elements (sparsity of the band 4.4%). The characteristics of the matrix in the test 1.b are: N_B=7656 (size 350×350), n=2679600, b=820, nnz=72865028, sparsity of the band 3.3%. The execution time and the speed-up factor over 16 CPUs are also reported.

$\lambda_{1.a}$=9.0 and $\lambda_{1.b}$=12.7 for the factorization of the test 1.a and 1.b, respectively. These performances are comparable to those of the "SPIKE" algorithm for "dense in the band" matrices[8]. The λ's estimated by Eq. (13) are larger than the measured λ's since they represent a theoretical limit. However, the measured and the estimated ratio $\lambda_{1.b}/\lambda_{1.a}$ have about the same value of 1.4.

3.2 Test 2

The second test is run on the CRAY XT3 supercomputer for three block tri-diagonal matrices whose size n is linearly proportional to the number of CPUs P_{CPU} that is used to factorize and solve them. Hence, we expect that the factorization and solve time remain constant from 1 to 512 processors. The measured results and the characteristics of the matrices are given in Fig. 3 for the factorization stage. The times are normalized with respect to the time on a single processor. Consequently, in the ideal case, the curves should be close to the constant dashed line $y = 1$. In reality, the factorization time increases as function of P_{proc}, but on 512 processors less than 2× the time on a single processor is necessary for the tests 2.a, 2.b and 2.c.

3.3 Test 3

The Intel Xeon cluster is used for the last test. We consider 7 block tri-diagonal matrices corresponding to realistic nanowire structures with the same length

	n	nnz	b	N_B	sparsity %
Test 3.a	243100	8467450	1240	440	2.8
Test 3.b	291500	10374782	1460	440	2.4
Test 3.c	344300	12519854	1720	440	2.1
Test 3.d	371800	13678784	1840	440	2.0
Test 3.e	401500	14911578	1980	440	1.9
Test 3.f	431200	16182212	2120	440	1.8
Test 3.g	495000	18998168	2420	440	1.6

Table 1. Characteristics of the 7 matrices used for the tests 3.a to 3.g. They have a size n, nnz non-zero elements, a bandwidth b, all the same number of diagonal blocks N_B=440, and the sparsity of the band is given in the last column.

3.a	R. (s)	M. (s)	$\lambda_{\mathbf{M}}$	S. (s)	$\lambda_{\mathbf{S}}$
1	37.7/**1.0x**	95.4	**2.5**	126	**3.3**
2	19.2/**1.96x**	58.4	**3.0**	90.6	**4.7**
4	10.6/**3.55x**	35.6	**3.4**	78.1	**7.3**
8	6.2/**6.08x**	26.2	**4.2**	98	**15.8**
16	4.2/**9.04x**	23.1	**5.5**	364	**87.3**

3.b	R. (s)	M. (s)	$\lambda_{\mathbf{M}}$	S. (s)	$\lambda_{\mathbf{S}}$
1	63.2/**1.0x**	139.3	**2.2**	259	**4.1**
2	32.0/**1.98x**	85.9	**2.7**	166	**5.2**
4	17.5/**3.61x**	50.8	**2.9**	130	**7.5**
8	10.2/**6.19x**	40.4	**3.9**	158	**15.5**
16	6.9/**9.22x**	30.8	**4.5**	531	**77.6**

3.c	R. (s)	M. (s)	$\lambda_{\mathbf{M}}$	S. (s)	$\lambda_{\mathbf{S}}$
1	97.1/**1.0x**	256	**2.6**	428	**4.4**
2	49.7/**1.95x**	143	**2.9**	263	**5.3**
4	27.0/**3.6x**	83.9	**3.1**	195	**7.2**
8	15.9/**6.1x**	57.7	**3.6**	226	**14.2**
16	10.6/**9.16x**	50.1	**4.7**	686	**64.7**

3.d	R. (s)	M. (s)	$\lambda_{\mathbf{M}}$	S. (s)	$\lambda_{\mathbf{S}}$
1	150/**1.0x**	276	**1.8x**	498	**3.3**
2	75.8/**1.98x**	173	**2.3x**	309	**4.1**
4	41/**3.66x**	101	**2.5x**	224	**5.4**
8	23.3/**6.44x**	65.4	**2.8x**	260	**11.2**
16	15.2/**9.88x**	53.5	**3.5x**	795	**52.3**

3.e	R. (s)	M. (s)	$\lambda_{\mathbf{M}}$	S. (s)	$\lambda_{\mathbf{S}}$
1	153/**1.0x**	396	**2.6**	598	**3.9**
2	77.4/**1.97x**	227	**2.9**	360	**4.6**
4	42/**3.63x**	126	**3.0**	260	**6.2**
8	24.6/**6.2x**	101	**4.1**	307	**12.5**
16	16.4/**9.3x**	73.7	**4.5**	855	**52.1**

3.f	R. (s)	M. (s)	$\lambda_{\mathbf{M}}$	S. (s)	$\lambda_{\mathbf{S}}$
1	220/**1.0x**	403	**1.8**	902	**4.1**
2	111/**1.98x**	242	**2.2**	597	**5.4**
4	59.5/**3.69x**	138	**2.3**	358	**6.0**
8	34.1/**6.44x**	111	**3.3**	405	**11.9**
16	22.2/**9.9x**	77.5	**3.5**	1021	**46.0**

3.g	R. (s)	M. (s)	$\lambda_{\mathbf{M}}$	S. (s)	$\lambda_{\mathbf{S}}$
1	330/**1.0x**	568	**1.7**	1604	**4.9**
2	168/**1.97x**	334	**1.99**	919	**5.5**
4	89.5/**3.69x**	181	**2.02**	614	**6.9**
8	51.6/**6.4x**	137	**2.6**	621	**12.0**
16	33.7/**9.8x**	104	**3.1**	1374	**40.8**

Table 2. Factorization results for the tests 3.a to 3.g. The first column indicates the number of processors that were used to factorize the matrix. In the second column the results of the renormalization matrix are shown with the speed-up factor over 1 processor in bold. The third column contains the results of MUMPS 4.6.3 and the fourth the speed improvement $\lambda_{\mathbf{M}}$ vs MUMPS obtained with the renormalization algorithm. The fifth and sixth columns are dedicated to the results of SuperLU_DIST 2.0 and the speed improvement $\lambda_{\mathbf{S}}$ vs SuperLU_DIST obtained with the renormalization algorithm.

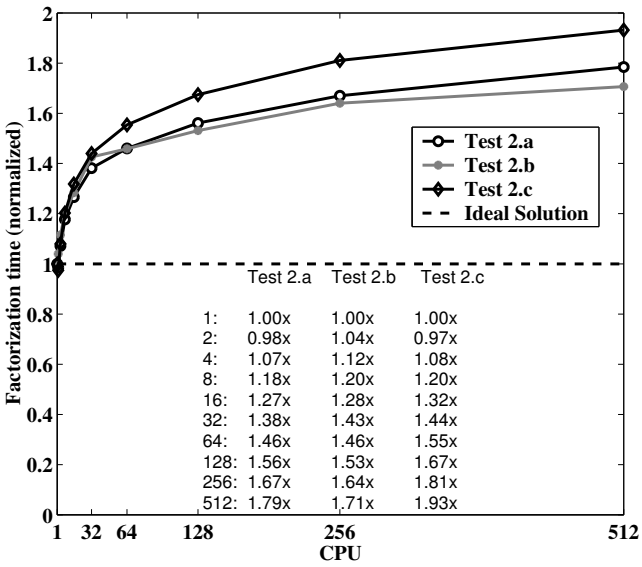

Fig. 3. Factorization time on 1 to 512 CPUs for three different matrices with the size $n = n_0 \times P_{CPU}$ and the number of diagonal blocks $N_B = N_{B0} \times P_{CPU}$ linearly increasing as function of the number of CPU P_{CPU}. The execution time is normalized with respect to the time on a single processor and is reported in the lower part of the figure. In the test 2.a N_{B0}=16, b=720, and n_0=5120, in the test 2.b N_{B0}=12, b=860, and n_0=5280, and in the test 2.c N_{B0}=12, b=800, and n_0=4200.

of 60 nm (=> same number of diagonal blocks N_B), but with a cross section increasing from 2.8×2.8 to 4.1×4.1 nm^2 (=> increasing band b). The characteristics of these matrices are given in Table 1. The number of blocks N_B of these matrices is too small to factorize them on more than 16 processors.

In Table 2 the performances of the renormalization algorithm (second column) are compared to those of MUMPS 4.6.3[7] (third) and SuperLU_DIST 2.0[6] (fifth) on 1 to 16 processors. The factor $\lambda_{\mathbf{M}}$ ($\lambda_{\mathbf{S}}$) in the fourth (sixth) column is the speed improvement of the renormalization algorithm over MUMPS (SuperLU_DIST). In the second column, the bold values refer to the parallel speed improvement of the renormalization algorithm over 1 processor.

First, we observe that the renormalization algorithm scales well for the tests 3.a to 3.g. A mean speed-up of 1.97× is obtained on 2 processors, 3.63× on 4, 6.27 × on 8, and 9.47 × on 16. Then, we find that the renormalization algorithm is between 1.7× and 2.6× faster than MUMPS and between 3.3× and 4.9 × faster than SuperLU_DIST on a single processor. This is due to the fact that both MUMPS and SuperLU_DIST work in complex arithmetic while the renormalization algorithm decouples all the blocks in real arithmetic except the first and the last ones that include the complex boundary conditions. Finally, the speed improvement $\lambda_{\mathbf{M}}$ and $\lambda_{\mathbf{S}}$ increase with the number of processors showing

that the renormalization algorithm scales better than the two other packages.

4 Conclusion

In this paper we presented a parallel sparse linear solver for block tri-diagonal matrices, the renormalization algorithm. A sequential version was introduced in 1989 in the physics community, but we optimized it to treat larger systems and to work in parallel. For matrices with a bandwidth smaller than one thousand and a size larger than one million the renormalization algorithm scales well up to 512 processors. For more realistic structures (bandwidth larger than 1500 and size smaller than 500000) it is faster and it scales better than the direct sparse linear solvers MUMPS and SuperLU_DIST

As the size of the blocks gets larger it will be useful to use parallel solvers like ScaLAPACK to decouple full diagonal blocks. This improvement will create a second level of parallelism inside of the renormalization algorithm. Then, the factorization of linear systems with a large band and a relative small size will scale on more than 16 processors and require less RAM.

5 Acknowledgements

This work was supported by the Swiss National Science Foundation (NEQUATTRO SNF 200020-117613/1). nanohub.org computational resources were used.

References

1. T. B. Boykin, G. Klimeck, and F. Oyafuso: Valence band effective-mass expressions in the $sp^3d^5s^*$ empirical tight-binding model applied to a Si and Ge parametrization. Phys. Rev. B **69** (2004) 115201
2. S. Datta: Nanoscale device modeling: the Green's function method. Superlattices Microstruct. **28** (2000) 253-278
3. M. Luisier, G. Klimeck, A. Schenk, and W. Fichtner: Atomistic Simulation of Nanowires in the $sp^3d^5s^*$ Tight-Binding Formalism: from Boundary Conditions to Strain Calculations. Phys. Rev. B **74** (2006) 205323
4. T. B. Boykin, M. Luisier, and G. Klimeck: Multi-band transmission calculations for nanowires using an optimized renormalization method. Phys. Rev. B (submitted)
5. G. Grosso, S. Moroni, and G. Pastori Parravicini: Electronic structure of the InAs-GaSb superlattice studied by the renormalization method. Phys. Rev. B **40** (1989) 12328
6. X. S. Li and J. W. Demmel: SuperLU_DIST: A Scalable Distributed Memory Sparse Direct Solver for Unsymmetric Linear Systems. ACM Trans. on Math. Software **29** (2003) 110
7. P. R. Amestoy, I. S. Duff, and J.-Y. L'Excellent: Multifrontal parallel distributed symmetric and unsymmetric solvers. Comput. Methods in Appl. Mech. Eng. **184** (2000) 501
8. E. Polizzi and A. H. Sameh: A parallel hybrid banded system solver: the SPIKE algorithm. Parallel Comp. **32** (2006) 177