Parallel Computing 111 (2022) 102920

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Towards electronic structure-based *ab-initio* molecular dynamics simulations with hundreds of millions of atoms

Robert Schade [a], Tobias Kenter [a,b], Hossam Elgabarty [c], Michael Lass [a,b], Ole Schütt [d], Alfio Lazzaro [e], Hans Pabst [f], Stephan Mohr [g,h], Jürg Hutter [i], Thomas D. Kühne [a,c,*], Christian Plessl [a,b]

[a] Paderborn Center for Parallel Computing, Paderborn University, Warburger Str. 100, 33098 Paderborn, Germany
[b] Department of Computer Science, Paderborn University, Warburger Str. 100, 33098 Paderborn, Germany
[c] Department of Chemistry, Paderborn University, Warburger Str. 100, 33098 Paderborn, Germany
[d] Department of Materials, ETH Zürich, CH-8092, Zürich, Switzerland
[e] HPE Switzerland GmbH, Basel, Switzerland
[f] Intel Extreme Computing, Software and Systems, Zürich, Switzerland
[g] Nextmol (Bytelab Solutions SL), Barcelona, Spain
[h] Barcelona Supercomputing Center (BSC), Spain
[i] Department of Chemistry, University of Zurich, Switzerland

ARTICLE INFO

Keywords:
Supercomputing
High-performance computing
Massively-parallel algorithms
Large-scale linear algebra
Ab-initio molecular dynamics
Approximate computing

ABSTRACT

We push the boundaries of electronic structure-based *ab-initio* molecular dynamics (AIMD) beyond 100 million atoms. This scale is otherwise barely reachable with classical force-field methods or novel neural network and machine learning potentials. We achieve this breakthrough by combining innovations in linear-scaling AIMD, efficient and approximate sparse linear algebra, low and mixed-precision floating-point computation on GPUs, and a compensation scheme for the errors introduced by numerical approximations.

The core of our work is the non-orthogonalized local submatrix method (NOLSM), which scales very favorably to massively parallel computing systems and translates large sparse matrix operations into highly parallel, dense matrix operations that are ideally suited to hardware accelerators. We demonstrate that the NOLSM method, which is at the center point of each AIMD step, is able to achieve a sustained performance of 324 PFLOP/s in mixed FP16/FP32 precision corresponding to an efficiency of 67.7% when running on 1536 NVIDIA A100 GPUs.

1. Overview of the problem

1.1. Atomistic computer simulations

The exponential increase in the performance of high-performance computers over the past decades, together with advances in computer science and applied mathematics, has led to the birth of a new way of doing science at the intersection of theory and experiment. This field is generally referred to as computational science and allows for experiments *in silico* that otherwise would be too difficult, expensive, or simply impossible to perform. As a result, computer simulations have been very successful in predicting and rationalizing a large variety of novel physical phenomena.

For systems made of atoms, the two most common computational techniques to conduct such simulations are the Monte Carlo and the molecular dynamics (MD) algorithms [1,2]. The latter is simply the numerical solution of Hamilton's equation of motion, which allows both equilibrium thermodynamic and dynamic properties of a system at finite temperature to be computed. Since it also provides a window into the real-time evolution of the atoms, another role of MD is that of a computational microscope.

One of the most challenging and very important aspects of MD simulations is calculating the interatomic forces. In classical simulations they are computed by conventional force fields, or novel neural network and machine learning potentials, which have been parameterized to reproduce experimental or accurate *ab-initio* data of small model systems [3,4]. Even though great strides in improving such empirical potentials have been made and often render them surprisingly accurate [5,6], the transferability to systems or regions of the phase diagram different from the ones to which they have been trained in the first

* Corresponding author.
E-mail address: thomas.kuehne@uni-paderborn.de (T.D. Kühne).

https://doi.org/10.1016/j.parco.2022.102920
Received 26 July 2021; Received in revised form 17 January 2022; Accepted 21 February 2022
Available online 5 March 2022

place may be restricted. Ultimately, when assuming a classical model, as ingenious it may be, the access to the quantum mechanical electronic structure is irrevocably lost. However, some of the most relevant and interesting phenomena of modern chemistry and physics are inherently non-classical.

1.2. Electronic structure-based ab-initio molecular dynamics

Therefore, an electronic structure-based *ab-initio* MD (AIMD) approach [7,8], where the forces are computed on-the-fly from accurate quantum mechanical calculations, is very attractive since many of these limitations can, in principle, be removed. Nevertheless, the accuracy and increased predictive power of AIMD simulations come at a significant computational cost, which has to be carefully balanced against system size and sampling requirements, thus limiting the attainable length and time scales despite substantial progress [9]. Hence, effective single-particle theories such as Hartree–Fock, density functional theory (DFT), and semi-empirical tight-binding (TB) approaches are to date the most commonly used electronic structure methods within AIMD [10]. However, for very large systems, like those occurring in biology, nanotechnology, materials science, or mechanical engineering, which contain many millions of atoms, self-consistently solving the corresponding Schrödinger-like equations is computationally not feasible even on today's largest supercomputers. This practical limitation stems from the fact that these equations are very high-dimensional eigenvalue problems with up to trillions of unknowns. The computation of all eigenvalues and corresponding eigenvectors requires the diagonalization of the quantum mechanical Hamilton operator that uniquely defines the specific system and typically scales cubically with its size.

1.3. Linear-scaling electronic structure theory

Therefore, novel computational methods that scale linearly with the size of the system to directly calculate the all-important density matrix instead of all eigenvectors would be very desirable, thus making a new class of systems accessible to AIMD that were previously thought not feasible. Several so-called linear-scaling methods have been proposed to circumvent the cubic scaling diagonalization that is the main bottleneck of DFT and TB [11–14]. Underlying all of these methods is the concept of "nearsightedness" [15], an intrinsic system-dependent property, which states that at fixed chemical potential the electronic density depends just locally on the external potential so that all matrices required to compute the Fermi operator will become sparse [16]. When using sparse matrix algebra techniques the property can be exploited to devise computational methods whose memory requirements and computational effort increase only linearly with problem size. However, the crossover point after which linear-scaling electronic structure methods become advantageous has remained rather large, in particular if high accuracy is needed.

1.4. Approximate computing-based submatrix method

Beyond algorithmic improvements, it is also possible to relax the requirement for the accuracy of computations and profit from the substantially improved performance of modern computer architectures for low-precision arithmetic. We demonstrate that by leveraging the approximate computing (AC) paradigm [17,18], the usage of mixed- and low-precision numerics can be rigorously compensated by an appropriately modified Langevin-type equation. The noise within the nuclear forces can be assumed as white, thus facilitating the exact computation of ensemble-averaged expectation values [14,19,20]. One possible route is the linear-scaling sign-method [21], whose accuracy had been previously systematically investigated in detail and it has been demonstrated that good accuracy can be obtained [22]. The chief advantage of the employed sign-method, however, is that it only relies on large sparse matrix–matrix multiplies. Yet, due to the distributed nature of the required matrix multiplications, large-scale applications are usually limited by a communication bottleneck. To avoid this bottleneck, we have extended the recently developed submatrix method [23,24], which transforms calculations on large distributed sparse matrices into computations on small local dense matrices and combined it with the second-generation Car–Parrinello method of Kühne et al. [9,25] to bypass the previously mentioned self-consistent solution. This transformation opens the door to employ hardware-accelerated low-precision linear algebra without compromising the accuracy of the eventual results.

2. Current state of the art

Previous attempts to push the boundaries of electronic structure-based structure relaxation and AIMD simulations are summarized in Table 1. They include DFT calculations using delocalized plane wave (PW) basis sets, as implemented in the CPMD [26], Qbox [27], LS3DF [28] and OpenAtom [29] codes, as well as localized orbital DFT simulations based on real-space finite difference (RS-DFT) and finite element methods (FEM) using the RSDFT [30] and DFT-FE [31] codes, respectively. The largest simulations, however, are conducted employing low-scaling electronic structure methods such as linear-scaling DFT (LS-DFT). With the exception of the LDC-DFT code [32], which relies on an extended real-space multigrid PW (RMG-PW) basis within a less correlated subsystem DFT (SS-DFT) approach, localized basis functions with finite spatial extent are used. Examples of the latter are non-orthogonal generalized Wannier functions (NGWF), finite difference (FD), polarized atomic orbitals (PAO) and Gaussian and plane waves (GPW) basis sets that are implemented in the ONETEP [33], MGmol [34], CONQUEST [35] and CP2K [36] codes, respectively.

To put our achievement in context to previous work, we find it important to point out that the hitherto largest electronic structure calculation conducted so far with 6.3 million atoms has been achieved using the SS-DFT approach, which subdivides the total system in a divide-and-conquer fashion into overlapping fragments that can be computed independently from each other. Even though this offers an intriguing additional level of parallelism, which is reflected by a peak performance of more than 5 PFLOP/s, it also entails a further approximation. Along similar lines, simulations involving multiple independent k-points can also be trivially parallelized over each of these points. Interestingly, the simulation with the so-far largest peak performance of 46 PFLOP/s and an efficiency of 27.8% has been conducted for just 10.5 thousand atoms, even though with electronically rather complicated alkaline earth metals atoms.

In the present work, we have conducted individual AIMD-based dynamical simulated-annealing steps to mimic the relaxation of the structure of a whole human immunodeficiency virus-1 (HIV-1) capsid in aqueous solution containing more than 62.5 million atoms, as well as for water with about 102 million atoms. For that purpose, we have extended the Geometry, Frequency, Noncovalent, eXtended Tight-Binding (GFN-xTB) scheme towards periodic systems and implemented it within the CP2K code [36,45].

3. Innovations realized

3.1. Summary of contributions

The central innovation of this work is the approximate mapping of a matrix function of a very large sparse matrix to a series of matrix functions of much smaller but dense matrices. Since in this way inter-node communication is avoided, a very favorable parallel scaling is obtained. The evaluations of the matrix functions for the small dense matrices, with dimension of ~ 500 to ~ 10000 for the applications in this work, are computed with iterative schemes and mixed-precision arithmetic on tensor cores of GPUs. The resulting noise from these approximations is rigorously compensated by making use of the fluctuation–dissipation theorem so that the desired thermodynamic expectation values can nevertheless be obtained accurately.

Table 1
Performance of previously conducted electronic structure-based structure relaxation or AIMD simulations. Therein, the employed electronic structure method is abbreviated by DFT, NSC-DFT, LS-DFT and SS-DFT, which stands for density functional theory and its non-self-consistent, linear-scaling and subsystem variants, respectively. The corresponding basis set to represent the single-particle orbitals are denoted by PW for conventional plane waves, RMG-PW for real-space multigrid plane waves, GPW for Gaussian and plane waves, GTO for Gaussian-type orbitals, FD for finite difference, RS-FD for real-space finite difference, FEM for finite element method, NGWF for non-orthogonal generalized Wannier functions and PAO for polarized atomic orbitals. If the calculation was conducted involving trivial k-point parallelism, the total number of atoms is given as the product of number of independent instances time the number of atoms in anyone of them. The sustained efficiency is either given with respect to the corresponding peak performance, or estimated in terms of parallel efficiency and identified by the "≈" sign.

Code	Year	Method	Basis	System	# Atoms	# Cores	Machine	Peak performance	Efficiency
CPMD [37]	2005	DFT	PW	Bulk SiC	1k	1.2k CPU	IBM p690	1.087 TFLOP/s	≈ 20%
Qbox [38]	2006	DFT	PW	Bulk Mo	8*1k	128k CPU	IBM BlueGene/L	207.3 TFLOP/s	56.5%
LS3DF [28]	2009	DFT	PW	Bulk ZnTeO	36k	147 k CPU	Cray Jaguar	442 TFLOP/s	≈ 33%
CONQUEST [39]	2010	NSC-DFT	PAO	Bulk Si	2.1M	4k CPU	Cray XT4		≈ 60%
CP2K [40]	2012	LS-DFT	GPW	Bulk H_2	1M	47k CPU	Cray XT5		
ONETEP [41]	2014	LS-DFT	NGWF	Amyloid fibril trimer	42k	115k CPU	IBM BlueGene/Q		
CONQUEST [42]	2014	LS-DFT	PAO	Bulk Si	786k	200k CPU	K-Computer		
RSDFT [30]	2014	DFT	RS-FD	Si nanowire	107k	664k CPU	K-Computer	5.48 PFLOP/s	51.67%
CP2K [43]	2016	SS-DFT	GPW	Satellite tobacco mosaic virus	1M	20k CPU	Cray XC30		
LDC-DFT [32]	2014	SS-DFT	RMG-PW	Bulk SiC	6.3M	786k CPU	IBM BlueGene/Q	5.08 PFLOP/s	50.5%
OpenAtom [29]	2016	DFT	PW	Periodic MOF	32*424	262k CPU	IBM BlueGene/Q		≈ 52%
MGmol [34]	2016	LS-DFT	FD	Bulk H_2O	1.2M	1.6 m CPU	IBM BlueGene/Q		≈ 39%
DFT-FE [44]	2019	DFT	FEM	Mg cluster	10.5k	159k CPU +22.8k GPUs	IBM Summit	46 PFLOP/s	27.8%
This work	2021	LS-DFT	GTO	Bulk water	102M	18.4k CPU +1.5k GPUs	JUWELS Booster	206 PFLOP/s	43%
This work	2021	LS-DFT	GTO	HIV-1 capsid in solution	62.5M	18.4k CPU +1.5k GPUs	JUWELS Booster	324 PFLOP/s	67.7%

3.2. Algorithmic innovations

3.2.1. Approximate computing

The ideas of AC can be applied to the field of electronic structure-based AIMD simulations by recognizing that algorithmic or numerical approximations cause noise in the computed total energy E^N of the system and in consequence noise $\boldsymbol{\Xi}_i$ in the forces $\boldsymbol{F}_i$ on the atoms. Thus, the computed noisy forces $\boldsymbol{F}_i^N$ can be written as

$$\boldsymbol{F}_i^N = -\frac{\partial E^N}{\partial \boldsymbol{R}_i} = \boldsymbol{F}_i + \boldsymbol{\Xi}_i, \tag{1}$$

where $\boldsymbol{F}_i$ denote the exact forces. All quantities depend on the position of the atoms $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_n$. In previous works we have demonstrated that in the present context $\boldsymbol{\Xi}_i$ can be assumed to be nearly unbiased [14,19,20], thus fulfilling the so-called fluctuation–dissipation theorem

$$\langle \boldsymbol{\Xi}_i(0) \cdot \boldsymbol{\Xi}_i(t) \rangle_T \approxeq 2\gamma_N M_i k_B T \delta(t), \tag{2}$$

where $\langle \cdots \rangle_T$ denotes the Boltzmann-weighted ensemble average at the temperature T, k_B the Boltzmann constant, M_i the atomic masses, and γ_N a friction coefficient, whose exact value needs to be determined. However, if we would know γ_N such that Eq. (2) is satisfied, a modified Langevin-type equation

$$M_i \ddot{\boldsymbol{R}}_i = \boldsymbol{F}_i + \boldsymbol{\Xi}_i - \gamma_N M_i \dot{\boldsymbol{R}}_i \tag{3}$$

is recovered, which guarantees for an accurate canonical sampling of the Boltzmann distribution and to compute precise thermodynamic expectation values [9]. Fortunately, the exact value of γ_N does not need to be known *a priori*, but can be bootstrapped so as to generate the correct average temperature [25], as measured by the equipartition theorem

$$\left\langle \frac{1}{2} M_i \dot{\boldsymbol{R}}_i^2 \right\rangle = \frac{3}{2} k_B T. \tag{4}$$

More precisely, in order to determine the hitherto unknown value of γ_N, we perform a short preliminary simulation on an identical but smaller system in which we vary γ_N on-the-fly using a Berendsen-like algorithm until Eq. (4) is eventually satisfied [46]. Previous studies have demonstrated the efficacy of this approach for a wide variety of systems ranging from insulators to semiconductors and even to metals in condensed phases [10,20].

3.2.2. Linear-scaling eigenvalue solver via the non-orthogonalized local submatrix method

3.2.2.1. Linear-scaling electronic structure calculations. In electronic structure-based AIMD simulations, forces $\boldsymbol{F}_i$ on the atoms are derived in every time step on-the-fly from the solution of the quantum mechanical problem of electrons in the electrostatic field generated by the nuclei. The total energy of a system can be written as

$$E = E_{elec} + E_{dc} + E_{ion} = \sum_i^{occ.} \langle \psi_i | \hat{H}_0 | \psi_i \rangle + E_{dc} + E_{ion}, \tag{5}$$

where the summation runs over all occupied electronic states $|\Psi_i\rangle$ in the ground state, the Hamiltonian operator $\hat{H}_0$, additional double counting terms E_{dc} and the nuclear Coulomb repulsion energy E_{ion}. The form of the Hamiltonian matrix $\boldsymbol{H}_0$ and the double counting terms depend on the level of the theory. In any case, however, $\boldsymbol{H}_0$ is dependent on the one-particle density matrix $\boldsymbol{D}$, or on the electron density, which necessitates a self-consistency cycle (SCF). A linear-scaling algorithm to solve the quantum mechanical problem, $\hat{H}_0|\psi_i\rangle = \epsilon_i|\psi\rangle$, is required to find the ground-state energy of the system that determines the forces via Eq. (1). Linear-scaling density-matrix-based electronic structure algorithms directly purify the Hamiltonian into the density matrix $\boldsymbol{D}$ [47], i.e.,

$$\boldsymbol{D} = \frac{1}{2}\left(\boldsymbol{I} - \mathrm{sign}(\boldsymbol{S}^{-1}\boldsymbol{H}_0 - \mu \boldsymbol{I})\right) \boldsymbol{S}^{-1}, \tag{6}$$

where $\boldsymbol{S}$ denotes the overlap matrix and μ the chemical potential. The electronic energy and the forces can now be obtained via

$$E_{elec} = \mathrm{Tr}(\boldsymbol{D}\boldsymbol{H}_0). \tag{7}$$

To evaluate the contribution to the forces from the localized atom-centered basis functions (Pulay forces) [48], the energy-weighted density matrix

$$\boldsymbol{W} = \boldsymbol{D}\boldsymbol{H}_0\boldsymbol{D} \tag{8}$$

is required. The matrix-sign function in Eq. (6) can be evaluated iteratively, for example with the Newton–Schulz iteration [49]

$$\boldsymbol{X}_0 = \boldsymbol{A}, \quad \boldsymbol{X}_{k+1} = \frac{1}{2}\boldsymbol{X}_k(3\boldsymbol{I} - \boldsymbol{X}_k^2) \tag{9}$$

$$\mathrm{sign}(\boldsymbol{A}) = \lim_{k\to\infty} \boldsymbol{X}_k, \tag{10}$$

which converges quadratically if the matrix $\boldsymbol{A}$ has no purely imaginary eigenvalues and $\|\boldsymbol{A}^2 - \boldsymbol{I}\| < 1$ [50]. The matrices $\boldsymbol{A}$ in this work have a real eigenspectrum and the probability that an eigenvalue is numerically zero is negligible. The norm condition is ensured by a rescaling of the matrix prior to the sign iteration. Other possibilities than the Newton–Schulz iteration are higher-order Padé-approximants [51], or arbitrary-order iteration schemes [52]. In conventional linear-scaling schemes the underlying multiplications of large sparse matrices are

performed with global matrix operations [21,36]. In contrast, we view the purification as a matrix function and approximate it with our submatrix method so that no global matrix multiplications are required that would otherwise lead to a communication-bound algorithm.

3.2.2.2. The submatrix method. The submatrix method [23,24], recently developed by some of the authors, approximates a matrix function of a large sparse matrix by evaluating it on a series of much smaller and denser matrices. The underlying idea of the submatrix method is described by Fig. 1. It entails three major steps: (i) submatrix construction, (ii) application of the matrix function and (iii) write-back of the result. In the first step a submatrix is constructed for each column of the original matrix $\boldsymbol{A}$ by removing the rows and their corresponding columns, where the chosen column has zero values. Hence, $\mathcal{T}_j$ constructs the submatrix for column j. Thus, $\mathcal{T}_j(\boldsymbol{A})$ represents the dense submatrix that is constructed for column j of the large sparse matrix $\boldsymbol{A}$. In the second step the matrix function f is applied to every submatrix independently, i.e., $f(\mathcal{T}_j(\boldsymbol{A}))$ for every j. Because the submatrices are by construction much denser and much smaller than the original matrix, efficient dense matrix algorithms can be employed for all operations. In the last step, the write-back step, the relevant column of the matrix $f(\mathcal{T}_j(\boldsymbol{A}))$ is written back into the jth column of the result matrix and represents an approximation of the jth column of $f(\boldsymbol{A})$.

As a result, the submatrix approximation of $f(\boldsymbol{A})$ has the same sparsity pattern as the matrix A. Please note, that the submatrix merging procedure proposed in Section 3.2.3 can make the submatrix approximation of $f(\boldsymbol{A})$ more dense than $\boldsymbol{A}$. Moreover, when applied to well behaved matrices, i.e., not arrowhead-type matrices the dimensionality of the submatrices are independent of the system size for a sufficiently large system (i.e. within the linear-scaling regime). This is for example the case if the submatrix method is applied to the overlap, or Hamiltonian matrix and when localized basis functions are used to describe the electronic wave functions.

The application of the submatrix method to an atomistic system described with local atom-centered basis functions can be understood by identifying the columns (or sets of columns) of the input matrix $\boldsymbol{A}$ with atoms. The construction of a submatrix for a column j corresponding to an atom J can then be seen as the construction of a subsystem containing all atoms in the vicinity of the atom J that have non-zero matrix elements with atom J. For this subsystem, the density matrix is computed as a matrix function and the matrix elements between the atom J and other atoms are used as an approximation to the elements of the density matrix of the full system. Thus, the submatrix method makes use of the intrinsic nearsightedness of the electronic matter and adaptively defines an environment around each atom as a subsystem [15].

Hence, the submatrix method is particularly suitable to estimate the total energy and quantities derived from it. The electronic energy E_{elec} given in Eq. (7) can be written as

$$E_{\text{elec}} = \sum_{i,j} \boldsymbol{D}_{i,j} \boldsymbol{H}_{0,j,i}. \tag{11}$$

Thus, only elements $\{i, j\}$ contribute if $\boldsymbol{H}_{0,j,i} \neq 0$. As long as elements of $\boldsymbol{D}_{j,i}$, where $\boldsymbol{H}_{0,j,i} \neq 0$, are accurately approximated, then the total energy is also estimated accurately, as we have previously demonstrated [23]. This shows that the restriction of the sparsity pattern of $\boldsymbol{D}$ to be the same as for $\hat{H}_0$, which is implied by the submatrix method, is suitable for electronic structure calculations.

3.2.2.3. The non-orthogonal local submatrix method. As an extension of the submatrix method, we propose here the NOLSM method, which for the purification in Eq. (6) views $\boldsymbol{D}(\boldsymbol{H}_0, \boldsymbol{S})$ as a matrix function, i.e., the submatrix idea is applied here for the first time simultaneously to two non-orthogonalized matrices.

Accordingly, the steps of the submatrix method are generalized as follows. In the submatrix generation step the sparsity patterns of $\boldsymbol{H}_0$ and $\boldsymbol{S}$ are merged so that a row i is included in the submatrix for column j if either $\boldsymbol{H}_{0,i,j} \neq 0$, or $\boldsymbol{S}_{i,j} \neq 0$. Thus, the merged sparsity pattern determines the pair of submatrices $\mathcal{T}_j(\boldsymbol{H}_0)$ and $\mathcal{T}_j(\boldsymbol{S})$ that are equal in size and contain the same indices. The matrix functions, i.e., the purification

$$\mathcal{T}_j(\boldsymbol{D}) = \frac{1}{2}\left(\boldsymbol{I} - \text{sign}(\mathcal{T}_j(\boldsymbol{S})^{-1}\mathcal{T}_j(\boldsymbol{H}_0) - \mu\boldsymbol{I})\right)\mathcal{T}_j(\boldsymbol{S})^{-1}, \tag{12}$$

as well as the computation of the energy-weighted density matrix

$$\mathcal{T}_j(\boldsymbol{W}) = \mathcal{T}_j(\boldsymbol{D})\mathcal{T}_j(\boldsymbol{H}_0)\mathcal{T}_j(\boldsymbol{D}) \tag{13}$$

are applied directly to the submatrices to yield the submatrix of the density matrix $\mathcal{T}_j(\boldsymbol{D})$ and the energy-weighted density matrix $\mathcal{T}_j(\boldsymbol{W})$. The entries in columns of these matrices that correspond to the column of the initial matrix form an approximation of the elements in the corresponding columns of the result matrices $\boldsymbol{D}$ and $\boldsymbol{W}$, respectively.

The general arguments for the suitability of the submatrix idea for chemical applications given in Section 3.2.2.2 also holds for the non-orthogonal local submatrix method.

3.2.2.4. The non-orthogonal local submatrix method with GPUs. The schematic for the implementation of the NOLSM method with accelerators is shown in Fig. 2. We discuss here general implementation aspects that are required to obtain an efficient parallel scaling. Within our implementation, the sparse input matrices $\boldsymbol{H}_0$ and $\boldsymbol{S}$ are generated such that each column is completely owned by one MPI rank (Fig. 2 $a_1./a_2.$). Thus, the row indices required for the submatrices of $\boldsymbol{H}_0$ and $\boldsymbol{S}$ for a column j (Fig. 2 $b_1./b_2.$) can be determined and merged without communicating between the nodes (Fig. 2 $c.$). The row information and additional data for the construction of the matrix elements are transferred from the host to the GPU (Fig. 2 $d.$). The matrix elements of the submatrices $\mathcal{T}_j(\boldsymbol{H}_0)$ and $\mathcal{T}_j(\boldsymbol{S})$ are not transferred from the host or from other ranks, but are computed locally (Fig. 2 $e.$) on the GPU. The submatrix of the overlap matrix can then be inverted (Fig. 2 $f.$) and the resulting $\mathcal{T}_j(\boldsymbol{S})^{-1}$ can be used in the purification (Fig. 2 $g.$) given in Eq. (13). The columns of $\mathcal{T}_j(\boldsymbol{D})$ and $\mathcal{T}_j(\boldsymbol{W})$ that correspond to the columns of the original matrices hold the approximate matrix elements of the matrices $\boldsymbol{D}$ and $\boldsymbol{W}$. These elements are transferred from the GPUs to the host (Fig. 2 $h.$) and written to the sparse result matrices $\boldsymbol{D}$ and $\boldsymbol{W}$ (Fig. 2 $i_1./i_2.$), respectively. This write-back is a local operation so that in total the NOLSM method avoids any inter-node communication during the evaluation of the matrix function.

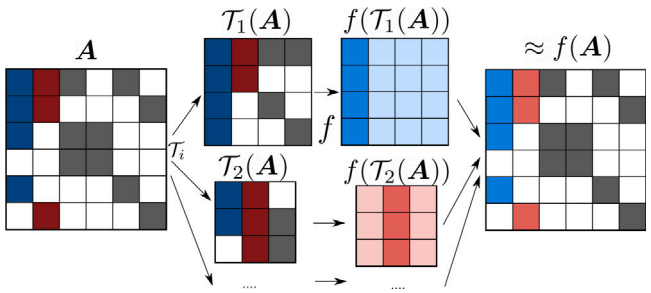

Fig. 1. Schematic representation of the steps of the submatrix method for the approximate calculation of a matrix function $f(\boldsymbol{A})$ of a large sparse matrix $\boldsymbol{A}$. The first step is the construction of a submatrix $\mathcal{T}_i(\boldsymbol{A})$ for every column of the matrix $\boldsymbol{A}$. Then the matrix function is applied to the dense submatrices, i.e., $f(\mathcal{T}_i(A))$ and finally the relevant result columns are inserted into the sparse result matrix.

3.2.3. Submatrix combination heuristics

The NOLSM method as described in Section 3.2.2 can be optimized by generating common submatrices for multiple similar columns instead of individual submatrices for each column. However, it is worthwhile to combine more columns: Let n_i and n_j be the dimensions of submatrices $\mathcal{T}_i$ and $\mathcal{T}_j$, whereas $n_{i\wedge j}$ denote the common rows contained both in $\mathcal{T}_i$ and $\mathcal{T}_j$. Then a combined or merged submatrix $\mathcal{T}_{i,j}$ will

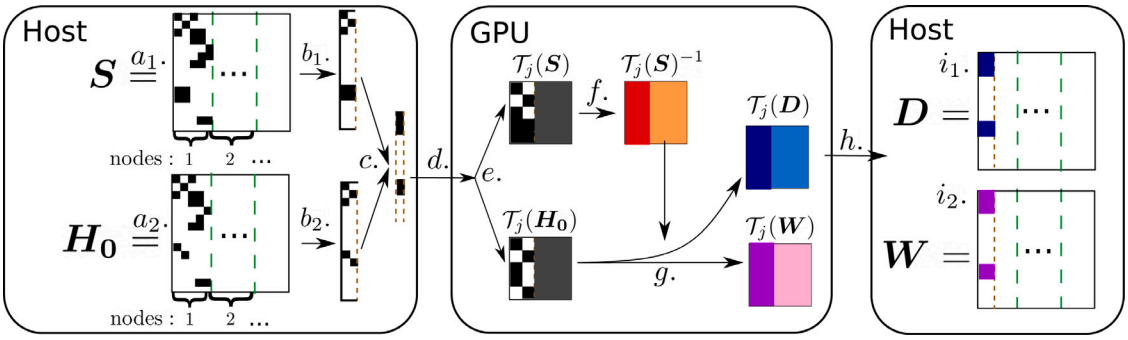

Fig. 2. Schematic representation of the steps within our NOLSM method: $a_1./a_2.$ the overlap matrix $\boldsymbol{S}$ ($a_1.$) and the Hamiltonian $\boldsymbol{H}_0$ ($a_2.$) are stored as large sparse matrices, where each column is owned by a node; $b_1./b_2.$ the row indices for the submatrix of a column or multiple columns are extracted; $c.$ row indices of the column from the Hamiltonian and from the overlap matrix are merged; $d.$ row information together with additional data required for the construction of the matrix elements such as atomic positions are transferred to the GPU; $e.$ the matrix elements of the submatrices are generated; $f.$ the submatrix of the overlap matrix is inverted; $g.$ the Hamiltonian submatrix is purified into the density matrix and the energy-weighted density matrix is calculated; $h.$ the result columns are transferred back to the host and inserted at the corresponding places into the sparse matrices of the density matrix $\boldsymbol{D}$ and energy-weighted density matrix $\boldsymbol{W}$ ($i_1./i_2.$).

contain $n_i + n_j - n_{i\wedge j}$ rows. Considering that the required FLOPs for the evaluation of each submatrix scales cubically, combining $\mathcal{T}_i$ and $\mathcal{T}_j$ into $\mathcal{T}_{i,j}$ yields a speedup if and only if

$$(n_i + n_j - n_{i\wedge j})^3 < n_i^3 + n_j^3. \tag{14}$$

In this work, we use this relation as a strict acceptance criterion for an iterative combination heuristic. This choice is different from our earlier work that used the spatial location of atoms as guiding properties for the combination of submatrices [24]. Also, the presented approach requires no target parameter for the number or size of clusters, but automatically stops when no more improvement of the target metric is found.

For the identification of candidate submatrices $\mathcal{T}_j$ to be merged into $\mathcal{T}_i$, the row entries of $\mathcal{T}_i$ itself are used as a first filter because only submatrices connected in the global sparse matrix tend to have many common row entries (i.e. large $n_{i\wedge j}$). Given this neighborhood information, valid candidates conforming to the criterion from Eq. (14) are considered iteratively in a sequence that depends on the number of unique elements that $\mathcal{T}_j$ would add to $\mathcal{T}_i$, i.e. $n_{j\setminus i} = n_j - n_{i\wedge j}$. For each iteration $n_{j\setminus i} \in \{0, \ldots, n\}$, the valid merge candidates are first identified and prioritized in parallel and then merged into a common representative $\mathcal{T}_{i,j}$. This is a similar process to the position update in the k-means clustering employed by Lass et al. [24]. However, by using the exact row representation of $\mathcal{T}_{i,j}$ instead of a spatial position, it allows for the more exact proximity metric (Eq. (14)), while, at the same time, avoiding separate data structures and updates for nearest-neighbor information, thus also facilitating the required scaling to many millions of atoms and submatrices. As the combination approach is not performed on individual columns, but on groups of all columns corresponding to each atom, the initial set of submatrices corresponds directly to the number of atoms. Applying the heuristic from scratch for a system of about 62.5 million atoms takes about two hours on a single compute node and the result can be used for many AIMD steps. Although in the current work the merging was performed for the full system, parallelization is possible. For that purpose, the system can be subdivided into sufficiently large subcells and the submatrix merging can be performed within every subcell independently. The borders between the subcells can either be treated in a second step by clustering in the border area, while forbidding modifications of clusters that do not touch the border, or simply by neglecting them. In this way, a linear-scaling procedure is obtained.

Yet, instead of fully recomputing the optimal merging of submatrices, if atoms have moved appreciably from the positions for which the merging was originally conducted, simple update strategies of the merging are possible by splitting up merged submatrices and recombining them with other submatrices on-the-fly. However, the development of such update approaches and the necessary row exchanges between the MPI ranks is a topic for future research.

3.3. Implementation innovations

3.3.1. Distributed block compressed sparse row library: DBCSR

The Hamiltonian and the overlap matrix have an underlying block-structure originating from the fact that multiple spatial basis functions describe the electronic wave function in the vicinity of an atom and each basis function corresponds to a column of the matrices. The DBCSR sparse matrix library [53], which handles the sparse matrix operations in CP2K [36], stores such small blocks in a dense format, while referencing these blocks aka non-zero elements in CSR format. The library is used in this work for the storage and operations on sparse matrices outside of the submatrix method.

3.3.2. Minimization of communication

Due to the favorable parallel properties of the NOLSM method it avoids inter-node communication by construction. The transfers between CPU-main memory and GPUs are minimized by constructing the matrix elements of the submatrices directly on the GPUs. Thus, only metadata, i.e., the row indices required for a submatrix, as well as atom species information, atomic positions for the atoms involved in this submatrix and additional data for the underlying electronic structure method have to be transferred. To reduce the overhead of the matrix-element generation routines on the GPUs, an automatic code generation approach was employed that directly yields expressions for the matrix elements of the overlap matrix between two atomic species that only depend on the distance vector between two atoms. The matrix elements are computed in FP32 to make efficient use of the special function units in NVIDIA GPUs for single-precision floating-point transcendental functions [54]. In addition, we employ a load-balancing between the GPUs in a compute node so that submatrices are assigned to GPUs dynamically.

3.3.3. Efficient iterative matrix function solvers on GPU tensor cores

The matrix function for the submatrices, which in the present case is the purification in Eq. (13), can be solved with libraries for dense linear algebra. However, we solve this problem with lower precision linear algebra, specifically making use of the low-precision matrix-multiplications available with tensor cores in GPUs such as the NVIDIA A100 that support FP16 (with FP32 accumulation), bfloat16 (with FP32 accumulation), TensorFloat32 and FP64 [55]. For this purpose, the matrix inversion $\mathcal{T}_i(\boldsymbol{S})^{-1}$ is performed by an iterative scheme [51]

$$\boldsymbol{Y}_0 = \mathcal{T}_i(\boldsymbol{S}), \quad \boldsymbol{Z}_0 = \boldsymbol{I}, \quad \boldsymbol{V}_k = \boldsymbol{Z}_k \boldsymbol{Y}_k \tag{15}$$

$$\boldsymbol{Y}_{k+1} = \frac{1}{2}\boldsymbol{Y}_k(3\boldsymbol{I} - \boldsymbol{V}_k), \quad \boldsymbol{Z}_{k+1} = \frac{1}{2}(3\boldsymbol{I} - \boldsymbol{V}_k)\boldsymbol{Z}_k \tag{16}$$

$$\boldsymbol{Y}_0^{-\frac{1}{2}} = \lim_{k\to\infty} \boldsymbol{Z}_k, \quad \mathcal{T}_i(\boldsymbol{S})^{-1} = (\boldsymbol{Y}_0^{-\frac{1}{2}})^2 \tag{17}$$

and the sign function with the Newton–Schulz iteration given in Eqs. (9)–(10) [24]. We perform all generalized matrix multiplications (gemm) with the tensor cores in mixed precision: FP16 with FP32-based accumulation. To reduce the overhead, the convergence of Eqs. (9)–(10) and (15)–(17) is not checked in every iteration, but a fixed number of steps was used. For the matrix multiplications, kernels from cuBLAS are used. Although individual small- or intermediate-sized matrix multiplications (dimension $\lesssim 2000$) cannot saturate an NVIDIA A100, we obtain a significant portion of the floating-point performance of the tensor cores by employing three additional strategies: firstly by implementing a suitable caching strategy for the matrices during the iterations in Eqs. (9) and (16), secondly by using concurrency via CUDA streams and thirdly by reducing the kernel launch overhead with CUDA graphs [54]. To reduce the number of required CUDA graphs and the corresponding graph recording overhead, the matrices are padded to multiples of 256. Furthermore, GPU memory allocations are reused.

4. How performance was measured

4.1. Computational details

To investigate the performance of the NOLSM method, we have chosen two kinds of systems: liquid water at ambient condition as a homogeneous system and the HIV-1 capsid solvated in water as an inhomogeneous system.

4.1.1. Simulation details

In all of our simulations the GFN-xTB approach is employed in conjunction with a London dispersion correction based on the rational Becke–Johnson damping function [56]. Therein, the electronic states are represented by localized Gaussian-type orbitals. Specifically, every hydrogen atom is represented by two, sulfur by nine, and all other elements used in this work by four basis functions, respectively. As described in Ref. [36], we have extended the GFN-xTB method towards periodic boundary conditions. For that purpose, the long-range electrostatic is computed using the fast Fourier transformation (FFT)-based smooth particle mesh Ewald summation with a spline interpolation of fifth order [57].

Within our NOLSM method, all submatrices of individual atoms, i.e. all columns corresponding to basis functions of the corresponding nuclei, are coalesced in a single submatrix by default. Moreover, the chemical potential has been fixed to a value that ensures the overall charge neutrality of the system. To that extent, a bisection-like mechanism as an outer loop for the determination of the chemical potential is implemented to satisfy the charge neutrality constraint. Such a procedure can also be directly integrated into the submatrix method as shown in [24]. In the spirit of the second-generation Car–Parrinello AIMD method, the electronic state is propagated in time by means of fictitious dynamics, thereby completely avoiding the computationally expensive SCF cycle [9,10]. Therewith individual AIMD-based dynamical simulated annealing steps with a discretized time-step of 0.5 fs were performed. The modified Langevin-type equation of Eq. (3) is integrated using the algorithm of Ricci and Ciccotti [58].

4.1.2. Water

The water benchmark is derived from the linear-scaling DFT benchmark included with CP2K [59]. The basic cell contains 32 water molecules and was equilibrated at a temperature of 300 K and a pressure of 1 bar. This cell is cubic with a length of 9.8 Å. The cell is then repeated in all spatial directions to create a scalable benchmark case. The submatrix combination heuristics is not used for water so that there is one atom per submatrix.

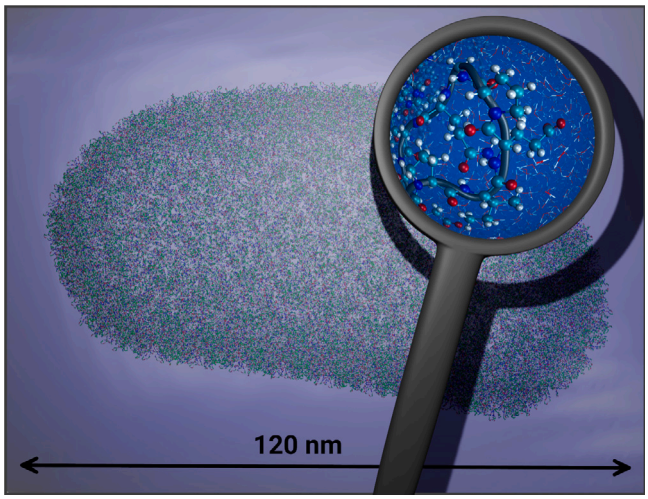

Fig. 3. Graphical representation of the present HIV-1 capsid in aqueous solution containing more than 62.5 million atoms.

4.1.3. HIV-1 capsid

The three-dimensional atomic structure of the entire HIV-1 viral capsid (PDB 3J3Q) was used as a starting point [60]. The structure is composed of 313,236 amino acid residues with a total of 2,440,800 atoms. Missing hydrogen atoms were added such that all the terminal amino acids, the side chains of lysine, arginine, aspartate, glutamate, and glutamine residues, were all in the charged state. The protonation states of the histidine residues were assigned based on the local hydrogen bonding patterns. The capsid was then placed in an orthorhombic unit cell of dimensions $1183.9 \times 800.5 \times 667.8$ Å and the entire structure was solvated in water. Thereafter, the total charge of the system was neutralized by randomly replacing water molecules with sodium ions. The final system, which is shown in Fig. 3 and deposited at [61], has a total of 62,589,576 atoms, i.e. 40,910,985 hydrogen, 1,537,704 carbon, 429,852 nitrogen, 19,689,348 oxygen, 4059 sodium and 17,628 sulfur atoms, respectively.

4.2. Measurements

The main measurements presented here are:

4.2.1. Wall clock time of the NOLSM method T_{NOLSM}

The wall clock time T_{NOLSM} of the core computational routine NOLSM method, i.e., steps $d.-h.$ in Fig. 2 are measured. This includes all transfers between host and GPU.

4.2.2. FLOPs in the NOLSM method $\mathrm{FLOPs}_{\mathrm{NOLSM}}$

The floating-point operations $\mathrm{FLOPs}_{\mathrm{NOLSM}}$ in the FP16/FP32-mixed-precision matrix iterations in the NOLSM method are estimated as $2n^3$ for a gemm-operation $\boldsymbol{C} = \alpha \boldsymbol{A} \cdot \boldsymbol{B} + \beta \boldsymbol{C}$ with $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} \in \mathbb{R}^{n \times n}$. The construction of the matrix elements of the submatrices, which is performed in FP32, is neglected here because they constitute a small fraction of the total workload and due to the ambiguity of counting the exponential functions as floating-point operations. Other operations that scale as $\mathcal{O}(n^2)$ in the size of the submatrices such as norms and scalings are also neglected in the FLOP count.

4.2.3. Sustained performance of NOLSM method P_{NOLSM}

To judge the sustained performance obtained from the GPU-acceleration we define the sustained performance P_{NOLSM} of NOLSM method as $P_{\mathrm{NOLSM}} = \mathrm{FLOPs}_{\mathrm{NOLSM}}/T_{\mathrm{NOLSM}}$.

4.2.4. Wall time clock for one AIMD step $T_{\mathrm{MD-step}}$

The wall clock time $T_{\mathrm{MD-step}}$ for a single AIMD step is measured as the average over at least three AIMD steps. This includes all operations required for one AIMD step, i.e., also IO. This time does not include operations that are only performed once per full MD simulation such as MPI and GPU initialization/finalization or setup of the physical system as well as the heuristic for combining submatrices.

4.2.5. Time-to-solution T_{sol}

With the quantities introduced above we define the time-to-solution T_{sol} as the wall-clock time $T_{\mathrm{MD-step}}$ for a single AIMD step. This definition is reasonable because the number of AIMD steps per MD calculation can vary greatly depending on the physical or chemical objective of the calculation. Often many thousands or more steps are performed so that operations performed only once per MD calculation can be neglected.

4.3. HPC system and environment

The benchmark calculations have been performed on the JUWELS Booster [62]. The system is ranked as number 7 on the TOP500 list as of winter 2020 with a peak double-precision performance of nearly 71 PFLOP/s [63]. Each of the 936 compute nodes of the JUWELS Booster is a dual-socket system with AMD EPYC 7402 24-core CPUs in NPS4-configuration with 512 GB of DDR4 main memory. Each socket is connected to an individual PCIe-switch that in turn is connected to two NVIDIA A100 GPUs and two Mellanox HDR200 InfiniBand ConnectX6 HCAs. Thus, the theoretical inter-node-communication bandwidth is 800 GBit/s. The four GPUs per node are fully interconnected with NVLink3. The cluster interconnect is configured in a DragonFly+ topology with groups of 48 nodes forming a non-blocking cell. The cells are interconnected with 10 links between each cell.

The NVIDIA A100s in the JUWELS BOOSTER have 40 GB of HBM2 memory with a peak memory bandwidth of 1555 GB/s. The theoretical peak performance of the non-tensor-core execution units is 9.7 TFLOP/s in FP64, 19.5 TFLOP/s in FP32 and 78 TFLOP/s in FP16. The peak performances when using the tensor cores are listed as 19.5 TFLOP/s in FP64, 156 TFLOP/s in TF32 and 312 TFLOP/s in FP16 with FP32-based accumulate [55].

The relevant components of the software environment used in this work are GCC 9.3.0, OpenMPI 4.1.0, CUDA NVCC 11.0.221, and CUBLAS 11.0. All benchmarks have been performed with one MPI-rank per node to minimize data replication and 48 CPU-threads per rank. Four CUDA streams are used per GPU and each stream is controlled by a single CPU-thread.

5. Performance results

5.1. Performance of the NOLSM method

The performance and scaling of the core steps of the NOLSM method P_{NOLSM}, i.e., steps $d.-h.$ shown in Fig. 2 including transfers to/from GPUs are evaluated in this section.

5.1.1. Scaling

The physical system used for weak scaling investigations is water, as described in Section 4.1.2. To allow for scaling from a single node, we have used 22 × 22 × 22 basic cells as a basic unit. This basic unit holds 1,022,208 atoms (1M) and is repeated in z-direction. For n compute nodes used, we have n such basic units. For the strong scaling starting from one node, this single basic unit is used. Additionally, strong scaling is also shown for the large-system case with 102 × 102 × 102 basic cells corresponding to 101,875,968 atoms (102M). Fig. 4 shows the wall time and parallel efficiency results. Due to the favorable parallel nature of the non-orthogonalized local submatrix method, a parallel efficiency very close to one is achieved. In addition, Fig. 5 shows the achieved floating-point performance P_{NOLSM} and fraction of peak performance corresponding to the results shown in Fig. 4. As defined in Section 4.2, the floating-point operations counted here only include matrix multiplications on the tensor cores in mixed precision. Hence, the theoretical peak performance of 312 TFLOP/s per GPU in FP16-based matrix multiplies with FP32 accumulation was used for comparison. A fraction of about 43% of the theoretical peak performance is achieved in this example or, in other words, about 206 PFLOP/s for 384 compute nodes.

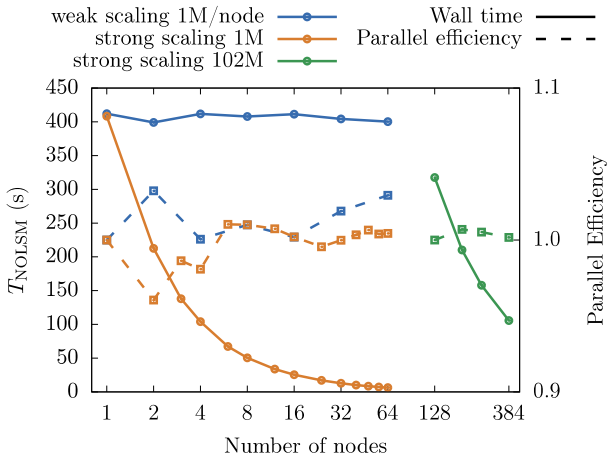

Fig. 4. Strong and weak scaling behavior of the NOLSM method for water: weak scaling for 1,022,208 atoms per node (1M) and strong scaling for 1,022,208 atoms (1M) as well as 101,875,968 atoms (102M).

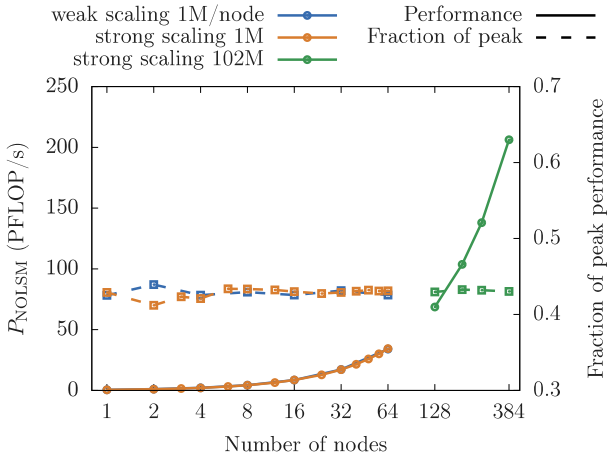

Fig. 5. Sustained performance of the NOLSM method and fraction of theoretical peak performance achieved in the strong and weak scaling calculations for water shown in Fig. 4.

5.1.2. Performance of matrix iterations on NVIDIA a100

Fig. 6 shows the increase of performance of cuBLAS-based square matrix multiplications on the tensor cores of the NVIDIA A100 from additional techniques like multiple CUDA streams and CUDA graphs. As a comparison also the performance of the matrix-sign iteration of Eqs. (9)–(10) while using these techniques is shown. Due to the cache-friendly nature of the matrix iteration, the overall performance is higher than for individual matrix multiplications of the same size. The well-known deficiency of the performance for matrix multiplications of small matrices persists but is mitigated by these additional techniques.

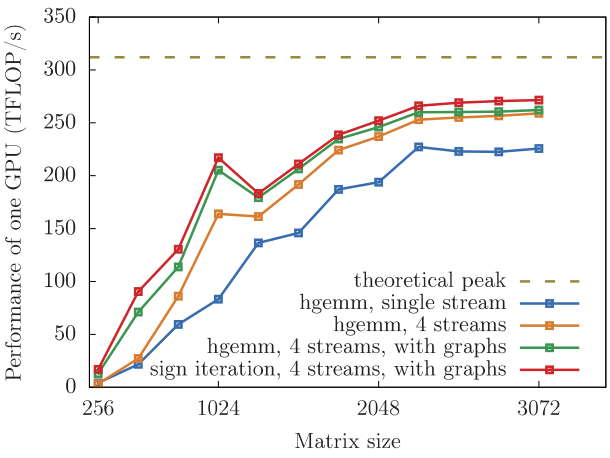

Fig. 6. Performance of mixed-precision FP16 with FP32-accumulate matrix multiplications (hgemm) compared to the sign iteration of Eqs. (9)–(10) with cuBLAS on NVIDIA A100 for different sizes of square matrices and with or without CUDA streams or CUDA graphs.

5.1.3. *Effect of submatrix combination heuristics and transition to HIV-1 system*

As discussed in Section 3.2.3, the main benefit of applying the submatrix combination technique is a reduction of the total workload. We investigate the effect with the example of the HIV-1 capsid with about 62.5 million atoms introduced in Section 4.1.3. The heuristic for the combination of submatrices described in Section 3.2.3 reduces the cubic work metric that is used as combination criterion in Eq. (14) by a factor of ≈ 1.65. The second benefit of submatrix combination is to increase the average submatrix dimensions into regions where the matrix iterations on GPU reach higher performance as demonstrated in Fig. 6. Fig. 7 shows the effect of submatrix combination on the numbers and sizes of submatrices for the HIV-1 capsid system. We see that the mean and median submatrix sizes increase, most notably by moving the peak of most common sizes from between 400 and 800 elements to between 800 and 1600 elements with an order of magnitude fewer submatrices. Furthermore, we see two disjoint large groups of sizes. The group of smaller submatrices contains up to around 1200 elements before combination and up to 3200 elements afterward, whereas a group of large submatrices exists between 4000 and 10200 elements that changes little by the combination of submatrices. While the combination process itself is completely driven by neighborhood properties encoded in global matrix entries and thus is agnostic of atom types, the large submatrices are formed around sodium atoms. After combination, most of the other submatrices are formed around hydrogen atoms.

The transition from water to the HIV-1 capsid also impacts the calculation of the matrix elements of the submatrices on the GPU. It becomes more elaborate because matrix elements involving species like sulfur require more effort than, for example for hydrogen or oxygen. For example, a matrix element from our automatic code generation approach between two sulfur atoms requires roughly ten times more exponential functions than a matrix element between two oxygen atoms. However, the impact of this is countered by a third effect of the submatrix combination heuristics: not only the cubic target metric of FLOPs during the matrix multiplications is reduced, but at the same time also a quadratic metric of submatrix elements used i.e., $\sum_i n_i^2$ improves by a factor of ≈ 3.3, which benefits the construction of the submatrices on the GPUs (see step *e.* in Fig. 2).

Fig. 8 shows the resulting strong scaling of the NOLSM method for the sustained performance and fraction of theoretical peak performance after combining submatrices for the HIV-1 capsid. Please note that the timing for submatrix combination is not included in the measurements because the combination can be precomputed and has to be refreshed only occasionally during an MD calculation. Thus, the NOLSM method has been shown to reach a sustained performance of about 324.7 PFLOP/s for 384 compute nodes and a fraction of the theoretical peak performance of 67.7% for the HIV-1 capsid with about 62.5 million atoms. In addition, Fig. 9 shows the strong-scaling wall time and parallel efficiency for the NOLSM method for the HIV-1 capsid.

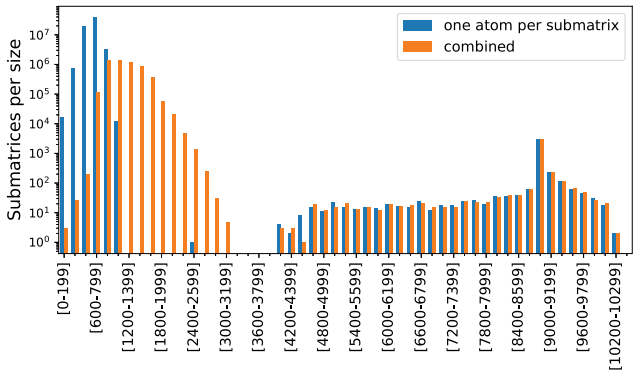

Fig. 7. Number of submatrices by groups of sizes n_i. The situation with one atom per submatrix resulting in 62,589,576 submatrices is compared to the result of the submatrix combination heuristics that yields 5,536,116 submatrices.

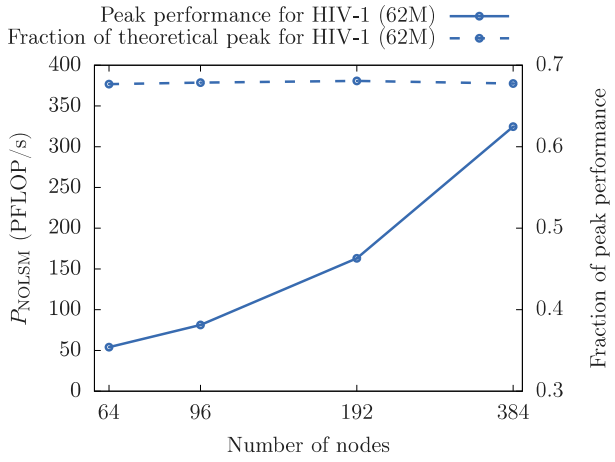

Fig. 8. Strong scaling behavior of the NOLSM method for the HIV-1 capsid with ≈ 62 million atoms.

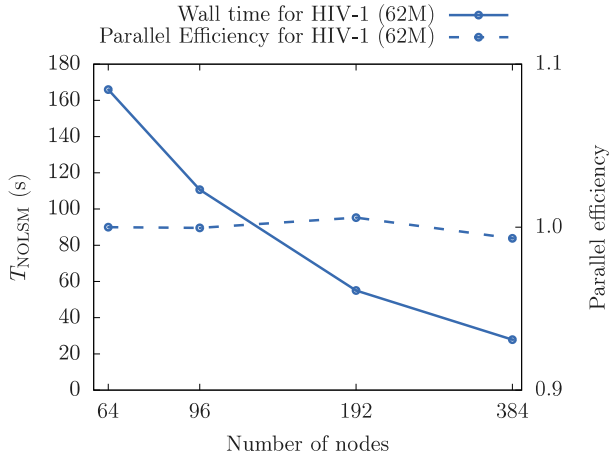

Fig. 9. Strong scaling behavior of the NOLSM method for HIV-1 capsid with ≈ 62 million atoms.

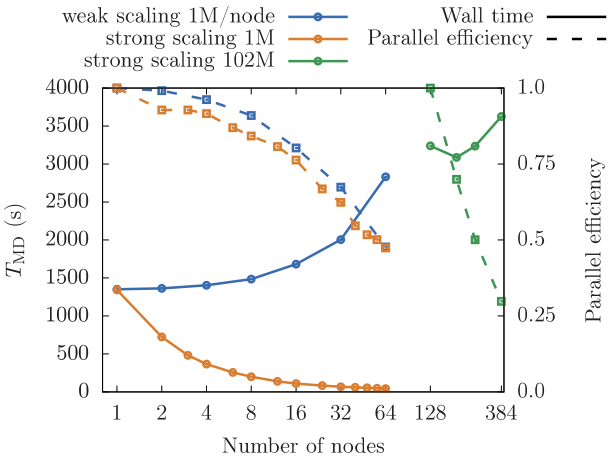

Fig. 10. Strong and weak scaling behavior of a full AIMD step for water: weak scaling for 1,022,208 atoms per node (1M) and strong scaling for 1,022,208 atoms (1M) as well as 101,875,968 atoms (102M).

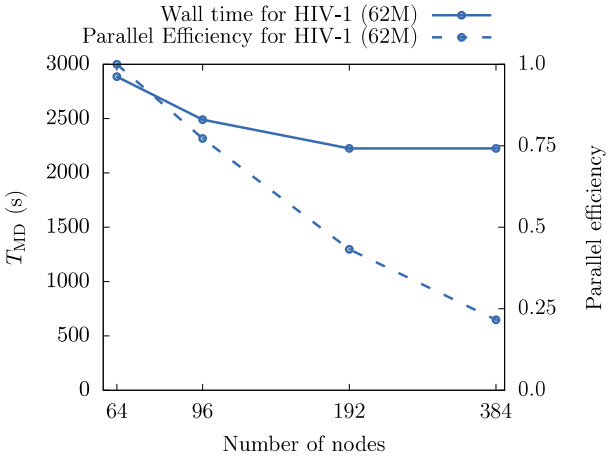

Fig. 11. Strong scaling behavior of a full AIMD step for the HIV-1 capsid with ≈ 62 million atoms.

5.2. Electronic structure-based molecular dynamics

As shown above, the NOLSM method drastically accelerates the computation of the electronic ground state, which is the core computational routine of an electronic structure-based AIMD calculation. In all previous approaches, this part by far dominated the overall runtime and has thus been a focus of this work.

Fig. 10 shows the resulting strong and weak scaling results for water for a complete electronic structure-based AIMD step. The overall scaling of a complete time step suggests that additional effort should be invested in the force evaluation. In total, the use of the NOLSM method has enabled the calculation of an electronic structure-based AIMD step for a system containing more than 100 million atoms in under one hour for the first time. Fig. 11 shows the corresponding results for a full AIMD step on the HIV-1 capsid. The total wall time for a complete time step is well below one hour.

6. Implications

The methods developed in this work have far-reaching implications for both scientific applications and extreme-scale simulation methods on modern computer architectures with accelerators.

From the application point of view, our technique allows for performing electronic structure-based simulations for systems with more than 100 million atoms at quantum mechanical accuracy. This contribution allows the application of *ab-initio* simulation methods in life-sciences, where simulation of very large molecules or long time scales are required that are typically only accessible to classical methods. Since our new developments and code improvements will be contributed to the official CP2K code, the broad atomistic-simulation community will be able to directly profit from this work.

On the methodological level, our work can be applied and generalized to a large variety of atomistic simulation and computational science problems. A natural extension is to apply the presented methods to DFT because in CP2K the required construction of the Kohn–Sham Hamiltonian also scales nearly linearly [36]. Further, our method to compute exact ensemble averages for observables despite truncation errors in the force computation is neither restricted to AIMD, nor errors introduced by low-precision arithmetic. On the contrary, our approach of using a modified Langevin-type equation is directly applicable to the complete field of MD simulations, in particular for classical methods, and can be used to compensate other forms of approximations that can be modeled as noise, for example, time-step errors, mixed-precision arithmetic or FFT-based Ewald summation for periodic structures [57].

Finally, the submatrix method we have used to compute the matrix sign function in this work is also applicable to other matrix functions, for example, arbitrary polynomials, roots, etc. The precondition is that the sparsity pattern of the initial matrix is approximately preserved under the matrix function. The submatrix method has two main advantages for extreme-scale computing applications: First, it decomposes the problem of computing matrix functions to make it highly parallel while only requiring very little communication. Second, the conversion of the problem from large sparse distributed matrices to much smaller dense local matrices is favorable for GPUs and other accelerators that are optimized for dense matrix algebra. Hence, it is possible to apply the submatrix method in many cases where the core computational problem is a linear eigenproblem that can be reformulated as a matrix function. Examples include the solution of Maxwell's equations in the frequency domain for electrodynamics simulations by casting the problem as a linear eigenproblem [64].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer JUWELS Booster at Jülich Supercomputing Centre (JSC). Additionally, we would like to thank for funding of this project by computing time provided by the Paderborn Center for Parallel Computing (PC^2), as well as the Federal Ministry of Education and Research (BMBF) and the state of North Rhine-Westphalia as part of the NHR Program. T.D.K. received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant Agreement No. 716142). T.D.K. and C.P. kindly acknowledge funding from Paderborn University's research award for "GreenIT". Finally, we thank Thomas Müller (JSC), Paul F. Baumeister (JSC), and Markus Hrywniak (NVIDIA) for valuable discussions.

References

[1] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, Equation of state calculations by fast computing machines, J. Chem. Phys. 21 (6) (1953) 1087.

[2] A. Rahman, Correlations in the motion of atoms in liquid argon, Phys. Rev. 136 (2A) (1964) A405–A411.

[3] A.K. Rappe, C.J. Casewit, K.S. Colwell, W.A. Goddard III, W.M. Skiff, UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations, J. Am. Chem. Soc. 114 (25) (1992) 10024–10035.

[4] A.D. MacKerell Jr., et al., All-atom empirical potential for molecular modeling and dynamics studies of proteins, J. Phys. Chem. B 102 (18) (1998) 3586–3616.

[5] A.P. Bartok, et al., Machine learning unifies the modeling of materials and molecules, Sci. Adv. 3 (12) (2017) e1701816.

[6] J.A. Keith, et al., Combining machine learning and computational chemistry for predictive insights into chemical systems, arXiv:2102.06321 [physics.chem-ph].

[7] R. Car, M. Parrinello, Unified approach for molecular dynamics and density-functional theory, Phys. Rev. Lett. 55 (22) (1985) 2471–2474.

[8] M.C. Payne, M.P. Teter, D.C. Allan, T.A. Arias, J.D. Joannopoulos, Iterative minimization techniques for ab initio total-energy calculations: Molecular dynamics and conjugate gradients, Rev. Modern Phys. 64 (4) (1992) 1045–1097.

[9] T.D. Kühne, M. Krack, F.R. Mohamed, M. Parrinello, Efficient and accurate Car-Parrinello-like approach to born-oppenheimer molecular dynamics, Phys. Rev. Lett. 98 (6) (2007) 066401.

[10] T.D. Kühne, Second generation Car–Parrinello molecular dynamics, WIREs Comput. Mol. Sci. 4 (2014) 391–406.

[11] S. Goedecker, Linear scaling electronic structure methods, Rev. Modern Phys. 71 (4) (1999) 1085–1123.

[12] W. Yang, Direct calculation of electron density in density-functional theory, Phys. Rev. Lett. 66 (11) (1991) 1438–1441.

[13] G. Galli, M. Parrinello, Large scale electronic structure calculations, Phys. Rev. Lett. 69 (24) (1992) 3547–3550.

[14] D. Richters, T.D. Kühne, Self-consistent field theory based molecular dynamics with linear system-size scaling, J. Chem. Phys. 140 (13) (2014) 134109.

[15] E. Prodan, W. Kohn, Nearsightedness of electronic matter, Proc. Natl. Acad. Sci. USA 102 (33) (2005) 11635–11638.

[16] T.D. Kühne, J. Heske, E. Prodan, Disordered crystals from first principles II: Transport coefficients, Ann. Physics 421 (2020) 168290.

[17] P. Klavik, A. Malossi, C. Bekas, A. Curioni, Changing computing paradigms towards power efficiency, Philos. Trans. R. Soc. A Math. Phys. Eng. Sci. 39 (2018) (2014) 372.

[18] C. Plessl, M. Platzner, P.J. Schreier, Approximate computing, Inform. Spektrum 15 (2015) 396–399.

[19] K. Karhan, R.Z. Khaliullin, T.D. Kühne, On the role of interfacial hydrogen bonds in "on-water" catalysis, J. Chem. Phys. 141 (22) (2014) 12B632_1.

[20] V. Rengaraj, M. Lass, C. Plessl, T.D. Kühne, Accurate sampling with noisy forces from approximate computing, Computation 8 (39) (2020) 1–11.

[21] J. VandeVondele, U. Borstnik, J. Hutter, Linear scaling self-consistent field calculations with millions of atoms in the condensed phase, J. Chem. Theory Comput. 8 (10) (2012) 3565–3573.

[22] K. Nemeth, G.E. Scuseria, Linear scaling density matrix search based on sign matrices, J. Chem. Phys. 113 (2000) 6035–6041.

[23] M. Lass, S. Mohr, H. Wiebeler, T. Kuhne, C. Plessl, A massively parallel algorithm for the approximate calculation of inverse p-th roots of large sparse matrices, in: Proc. Platform for Advanced Scientific Computing (PASC) Conference, ACM, New York, NY, USA, 2018.

[24] M. Lass, R. Schade, T. Kühne, C. Plessl, A submatrix-based method for approximate matrix function evaluation in the quantum chemistry code CP2K, in: Proc. International Conference for High Performance Computing, Networking, Storage and Analysis (SC), IEEE Computer Society, Los Alamitos, CA, USA, 2020, pp. 1127–1140.

[25] T.D. Kühne, E. Prodan, Disordered crystals from first principles I: Quantifying the configuration space, Ann. Physics 391 (2018) 120–149.

[26] J. Hutter, A. Curioni, Car–Parrinello molecular dynamics on massively parallel computers, ChemPhysChem 6 (9) (2005) 1788–1793.

[27] F. Gygi, Architecture of Qbox: A scalable first-principles molecular dynamics code, IBM J. Res. Dev. 52 (1.2) (2008) 137–144.

[28] Z. Zhao, et al., The linearly scaling 3D fragment method for large scale electronic structure calculations, J. Phys. Conf. Ser. 180 (1) (2009) 012079.

[29] N. Jain, et al., Openatom: Scalable ab-initio molecular dynamics with diverse capabilities, in: International Conference on High Performance Computing, Springer, 2016, pp. 139–158.

[30] Y. Hasegawa, et al., Performance evaluation of ultra-large-scale first-principles electronic structure calculation code on the K computer, J. High Perform. Comput. Appl. 28 (3) (2014) 335–355.

[31] P. Motamarri, et al., DFT-FE–A massively parallel adaptive finite-element code for large-scale density functional theory calculations, Comput. Phys. Comm. 246 (2020) 106853.

[32] K.-i. Nomura, et al., Metascalable quantum molecular dynamics simulations of hydrogen-on-demand, in: SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2014, pp. 661–673.

[33] J.C. Prentice, et al., The ONETEP linear-scaling density functional theory program, J. Chem. Phys. 152 (17) (2020) 174111.

[34] J.-L. Fattebert, D. Osei-Kuffuor, E.W. Draeger, T. Ogitsu, W.D. Krauss, Modeling dilute solutions using first-principles molecular dynamics: Computing more than a million atoms with over a million cores, in: SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2016, pp. 12–22.

[35] A. Nakata, et al., Large scale and linear scaling DFT with the CONQUEST code, J. Chem. Phys. 152 (16) (2020) 164112.

[36] T. Kühne, et al., CP2K: An electronic structure and molecular dynamics software package - Quickstep: Efficient and accurate electronic structure calculations, J. Chem. Phys. 152 (19) (2020) 194103.

[37] J. Hutter, A. Curioni, Dual-level parallelism for ab initio molecular dynamics: Reaching teraflop performance with the CPMD code, Parallel Comput. 31 (1) (2005) 1–17.

[38] F. Gygi, et al., Large-scale electronic structure calculations of high-Z metals on the bluegene/l platform, in: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, 2006, p. 45.

[39] D.R. Bowler, T. Miyazaki, Calculations for millions of atoms with density functional theory: Linear scaling shows its potential, J. Phys.: Condens. Matter 22 (7) (2010) 074207.

[40] J. VandeVondele, U. Borstnik, J. Hutter, Linear scaling self-consistent field calculations with millions of atoms in the condensed phase, J. Chem. Theory Comput. 8 (10) (2012) 3565–3573.

[41] K.A. Wilkinson, N.D. Hine, C.-K. Skylaris, Hybrid MPI-OpenMP parallelism in the ONETEP linear-scaling electronic structure code: Application to the delamination of cellulose nanofibrils, J. Chem. Theory Comput. 10 (11) (2014) 4782–4794.

[42] M. Arita, S. Arapan, D.R. Bowler, T. Miyazaki, Large-scale DFT simulations with a linear-scaling DFT code CONQUEST on K-computer, J. Adv. Simul. Sci. Eng. 1 (1) (2014) 87–97.

[43] S. Andermatt, J. Cha, F. Schiffmann, J. VandeVondele, Combining linear-scaling DFT with subsystem DFT in Born–Oppenheimer and Ehrenfest molecular dynamics simulations: From molecules to a virus in solution, J. Chem. Theory Comput. 12 (7) (2016) 3214–3227.

[44] S. Das, P. Motamarri, V. Gavini, B. Turcksin, Y.W. Li, B. Leback, Fast, scalable and accurate finite-element based ab initio calculations using mixed precision computing: 46 PFLOPS simulation of a metallic dislocation system, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2019, pp. 1–11.

[45] S. Grimme, C. Bannwarth, P. Shushkov, A robust and accurate tight-binding quantum chemical method for structures, vibrational frequencies, and noncovalent interactions of large molecular systems parametrized for all spd-block elements (Z=1–86), J. Chem. Theory Comput. 13 (5) (2017) 1989–2009.

[46] T.D. Kühne, M. Krack, M. Parrinello, Static and dynamical properties of liquid water from first principles by a novel Car-Parrinello-like approach, J. Chem. Theory Comput. 5 (2009) 235–241.

[47] R. McWeeny, Some recent advances in density matrix theory, Rev. Modern Phys. 32 (2) (1960) 335.

[48] P. Pulay, Ab initio calculation of force constants and equilibrium geometries in polyatomic molecules: I. Theory, Mol. Phys. 17 (2) (1969) 197–204.

[49] G. Schulz, Iterative berechung der reziproken matrix, J. Appl. Math. Mech. (ZAMM Z. Angew. Math. Mech.) 13 (1) (1933) 57–59.

[50] C. Kenney, A.J. Laub, Rational iterative methods for the matrix sign function, SIAM J. Matrix Anal. Appl. 12 (2) (1991) 273–291.

[51] N.J. Higham, Stable iterations for the matrix square root, Numer. Algorithms 15 (2) (1997) 227–242.

[52] D. Richters, M. Lass, A. Walther, C. Plessl, T. Kühne, A general algorithm to calculate the inverse principal p-th root of symmetric positive definite matrices, Commun. Comput. Phys. 25 (2) (2019) 564–585.

[53] U. Borštnik, J. VandeVondele, V. Weber, J. Hutter, Sparse matrix multiplication: The distributed block-compressed sparse row library, Parallel Comput. 40 (5–6) (2014) 47–58.

[54] NVIDIA Corporation, CUDA C++ Programming Guide, NVIDIA Corporation, 2021, [Online]. Available: https://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf.

[55] [Online]. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf.

[56] S. Grimme, S. Ehrlich, L. Goerigk, Effect of the damping function in dispersion corrected density functional theory, J. Comput. Chem. 32 (7) (2011) 1456–1465.

[57] U. Essmann, et al., A smooth particle mesh Ewald method, J. Chem. Phys. 103 (19) (1995) 8577–8593.

[58] A. Ricci, G. Ciccotti, Algorithms for Brownian dynamics, Mol. Phys. 101 (12) (2003) 1927–1931.

[59] [Online]. Available: https://github.com/cp2k/cp2k/blob/028e7b8381f1bc85b52fb82ab205a43ab6f0c339/benchmarks/QS_DM_LS/H2O-dft-ls.inp.

[60] G. Zhao, et al., Mature HIV-1 capsid structure by cryo-electron microscopy and all-atom molecular dynamics, Nature 497 (7451) (2013) 643–646.

[61] R. Schade, et al., Enabling electronic structure-based ab-initio molecular dynamics simulations with hundreds of millions of atoms, 2021, http://dx.doi.org/10.5281/zenodo.4692508, Zenodo.

[62] Hardware configuration of the JUWELS booster module, [Online]. Available: https://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUWELS/Configuration/Configuration_node.html.

[63] JUWELS booster TOP 500 entry, [Online]. Available: https://www.top500.org/system/179894/.

[64] S.G. Johnson, J.D. Joannopoulos, Block-iterative frequency-domain methods for Maxwell's equations in a planewave basis, Opt. Express 8 (3) (2001) 173–190.