Title

From NWChem to NWChemEx: Evolving with the Computational Chemistry Landscape

Permalink

https://escholarship.org/uc/item/4sm897jh

Journal

Chemical Reviews, 121(8)

ISSN

0009-2665

Authors

Kowalski, Karol
Bair, Raymond
Bauman, Nicholas P
et al.

Publication Date

2021-04-28

DOI

10.1021/acs.chemrev.0c00998

Peer reviewed

From NWChem to NWChemEx: Evolving with the computational chemistry landscape

Karol Kowalski,[†] Raymond Bair,[‡] Nicholas P. Bauman,[†] Jeffery S. Boschen,[¶] Eric J. Bylaska,[†] Jeff Daily,[†] Wibe A. de Jong,[§] Thom Dunning, Jr,[†] Niranjan Govind,[†] Robert J. Harrison,[‖] Murat Keçeli,[‡] Kristopher Keipert,[⊥] Sriram Krishnamoorthy,[†] Suraj Kumar,[†] Erdal Mutlu,[†] Bruce Palmer,[†] Ajay Panyala,[†] Bo Peng,[†] Ryan M. Richard,[¶] T. P. Straatsma,[#] Peter Sushko,[†] Edward F. Valeev,[@] Marat Valiev,[†] Hubertus J. J. van Dam,[△] Jonathan M. Waldrop,[¶] David B. Williams-Young,[§] Chao Yang,[§] Marcin Zalewski,[†] and Theresa L. Windus[*,∇]

†*Pacific Northwest National Laboratory, Richland, WA 99352*

‡*Argonne National Laboratory, Lemont, IL 60439*

¶*Ames Laboratory, Ames, IA 50011*

§*Lawrence Berkeley National Laboratory, Berkeley, 94720*

‖*Institute for Advanced Computational Science, Stony Brook University, Stony Brook, NY 11794*

⊥*NVIDIA Inc, previously Argonne National Laboratory, Lemont, IL 60439*

#*National Center for Computational Sciences, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6373*

@*Department of Chemistry, Virginia Tech, Blacksburg, VA 24061*

△*Brookhaven National Laboratory, Upton, NY 11973*

∇*Department of Chemistry, Iowa State University and Ames Laboratory, Ames, IA 50011*

E-mail: twindus@iastate.edu

Abstract

Since the advent of the first computers, chemists have been at the forefront of using computers to understand and solve complex chemical problems. As the hardware and software have evolved, so have the theoretical and computational chemistry methods and algorithms. Parallel computers clearly changed the common computing paradigm in the late 1970s and 80s, and the field has again seen a paradigm shift with the advent of graphical processing units. This review explores the challenges and some of the solutions in transforming software from the terascale to the petascale and now to the upcoming exascale computers. While discussing the field in general, NWChem and its redesign, NWChemEx, will be highlighted as one of the early co-design projects to take advantage of massively parallel computers and emerging software standards to enable large scientific challenges to be tackled.

Contents

1 Introduction

In 2015, the government of the United States of America launched the National Strategic Computing Initiative (NSCI)[1] to take advantage of and advance the nation's use of high performance computing (HPC), especially toward exascale computing. In particular, this multi-agency initiative invested significant funds in one of the largest efforts to date to enable the scientific computational community to advance hardware and software efforts in HPC. As part of this launch, the Department of Energy (DOE) initiated the Exascale Computing Project (ECP)[2,3] that is promoting a very forward-looking approach to software development — the ECP is ultimately concerned with software for the first generation of exascale systems and in laying the foundation for the new era of computational science over the coming decade(s). As part of the NSCI, the ECP is jointly funded by both the DOE Office of Science (SC) and the DOE National Nuclear Security Administration (NNSA). The ECP's goal is to accelerate the delivery of an exascale capable computing ecosystem that delivers 50 times more computational science and data analytic application power than is available on current DOE architectures to address challenges in scientific discovery, energy assurance, economic competitiveness, and national security.

ECP has three main focus areas: (1) Application Development to deliver the science-based applications, (2) Software Technology to deliver a comprehensive and coherent software stack to enable productive highly parallel application development that can portably target diverse exascale architectures, and (3) Hardware and Integration that supports vendor and lab hardware research and development activities. In addition to having a very broad participation in the ECP from multiple labs and academic partners, there is a wide representation of computational scientists, mathematicians, and computer scientists. This leads to a very exciting environment for the development of new software for the exascale era.

Fortunately, NWChem[4–11], along with GAMESS[12], LAMMPS[13], and QMCPACK[14], was selected to be part of the ECP effort. After extensive consideration, the NWChem team decided to design and implement a new software program, NWChemEx, to realize the full scientific potential of exascale. While NWChem was based on Fortran, NWChemEx will be based on C++ and Python. While NWChem was based mostly on conventional electronic structure theory (EST) algorithms, NWChemEx will be based on modern reduced-scaling methods. In addition, new software engineering techniques will be used to facilitate the development of NWChemEx by a broad community and integration of software from other parts of the ECP will be adopted, as appropriate, to facilitate portability and performance of the software.

As with most, if not all, chemistry codes, the software development is motivated by particular types of science goals. The development of NWChemEx is driven by a decadal computational science challenge important to the mission of the DOE – the development of advanced biofuels, the national need for which has both energy security and climate change considerations. The goal of DOE's advanced biofuels program is to develop fuels that can use the existing fueling infrastructure and replace existing fuels on a gallon-for-gallon basis. However, producing biofuels with these characteristics in a sustainable and economically competitive way is technically challenging, especially in a changing global climate.[15]

NWChemEx is being developed to address two interrelated science challenges in the advanced biofuels program: (*i*) development of a molecular understanding of proton controlled membrane transport processes driving stress responses in plants suitable for biomass production, and (*ii*) development of catalysts for the efficient conversion of biomass-derived intermediates into biofuels, hydrogen, and other bioproducts. *Both of these problems involve multi-faceted chemical transformations that occur in complex and dynamic molecular environments, where a complete physical description of both the active site and its environment is essential for achieving the needed predictive capability.* These science challenges are closely related and require a computational chemistry capability that can take full advantage of exascale computing systems. Truly predictive modeling of both of these molecular systems requires use of high-level quantum mechanical methods describing $O(10^3)$ atoms with coupled-cluster (CC) methods embedded in an environment of $O(10^5)$ atoms described by density functional theory (DFT) as well as the inclusion of thermal effects by using dynamical and/or statistical methods to sample the potential energy surface. In combination with exascale computers, NWChemEx will provide a major advance in our ability to calculate the structures and energetics of the very large molecules that represent the active sites in these decadal challenges.

This paper discusses challenges and some of the solutions toward moving software to the exascale. While the examplar will be the NWChem and NWChemEx software, this development will be placed in the broader context of the computational chemistry ecosystem. The next Section will describe the hardware and software evolutionary challenges in a chronological manner, providing context for the different computational chemistry software programs. After that, the design principles will be discussed to show the complexity involved in the hardware and software changes and the difficulties in designing for future architectures. This will be followed by a discussion of the general evolution of software engineering in the computational chemistry community. Following this is a discussion on various middle-ware tools that the computational chemistry community has come to rely on: tools and runtime environments, tensor methods, and mathematical solvers. The subsequent Sections will discuss the general progress made in the field for the various workhorse methods of chemistry, especially those that are a focus of NWChemEx: Gaussian basis Hartree–Fock (HF) and DFT, plane-wave DFT, CC, classical molecular dynamics (MD), and embedding methods. Since each of these methods could easily be a review paper on their own, existing reviews will be given for reference, and only salient points concerning HPC will be given. Also, other methods such as those to address strong correlation and molecular properties will not be included in this review. Finally, a conclusions and future directions Section will wrap up the paper.

2 Hardware and Software Evolution Challenges

NWChem was born in an era of extensive change in computing hardware. In the early 1990s, we were exiting the era of vector supercomputers, super minicomputers, and parallel mainframes. The dominant quantum chemistry approaches were based on HF, many-body perturbation theory, configuration interaction, generalized valence bond[16], and multi-configuration self-consistent field (MCSCF) and were implemented in various electronic structure programs such as Columbus[17], GAMESS[12], GAMESS UK[18], Hondo[19], Gaussian[20], Molcas[21], MOLPRO[22], and Turbomole[23]. Coupled-cluster methods were gaining traction (e.g., ACES[24]). Density functional methods were in their infancy in codes such as ADF[25], CASTEP[26], CPMD[27], deMon[28], and VASP[29]. Multiple molecular mechanics and MD codes were also in extensive use at the time, including Amber[30], Charmm[31], and Tinker[32]. Most, if not all, of these codes evolved from serial ancestors, some with extensive rework for vector computers and more modest efforts made to achieve small scale parallelism. (The authors have tried to list the major software packages of the time, but it is quite likely that some have been missed. The reader is referred to the Molecular Sciences Software Institute (MolSSI) Software Database[33] and the Wikipedia Computational Chemistry Software page[34].)

In the late 1980s a new kind of processor had emerged, Reduced Instruction Set Computer (RISC). RISC processors were fast and cheap, and people quickly envisioned supercomputers built from massive arrays of RISC processors. The problem was that none of the quantum codes was primed for this transition. For fast RISC code, cache reuse is paramount, which called for substantial changes to vector codes, e.g., inverting and blocking loops. In addition, none of the codes were designed to achieve the scaling efficiencies needed for large-scale parallelism for the emerging supercomputers with hundreds (and conceivably millions) of processor cores. It became clear that a new code was needed, designed expressly for massively parallel computers and RISC processors. This became the premise of NWChem. Several other codes were developed around this time – some to take advantage of the architecture changes and others based on novel algorithms. These codes include CP2K[35], Desmond[36], GROMACS[37], LAMMPS[13], Massively Parallel Quantum Chemistry (MPQC)[38], Q-Chem[39], and SIESTA[40].

NWChem was originally designed and implemented as part of the initiation of the construction project associated with the Environmental Molecular Sciences Laboratory (EMSL) at Pacific Northwest National Laboratory (PNNL) in 1992. At the time, the development of software before the hardware was available was a very forward way of thinking about software development. Of course, this raised challenges for the software developers, such as prognosticating about future architectures and how to obtain high performance on these architectures while maintaining programmer productivity. Overcoming these challenges led to a code design that sought for flexibility and extensibility, as well as high-level interfaces to functionality that hid some of the hardware issues from the chemistry software developer. Over the years, this design and implementation has successfully advanced multiple science agendas, and NWChem has grown to be an over two-million line, high-performance, scalable software code with advanced scientific capabilities that are used throughout the molecular sciences community. After a successful quarter of a century, the NWChem software design now faces significant challenges (discussed below) to obtain high performance with the projected exascale computers.

Much has changed about computer hardware and scientific software architecture since NWChem was first conceived, around 1990. The change in computers has been dramatic, across the board. In the early 1990s, it was becoming clear that the era of the large, shared-memory vector systems was over, and that arrays of simpler, distributed-memory processors would become the workhorses of NWChem. Commodity processors, versus custom vector processors, drove down costs as did eliminating expensive shared-memory capabilities. Furthermore, adding nodes systematically increased processing power, memory capacity, and bandwidth. Early distributed-memory computers (e.g., Intel ipsc 286, 386, Paragon and i860; FPS T-series; nCUBE; etc.) all had single-processor nodes with limited memory, and some had local disks and/or shared parallel file systems. In 1996/7 EMSL/PNNL installed what was then the largest IBM RS/6000 SP cluster, and this machine, along with the Intel Touchstone Delta, drove much of the early design of NWChem.

The most powerful computers became proprietary networks of serial (single-core) processors. Processor architectures dictated careful use of cache memory, and blocking operations like matrix multiply were critical to good performance. Compilers and software tools provided little assistance, with Fortran being the choice for generating optimized code. These distributed-memory systems required explicit data transport. Most codes used Message Passing Interface (MPI) and its send-receive paradigm, but NWChem adopted Global Arrays (GA)[41], where one-sided remote data access was better suited to quantum chemistry methods. GA also helped to mitigate the relatively high network latencies. A "massively parallel" computer of the early 1990s had dozens to hundreds of nodes, and hence dozens to hundreds of threads of execution. When necessary, NWChem design trade-offs were made in favor of good performance for large computations on large computers.

The first decade of the 21st century saw the growth of "massively parallel" high-performance computers toward clusters consisting of tens of thousands of nodes, connected via high-performance networks with various levels of connectivity. During this decade, we were also introduced to novel processor architectures that challenged the programming paradigm of NWChem that was not designed with large numbers of execution threads in mind. The energy-efficient but low clock speed BlueGene[42] and Cell[43] chips were introduced by IBM and powered some of the TOP-500 supercomputers. Efficiently programming these chips required significant rewriting of chemistry codes to take advantage of the large number of execution threads and memory hierarchy. Within the realm of chemistry, IBM's BlueGene processors have found some success with molecular dynamics[44] and plane-wave DFT[45]. Later in the decade, Intel introduced a Cell inspired many-core Xeon Phi[46] processor architecture with >80 integrated cores. NWChem and other chemistry codes were able to successfully adapted to use the OpenMP programming model[47] and take advantage of Intel's Xeon Phi[48–53]. Another version of the many-core computing paradigm arrived with the adoption of Graphical Processing Units (GPUs) in general-purpose supercomputers. Programming GPUs to efficiently run chemistry codes is not trivial. GPUs tend to organize cores into warps that act much like vector units. High throughput is achieved by operating many warps simultaneously. For example, the current V100 GPUs group 32 cores into a warp, and there are 160 warps to a GPU[54,55]. Various existing computational chemistry codes have been adapted[53,56–67] or designed from the outset (e.g., TeraChem)[67,68] to use GPUs.

The current decade has seen accelerating rates of change in computer hardware architecture, driving new code architecture at every level. The design of NWChemEx must efficiently use a much more complex set of exascale computer capabilities. GPUs are currently the most widely used hardware accelerators in large supercomputers because of their lower power requirements and high compute capability. GPU developments have reached a stage where nodes may have multiple GPUs per node – as many as 6 GPUs per node on current leadership class computers and more projected for the future – with special communication links between them. Unfortunately, due to the diversity of GPU hardware and vendors, no ubiquitous GPU code development tool has emerged. For NWChemEx, we primarily develop code using CUDA,[69] OpenMP, and OpenCL (and derivatives),[70] with different languages needed for different leadership computers. We also count on conversion tools to enable the transformation of code to software such as HIP[71] that is used for the AMD GPUs. We are studying the potential roles of more performance-portable tools like Kokkos[72] and RAJA[73], but so far, none addresses all our needs.

At the processor level, we now have large numbers of multi-threaded cores in a node, each vying for a share of cache, memory, and network bandwidth. Memory architectures are more complex, with additional levels of cache (often three in all), fast in-package DRAM memory (e.g., HBM), plus DDR DRAM, and perhaps storage class memory on the network fabric. Orchestrating efficient movement of data through this NUMA (non-uniform memory access) hierarchy is the predominant challenge in software design. However, our arsenal of software development tools is much better. The expressiveness of languages such as C++ and Python now suits chemistry well, allowing us to create high-level, composable capabilities with efficient implementations underneath. More recent codes such as Bagel[74], HOOMD[75], NAMD[76], OpenMM[77], Orca[78], PLUMED[79], Psi4[80], PySCF[81], Qbox[82] and Quantum Espresso[83] take advantage of these languages to build software that is flexible for users and developers and still performant. On the other hand, the more complex processor architectures increase the complexity of the compiler's job, and the promise of super-smart compiler code optimization remains a dream.

Within each core, the arithmetic units are ganged into SIMD units to enable fast and orderly operations. In their first incarnation, these vector units tended to operate on long registers holding 64 double-precision floating-point numbers, such as in the Cray-1 computer. In more recent years, vector units are making a comeback but the units tend to operate on shorter vectors of up to 8 double-precision floating-point numbers, such as in the AVX-512 instruction set. These shorter vector units suffer less from branch divergences that could severely impact the performance of early vector machines.

In addition to the increases in cores, modern hardware architectures offer further opportunities for increased performance by allowing floating-point units to be used in multiple ways. For example, a vector unit may support the execution of 8 double-precision floating-point operations simultaneously, alternatively it can execute 16 single-precision floating-point operations in the same time. Hence, there is a direct path for trading precision for speed. This provides a significant incentive for developing mixed-precision codes that perform as much computation as possible at low precision and as little as possible at high precision.

Simultaneously these codes need to ensure that the final computational results still meet the requirements of the science domain. In the latest accelerators, mixed-precision strategies have been taken even further by providing half-precision hardware. The latter is mainly in response to the ongoing growth of machine learning applications.

Filling up an exascale computer with a large NWChemEx computation will require coordination of perhaps a billion concurrent threads of execution. At the same time, it requires managing multiple complex memory hierarchies, multiple processor architectures, and multiple instances of each component while coordinating and combining work done at multiple precisions. Addressing this level of complexity effectively remains a challenge.

3 Design Challenges and Principles

Software design has become an increasingly important part of developing and sustaining any large software stack – even for small software projects, especially if it is to be a reusable library or component in a larger software ecosystem. While software design did not always garner the attention that it deserved, most modern computational chemistry codes have some design criteria or objectives and programming guidelines. However, even if the design criteria are not explicit, each computational chemistry code has at least implicit assumptions about the software. For example, some of the design choices include various levels of parallelism, minimal memory usage, amount of disk usage, ease of composition, and access to data. Each of these design choices will affect how the software is developed and how easy it is to maintain that software.

With the emergence of exascale computers imminent[84], there are many factors a new computational chemistry software development effort needs to consider in order to fully exploit these platforms:

- Increased scale of parallelism — Exascale will require developers to exploit over a 1000-fold more parallelism compared to the petascale platforms of today. This will require a much finer-grain parallelism to ensure parallel scalability;

- Greater heterogeneity — Exascale computers will consist of a mix of multi-core, many-core and hybrid processors (such as GPUs, FPGAs), each with very different performance characteristics. Optimized code will be needed for each computational chemistry algorithm on each type of processor. One pathway to maximize programmer productivity is to employ extensive automated code generation;

- Floating-point operations per second (flops) are now nearly free — We now need the balance of recomputing data with storing and moving data. To get scaling and performance, the focus has to be on designing algorithms that exploit the memory-hierarchy and data movements with maximum efficiency;

- Complex memory-hierarchy — Exascale computers will have a much deeper and more heterogeneous memory-hierarchy. Algorithmic design choices will have to be made to minimize memory induced latencies.

- Limited ability to hide latency — Data movement will have to be as asynchronous as possible. As such, memory models need to be an integral part of the execution model, and data/computation locality is essential[85];

- Data abstraction — In contrast to dense algorithms, fast/sparse algorithms and applications such as reduced scaling DFT and coupled cluster methods, or molecular dynamics will have to be designed with more complex data structures and finer granularity;

- Persistent data — Persistence of information such as wave functions is key in computational chemistry code. An explicit tie to external storage can make it hard to compose/nest calculations (e.g., perform a parameter sweep in parallel). Petascale and exascale platforms will have intermediate storage facilities using non-volatile random-access memory (NVRAM) or solid-state disks (SSD);

- Resilience, energy, and power — These exascale concerns[86–89] are rarely represented in the toolchains of today's computational chemistry codes.

In the exascale era, as we seek performance portability to future machines while maintaining scientific productivity, much greater separation of concerns is needed. The Tensor Contraction Engine (TCE), Tensor Algebra for Many-body Methods (TAMM), and TiledArray (TA), which will be discussed in section 6, are complementary approaches to turning high-level statements of many-body physics into executable code.

The resulting computation then needs to be expressed as tasks managed by an intelligent runtime that understands the algorithmic and data dependencies as well as the machine resources and memory/communication sub-systems. Such an approach facilitates over-decomposition to enhance load balance and tolerate latency (whether algorithmic or from hardware/software) as well as approaches to automating resilience. One such runtime is Charm++, which has been used by codes such as NAMD[90] and OpenAtom[91], and will be discussed in section 5.

The irregularity of electronic structure computations (e.g., arising from sparsity, symmetry, and widely varying costs of evaluating integral shell quartets)[92] make it hard to use just message passing to write fully-distributed chemistry applications with good parallel load balance. Alternatively, one can use one-sided message passing[93,94]. In this approach, a process that needs to access remote data or perform a remote operation can send a message directly to a remote procedure that is invoked without the explicit participation of the remote process. However, there were (and still are) significant performance and correctness concerns with this approach, and it was not portable to shared-memory machines and many distributed-memory machines. Instead, remote memory access (i.e., a distributed shared-memory programming model) was identified to more closely match the capabilities of both distributed computer networks and shared-memory computers.

Many algorithms were developed to be distribution agnostic using a simple NUMA performance model[95]. Under the ideal assumption that there are no communication or computational hot spots, the execution time is simply the perfectly parallel computation time plus an overhead that is the amount of data being communicated per processor divided by the effective bandwidth. For many electronic structure algorithms, the volume of computation grows super-linearly with the volume of data; thus it is in principle possible to increase the local problem size and have the computation dominate (e.g., matrix multiplication). If the communication intensity is low, the effective bandwidth approaches the link speed of the network; however, with intense communication it degrades to the bisection bandwidth divided by the number of processors (P). For a hypercube, this average worst-case bandwidth is half the link speed, but for a 2-D mesh, it is $O\left(P^{-1/2}\right)$. Thus, topology plays a very important role in scalable algorithms. The assumptions of no hot spots/links can be at least partially realized by over-decomposition and randomization.

NWChemEx applies many of the design principles discussed with a focus on performance and flexibility. The two principles are interconnected since typical performance modifications, *e.g.*, architecture-specific code, or new algorithms, tends to require the code to be refactored. The easiest way to achieve the desired flexibility is by ensuring the software infrastructure is as decoupled as possible. This, in turn, ensures that if an algorithm is modified, it does not affect the surrounding code. Within the NWChemEx project, the decision was made to achieve this decoupling through the use of "modules". Following typical object-oriented programming (OOP) conventions, each module is a self-contained, opaque, callable object adhering to one of many possible standardized application programming interfaces (APIs). Each standardized API is associated with a particular property — *e.g.*, an energy derivative, the Fock matrix, or a molecular orbital (MO) space — called its "property type". The property type APIs are solidified *via* a series of corresponding abstract base classes. When the package needs to compute a property, it does so through the abstract base class, thus allowing any module that inherits from that base class to be used interchangeably at that point. Despite the module system conceptually relying on a relatively basic OOP design, the actual implementation is quite a bit more complicated. The additional complexity arises from the fact that NWChemEx must not be restricted to a predefined set of modules (or property types) as it would be if NWChemEx had to instantiate the modules. The module system developed for NWChemEx is the simulation development environment (SDE) framework[96] and will be discussed in more detail in Section 5.

Fault tolerance design fall into three major categories: algorithmic error correction, in-memory redundancy methods, and checkpointing at runtime. Algorithmic error detection and subsequent correction is difficult[97]. In-memory redundancy methods are designed to undercut the performance overhead of traditional disk-based checkpointing methods by storing redundant copies of data in the memory of separate nodes[98]. In the case of a node failure, the backup copy of the node data can be redistributed along with a task load-balancing event if necessary. In-memory redundancy methods have been implemented in NWChem[99] and

Charm++[100]. Lossy compression and differenced/truncated checkpointing techniques[101] can be promising avenues for reducing checkpointing performance overhead. The final component of the fault tolerance design is the runtime, which guides the execution policy and frequency of checkpoints. The current checkpointing runtime in NWChemEx is a relatively spartan placeholder. Each *Cache* holds a handle to a checkpoint archive on disk, and a single parameter that defines the checkpointing frequency for the entire *Cache*. Development efforts are underway to integrate NWChemEx with the VeloC framework[102,103] developed under the ECP. VeloC is a scalable checkpoint/restart runtime with a high-level API that exposes functionality for asynchronous and multi-level checkpointing on complex storage hierarchies. VeloC will provide NWChemEx with a baseline of checkpointing support and performance optimization for a diverse range of hardware architectures.

Performance is a challenging design goal given the complications of multiple accelerator types and languages/compilers associated with them. While no one solution has yet emerged, best practices include localizing these kernels as much as possible to ensure minimal intrusions into the software. Of course, this is not always possible when performance on a particular platform is required. One of the ways to mitigate some of these issues is to rely on a tensor-based framework where possible and ensure that the framework is portable and performant across multiple architectures. Performance is discussed, as appropriate, in each of the following Sections.

4 Software Engineering Practices

Software engineering practices have been slowly making their way into computational chemistry codes. In addition to the design issues discussed above – a significant part of software engineering – other aspects involve the software and developer management life cycle. Early codes usually relied on one person to be the gatekeeper of the software – deciding on the overall architecture of the code (perhaps implicitly), enforcing any existing coding rules, merging changes into the code, adding any appropriate documentation, developing methods for testing the code, performing the tests, preparing any software releases, tracking any bugs, and communicating with users. This meant that while one (or perhaps up to a handful) of people were experts in the whole code, getting new functionality or bug fixes into the software could take time due to the bottleneck of the gatekeeper. This model has significantly changed over the last 10 years, with many of the software codes now using more advanced tools and approaches to their development process. As computational chemistry development teams in their publications tend to focus on the scientific aspects of their efforts rather than the software development aspects it is difficult to extract insights from the literature. As an alternative approach we have looked at the repositories and documentation of a variety of mainly open source computational chemistry packages to distill insights about the tools and processes used. As a first observation, it is clear that the development models have been refined considerably, in the sense that every aspect has become much more fine grained. To illustrate this the evaluation showed a large number of categories including the software license, the revision control technology, the source code hosting, continuous integration (CI) testing and technology, the CI status hosting, code coverage, code coverage hosting, code quality tools, documentation generators, documentation hosting, application distribution, and even application deployment. With this many categories there is considerable variability between codes as to how many and which ones they use. In the following paragraphs our findings are summarized.

The software license is a foundational statement that controls who and under what conditions can access the source code and executables. As a result it also determines who can learn the details of how the software works and suggest or make changes that can be fed back or redistributed. A large number of packages have proprietary or commercial licenses. Perhaps the first such package was Gaussian70[20], released by Pople in 1970 and commercialized since 1987 by Gaussian, Inc. This model has since been followed by other codes including Q-Chem[39], Jaguar[104], MOLPRO[22], ADF[25], Turbomole[23], PQS[105], ORCA[78], GAMESS-UK[18], and TeraChem[106]. Some codes use proprietary licenses that allow use of a code but not redistribution. Examples are GAMESS[12], COLUMBUS[17], and deMon2K[107]. A number of codes have adopted open source licenses. While there are many open sources licenses there are two distinguishing classes. One class of licenses are the copyleft ones and the other class are referred to as permissive licenses. The permissive licenses allow one to use, change and redistribute codes. The copyleft licenses, in addition, stipulate that codes have to be redistributed with the same license conditions as the original code. This stipulation can force a change of license if such code is incorporated into a package with a permissive or proprietary license. Examples of codes released with permissive licenses are QMCPack[14], PySCF[81], PyQuante[108], and NWChem. Codes with copyleft licenses include ABINIT[109], ACES III[110], BigDFT[111], CP2K[35], Dalton[112], MADNESS[113], MPQC[38], Octopus[114], and Siesta[40]. A last variant are limited copyleft licenses that provide that codes can be linked as libraries to other codes without affecting those code's license. The LGPL licenses are common licenses of this variant that are used by codes such as Psi4[80].

Revision control systems provide tools to track the changes of the source code of a package. Importantly this allows going back to previous versions as well as documenting what changes were made, when, by whom, and why. Over time these systems have become increasingly sophisticated. The early RCS[115] system offered such functionality to a single developer. CVS[116] added a server component allowing multiple people to collaborate on a single code, but it is also treated every source code file independently. Subsequently, Subversion[117] treated a whole package as an integral entity. All these revision control systems assumed a central repository containing the master version of the code. The Git[118] revision control system did away with the central repository and provided a mechanism to create multiple new repositories. Such a new repository can be used to experiment with new ideas that will feed back into the original repository if successful, but it may also be the beginning of a new code that follows an independent development path. In addition the concept of a pull request was introduced, which is a request to the maintainers of a package to incorporate a developer's changes. The pull request also provides an opportunity to review the proposed changes before they are merged into the master branch. Of the open source packages mentioned in the previous paragraph ABINIT, and BigDFT use Bazaar[119], ACES III uses CVS, PyQuante uses Subversion, CP2K, Dalton, MADNESS, MPQC, NWChem, Octopus, PySCF, QMCPack, and Siesta are all using Git.

Revision control systems later than and including CVS need a server where the code is hosted. The CVS and Subversion systems assumed a single master repository and it was common for an organization that owned a code to host its own server. With open source packages, there no longer was need to protect against unauthorized individuals getting a copy of the code. SourceForge[120] was one of the first code hosting services that relieved developers of maintaining their own servers. In a similar spirit other hosting services have emerged including GitHub[121], GitLab[122], and Launchpad[123]. ACES III, COLUMBUS, deMon2K, and GAMESS use their own source code servers. PyQuante uses Sourceforge, Dalton, Octopus and Siesta use GitLab, whereas ABINIT, CP2K, MADNESS, MPQC, NWChem, PySCF, Psi4, and QMCPack use GitHub. BigDFT uses Launchpad.

Continuous integration (CI) is an approach whereby software is tested – ideally after every change. Tools to assist with such an approach have been available for over a decade (e.g. Buildbot and Jenkins), but they required additional servers to manage the build process and provide access to the test results. Private servers may cause issues for external collaborators accessing the test results. Services such as CDash[124] sought to address this problem by hosting the CI results on a public server. However, with codes being hosted on publicly accessible servers a similar development of CI services emerged. Examples of such services are Travis-CI[125], CircleCI[126], GitLab-CI[127], and GitHub Actions-CI[128]. ABINIT, PyQuante, MADNESS, MPQC, use Travis-CI, CP2K uses CP2K-CI, a custom solution based on Kubernetes, Dalton, and Octopus use GitLab-CI, NWChem, and PySCF use GitHub Actions-CI, Psi4, and Siesta use Azure Pipelines, QMCPack uses CTest on a private server and hosts the results on CDash.

Code coverage is concerned with the question of how much and what parts of the code are tested by the test cases for the package. Analyzing the code coverage requires that the executable be instrumented so that instructions executed can be related back to the lines of source code that generated them. Such instrumentation is available, for example, with the GCC and the LLVM compilers. The results can be gathered in reports that can be hosted, for example at Codecov[129]. ABINIT, CP2K, and Dalton use the GCC compilers to instrument the executable and the results are hosted on a private server. PySCF and Psi4 use PyTest with the coverage.py package. For C++ components, Psi4 uses the GCC compiler instrumentation for code coverage. QMCPack seems to use the GCC compiler instrumentation for code coverage but the results do not seem to be publicly available. MPQC, Octopus, Psi4 and PySCF host the coverage results on Codecov.

A relatively new development is trying to associate a quality measure with a code base.

Semmle[130] have developed a query language "CodeQL" (formerly .QL[131]) that can be used to interrogate a code base for the presence of known types of issues. Each issue found generates an alert and, using the rate of alerts as a function of the code base siz,e a quality score can be generated. A limitation for computational chemistry projects is that Fortran is not a supported language. These quality metrics are calculated for MADNESS, PySCF, PyQuante, Psi4, and QMCPack.

Code documentation describes how a code works and why, giving details about the design and implementation of the software. The tools available for generating documentation depend on the programming language used. Today, Doxygen[132] is a commonly used tool that supports C/C++, Fortran and Python. Python also has its own documentation generators such as Sphinx[133] or Epydoc[134]. In addition documentation can be hosted in a number of ways as well, be is a project specific server, on a wiki associated with the source code hosting service, or on a documentation service such as Read the Docs[135]. ABINIT uses Robodoc[136] for documentation generation. BigDFT, Dalton, Psi4, PyQuante, PySCF, and QMCPack use Sphinx for documentation generation. CP2K, MADNESS, Octopus, and Siesta use Doxygen for documentation generation. BigDFT, Dalton, Psi4, and QMCPack use Read the Docs for hosting. PyQuante hosts the documentation on SourceForge. Siesta hosts the documentation on GitLab. ABINIT, CP2K, MADNESS, Octopus and PySCF use their own server to host the documentation.

Application distribution, in many cases, was or is based on providing tar-files containing source code that prospective users unpack and build. More recently, open source codes have been ported for Linux distributions so that they can be installed with the usual package managers (ABINIT, CP2K, MPQC, and NWChem). For Python codes the usual Python tools can be used such as setup.py, Pip, or PiPy. Alternatively, a package might be available with the Anaconda package manager (PySCF, PyQuante, and Psi4). In other cases applications may have been containerized with ready to run containers available for download (BigDFT, Dalton, GAMESS, NWChem, and QMCPack). In some cases an application may even have been ported for cloud based deployment (Terachem).

Turning to the development of NWChemEx, modern software engineering practices have been incorporated from the outset. To enable extensive collaborations, NWChemEx is released under the permissive Apache 2.0 license[137]. Collaboration is furthered by managing the entirety of NWChemEx under the Git version control system, which supports a decentralized development model. Furthermore, NWChemEx is hosted on GitHub, which provides convenient web access to the code, complemented with collaboration tools such as issue trackers, milestones, support for CI pipelines, and development activity reports. The NWChemEx code uses CMake[138] for its build infrastructure. While CMake is a powerful tool, it requires users to carefully manage build instructions for builds to proceed as intended. In fact, managing the multiple toolchain dependencies involved in a complex code such as NWChemEx is still extremely difficult across multiple hardware and software platforms. To simplify using CMake, a suite of CMake modules was created, called CMakePP[139], which focuses on automation and boilerplate reduction. Using CMakePP, dependencies can be downloaded, built, and installed with as little as their URLs; libraries and executables can be added simply by providing the path to the directory containing the source files; and packaging files are automatically generated for installed targets. At present, much of the NWChemEx software is still in private repositories as legal details are being worked out; however, the intent is to have open repositories in the near future.

A crucial part of NWChemEx's CI pipeline is unit testing. NWChemEx's unit tests are designed to run quickly and are focused solely on determining if all "units" of NWChemEx are running correctly. Notably, this means that the unit tests do **not** ensure that NWChemEx runs at scale, or that NWChemEx is meeting the required performance metrics, rather such considerations are left to performance tests. Since performance tests require significantly more computing resources and time than unit tests, performance testing is done periodically outside of CI. NWChemEx is a dual-language project, and both the C++ and Python components are subject to unit testing. Catch2[140] is used for unit testing the C++ components. Compared to other C++ unit testing frameworks, we chose Catch2 because it:

- is light-weight (source consists of a single header file),
- supports a wide range of correctness checks, and
- boasts a syntax for assertions natural for typical C++ code.

For Python unit tests, NWChemEx relies on the *de facto* standard, Python's built-in `unittest` module. It is worth noting that NWChemEx's unit tests not only ensure correctness, but also that errors are caught and reported correctly, *i.e.*, that a function throws when it should.

In addition to unit testing, NWChemEx's CI pipeline also automatically ensures that the source code is formatted consistently, thereby facilitating code readability. Potential software contributions are also subject to human-based code review as part of the CI pipeline. This code review is focused on ensuring code is correct and extensible, but also serves as an informal means for developers to improve their coding technique. Development efforts are presently underway to additionally use the CI pipeline to automate the generation and hosting of user and developer documentation, code coverage, and ensuring that coding best practices are followed. The documentation is generated using a range of tools. Doxygen[132] is used to generate documentation from C++ sources, which is included in the final documentation by Sphinx[133]. Sphinx is also used to generate the documentation for Python sources. Code coverage reports are generated from the unit tests using GNU compiler features in combination with Gcov[141] and Gcovr[142]. The reports are hosted on Codecov[129].

Admittedly, many of the aforementioned software engineering practices are becoming routine in no small part because sites like GitHub make such practices easy. In addition, the National Science Foundation funded MolSSI has aided in educating new and current developers on the use of best practices in molecular software implementation.[143] That said, some of NWChemEx's most appealing features come from software engineering practices in the code itself. In particular, the memoization[144] design allows the SDE to automate saving and loading (checkpoint/restart) an ongoing calculation and enables interactive Python workflows. Without memoization, computationally expensive steps would need to be rerun every time the line of code is rerun (such as when a cell in a Jupyter Notebook is refreshed), but with memoization such calculations are only rerun when the input changes. Another somewhat unique feature of the SDE is that it enables modules to self-document themselves through reflection. In addition to saving the developer time, it also ensures that the documentation stays up-to-date with the source code.

5 Tools and Runtime Environments

Computational chemistry software often relies on underlying libraries to accomplish very specific tasks such as memory management, communication between nodes, and scheduling of execution on those nodes. Much of the computational chemistry community has adopted standards such as MPI, OpenMP, OpenACC, and CUDA to accomplish parts of these tasks. However, there have been specific developments within the chemistry community (often in collaboration with computer scientists) to manage these tasks. A few examples include the early development of TCP Linda[145] in Gaussian to handle memory and some distributed multiprocessor communication, the Distributed-Data Interface (DDI) in GAMESS[146], and the CHARM++[147–149] programming and runtime environment that is used for multiple codes such as NAMD and OpenAtom[150]. Each tool provides a base on which the overlying structure of the code is built.

Different tools reflect different approaches to achieving performance at scale. For example, CHARMM++ focuses on reducing load imbalance and overlapping communication and computation to eliminate bottlenecks to scaling. The runtime enables users to decompose there computations into relatively small units of work which are then distributed to available processors. CHARMM++ seeks to distribute work so that processor idle time is minimized while also scheduling tasks so data is continually moving to and from processes as tasks are being executed. The fine-grained decomposition of the problem ensures that there are always a large number of tasks relative to processors and provides flexibility to the runtime to backfill idle processors with more tasks. This strategy has been used to implement the highly scalable NAMD molecular dynamics code. The force matrix can be decomposed much more finely than the traditional spatial decomposition, vastly increasing the number of units of work available for parallelization, and therefore increasing the number of processors that can be used for solving a given sized problem.

As another example, CP2K's strategy for solving large scale systems on parallel computers is to concentrate on methods that solve for the density matrix and avoid matrix diagonalizations altogether[151]. The density matrix remains relatively sparse, in contrast to the eigenvector matrix, which is typically dense. Furthermore, the non-zero values tend to be concentrated in blocks that align with blocks of basis functions associated with atomic centers. The density matrix then has a sparse block structure in which relatively dense blocks are distributed sparsely in a larger matrix. The core operation in the algorithms used in CP2K is matrix-matrix multiplication, where two matrices with the sparse block structure are multiplied together to produce a third matrix, again with a sparse block structure. This operation is encapsulated in the Distributed Block, Compressed Sparse Row (DBSCR) library that provides a very high performance implementation of this operation. The DBSCR library is a combination of data transfer, node local multiplication and a very highly tuned library for multiplying individual blocks. The block multiplications consist of small matrices, with moste dimensions selected from a finite, enumarable set of values, enabling optimization for each set of block dimensions. The DBSCR library provides most of the parallel capability on which the remaining algorithms and models are built. Encapsulation of the matrix-matrix operation enables computer scientists with no background in chemistry to work on its implementation and help with both algorithm development and performance tuning.

Similarly, NWChem is an example of a co-design effort where parallel tools were developed in conjunction with the application to enable tera- and peta-scale computations. In this context, the most important co-designed tool is the GA environment,[41,152–157] which is used in NWChem, MolPro, GAMESS, GAMESS-UK, Columbus, and Molcas. After some experimentation in the early development stages of NWChem, the GA programming model emerged, and a collaboration with IBM supported GA with the development of LAPI (low-level API).[158] The GA portable interface allows each process to independently, asynchronously, and efficiently access logical blocks of physically distributed, dense, multi-dimensional arrays, with no need for explicit cooperation by other processes and providing one-sided put/get/accumulate access operations upon patches of arrays regardless of their physical location. Thus, the GA model retains many familiar and productive aspects of shared-memory programming models. However, the GA NUMA model also acknowledges that remote data are slower to access than local data, and it allows data locality to be explicitly specified and used. In these respects, the GA model is similar to message passing. NUMA is a very important concept in the performance of all modern computers — high performance demands that algorithms and compilers optimize usage of the memory hierarchy formed by registers, caches, memory, and data accessed via the network or from storage. If a program ignores this structure, performance is seriously degraded. In many situations, GA could actually deliver higher performance than message passing since the latter requires cooperation between sender and receiver, which makes this programming paradigm difficult to use efficiently. Eliminating this synchronization and facilitating random data access to distributed data greatly enhances the scalability and load balancing of irregular algorithms, such as SCF, for which sparsity and the highly variable cost of integral evaluation produce great irregularity in task execution times. Moreover, since message passing requires a protocol be built upon communication networks that, at their lowest level, just move data between buffers in memory, message passing can degrade latency and bandwidth, especially for short messages.

In addition to the primitive one-sided remote memory access capabilities that facilitated writing high-performance kernels, GA provided a very high-level interface for operations on entire matrices (e.g., scaling, addition, many linear algebra operations) that made the rest of the code much easier to write. Indeed, our first few attempts at a parallel programming model for NWChem failed this crucial productivity test — 90+% of any code is not performance critical (it just needs to be correct and fast enough). Hence our focus has always been on both performance and productivity.

The GAMESS team developed a very similar framework called Distributed Data Interface (DDI)[146], though it does not rely on one-sided communication. DDI was later generalized (GDDI) to a two-level hierarchy, where subgroups are assigned coarse-grained computational tasks (for example for fragment molecular orbital approaches) and each subgroup utilizes finer-grained parallelism[159]. MOLPRO builds on the Parallel Programming Interface for Distributed Data (PPIDD), which relies on either GA or MPI2 for its parallel data movement[160], while MOLCAS has integrated GA in certain modules for parallel scalability[161].

Despite the success of GA, it is no longer sufficient by itself to support the development of code using modern computer architectures and programming models. GA is primarily focused on asynchronous data movement and provides a simple memory model, whereas in modern computation, many new features come into play. It can often be more efficient to move computation to the data through techniques such as remote task creation or active messages. Since it lacks any understanding of execution, GA cannot assist the programmer in optimizing for multi-threaded or accelerated architectures. It has only two levels of memory (local and remote), whereas modern memory hierarchies are much more complex. Data consistency in GA requires synchronization and exposes algorithms to the full round trip latency of communication unless explicit and complex asynchronous programming methods are used. Modern computation understands how to use oversubscription and parallel slackness to effectively hide all latencies. With the advent of processors that efficiently handle many threads, devoting one or more threads to provide rich remote services is no longer a waste of substantial computing resources. Finally, while there has been substantial work extending the GA model to sparse computation, efficient support of distributed-data structures requires a richer set of primitives than simply remote data access. Both TAMM and TA described below take into account sparsity that can be used for reduced-scaling chemical algorithms.

One of the major tools that NWChemEx will rely on to aid in effective runtime data movement and execution is MADNESS. The MADNESS (Multiresolution Adaptive Numerical Environment for Scientific Simulation) software [113] provides a general-purpose numerical environment for general applications by providing adaptive meshes and fast solvers on trees to solve differential and integral equations in many dimensions[162,163]. The core data structures in MADNESS are irregular k-d trees, used for compact representation of numerical functions over multi-dimensional spaces. The MADNESS runtime environment provides a native C++ interface to solve the challenges of expressing novel and still evolving numerical algorithms and representations, while achieving good scalability for the compute functions traversing irregular spatial trees. The runtime environment includes an active messaging layer, distributed-data management customized for spatial trees, and extensive asynchronous execution and communication. MADNESS has applications in nuclear physics[164–167], boundary value problems[168], chemistry[163,169–171], solid-state physics[172], and atomic and molecular physics in intense laser fields[173].

Like all of the numerical libraries of MADNESS, the MADNESS parallel runtime provides a user friendly interface that allows the user to compose algorithms as a set of dynamically scheduled tasks that operate on objects in global namespaces. This high-level approach to parallel programming offers greater composability than that of MPI and explicit thread programming. Key runtime features include the use of

1. *global namespaces* for building applications that are composed of (global) objects interacting via remote methods;

2. *tasks* as first-class entities for fine-grained work decomposition;

3. *futures*[174] for expressing dependencies between the tasks.

These features permit the programmer to focus more on high-level concepts central to the scientific domain of interest and to deal less with the explicit low-level details of computation and data partitioning (e.g., message passing between processes with disjoint data or threads operating on shared data). The MADNESS runtime enables an algorithm to achieve dynamic load balancing using data-driven work scheduling and to hide significant amounts of the data communication.

All of these individual concepts appear in other packages, for example, in Cilk[175,176], Charm++[147–149], Intel® Threading Building Blocks (TBB)[177], and other projects, including ACE[178,179] and partitioned global address space (PGAS) languages[180]. Some of these features have made their way into mainstream programming languages. C++, for example, has included task-based concurrency with asynchronous function calls and futures since 2011. The MADNESS runtime is composed to allow powerful construction of parallel programs, one of its distinguishing features. For example, MADNESS futures, unlike other future implementations, can refer not only to results of local tasks, but also to those of *remote* tasks, thus directly supporting composition of *distributed* task-parallel algorithms.

As mentioned in Sections 3 and 4, the NWChemEx project has developed the SDE framework[96] to decouple the software into modules with standardized APIs using property types that are unique for each type of module. The SDE is responsible for storing the list of available modules and properly injecting them at call back locations. The list of available modules, as well as where they are to be injected, is populated at runtime. Thus changing the contents of this list provides a straightforward mechanism for interoperability and rapid prototyping without direct modification of the NWChemEx source code. In order to further facilitate interoperability and rapid prototyping, the entirety of the SDE's API is available *via* optional Python bindings. The result is that modules can be written in C++ or Python, and called from either language without any further consideration. Python is a natural language choice for the input layer to a package, hence a natural extension of the SDE's Python bindings is a user-friendly scripting layer. Psi4 (among others) also uses Python as it's front end for chemical computations[181], using MolSSI's Quantum Chemistry Schema as a standardized data format[182].

While the modular nature of NWChemEx arising from the SDE is ideal for rapidly refactoring the code base, it also introduces challenges with the data flow. This is because the rigidity of the property type API prohibits one from passing any additional input data to the module. To illustrate, consider a self-consistent field (SCF) energy module and a separate second-order Møller–Plesset perturbation theory (MP2) energy module. Both modules adhere to an energy property type API, which designates the molecular system as the only input parameter. The MP2 module perturbs the reference energy which is precomputed by the SCF module, but the property type API forbids passing the SCF energy to the MP2 module. This data flow restriction is addressed within the SDE by utilizing memoization[144] to cache and retrieve module results. Under the memoization technique, module results are indexed by hash keys which uniquely identify each result according to the module's input parameters. Every key-result pair is stored in a hash table data member of the SDE *Cache* class. Note that the *Cache* does not store the actual result data, but instead holds a shared pointer which references the storage location of the result. Whenever a module is called, the input parameters are hashed to produce the hash key prior to the execution of the module body. If the hash key matches an entry which already exists in the *Cache*, then the module call is replaced by retrieval of the stored result. Subsequently, one can share data between modules by nesting module calls. For example, the body of an MP2 energy module may include a call to an SCF energy module. The SCF hash key would match the hash table entry for the precomputed SCF energy, so the result would be retrieved for subsequent use in the MP2 module.

Inter-module data flow is just one of several applications of memoization in NWChemEx. The technique is also used in the checkpoint/restart component of the fault tolerance design. Checkpointing is accomplished by copying *Cache* entries to a stable storage medium, such as disk or non-volatile memory. The entries must be serialized into a format from which the module results can be reconstructed during a restart event. Serialization/deserialization support for most C++ Standard Template Library (STL) types is provided by the cereal[183] library and the MADNESS runtime[113]. Native support for binary, JavaScript Object Notation (JSON)[184], and Extensible Markup Language (XML)[185] representations already existed in cereal and MADNESS. Some other codes that utilize XML or JSON as output include MOLPRO, Psi4, MPQC and PySCF. NWChem initially explored the use of the Chemical Markup Language (CML)[186], before switching to JSON[187]. JSON and XML formats provide human-readable data representations which are useful for analyzing results and are common formats for data interfaces to external software such as workflow tools. The Hierarchical Data Format (HDF5) format is the primary format used for checkpoint/restart owing to its parallel capabilities and high performance when manipulating large datasets. Calculations are restarted by reconstructing the *Cache* from the checkpoint file. Other than loading the checkpoint file, there is no explicit code to implement restart logic. The call tree for the computation is simply re-executed, with all modules retrieving memoized results until parity is reached with the checkpointed state. Notably, there are additional use cases beyond resiliency for reconstructing checkpointed *Cache* entries. For example, a performance benefit is possible for sets of closely-related computations (e.g., algorithmic parameter sweeps) with redundant computations by loading repeated *Cache* entries from a single checkpoint file.

6 Tensor Methods

Tensor contractions occur widely in various formulations of many-body formalisms ranging from independent particle models to correlated methods such as CC formulations. A typical example of a tensor contraction is given by the expression

$$A(i,j,k,l) += B(i,j,m,n) \times C(m,n,k,l) \ , \tag{1}$$

where A is the four-dimensional output tensor, B and C are the four-dimensional input tensors, and Einstein summation convention over the repeated indices is assumed - in the above example summation runs over m and n indices. One of the first uses of tensors was the Pinnacle software that was an early, in-house C++ code to compute Møller–Plesset energies.[188] While this early work did not gain traction at the time, these ideas have been combined with automatic generation of CC codes such as those by Janssen and Schaefer,[189] Li and Paldus,[190] Kállay,[191] and Nooijen and co-workers[192,193] leading to the modern ideas of many-body interaction codes.

The emergence of parallel computing has also triggered a significant effort toward automatic generation of scalable CC codes, which resulted in the development of specialized systems that integrate elements of symbolic algebra for manipulating and optimizing tensor expressions with efficient parallel tensor libraries. Specialized distributed-memory (or in some cases shared-memory) libraries for automatic tensor blocking, tensor redistribution, and efficient utilization of tensor symmetries such as Tensor Contraction Engine (TCE)[194,195], super instruction assembly language (SIAL)[110,196], libtensor[197], Cyclops Tensor Framework (CTF)[198–200], and TiledArray (TA)[201] play a key role in enabling high-accuracy CC methods in many community codes including NWChem[202], Aquarius[203], CFOUR[204], MPQC[38], Q-Chem[39], ACES[24], and MRCC[205]. Below, three of these tensor packages, TCE, TAMM, and TA are discussed as relevant to the NWChem (TCE) and NWChemEx (TAMM and TA) packages.

6.1 TCE

An integral part of NWChem is the TCE,[194,195,206] which is a software package that provides a high productivity abstraction layer by enabling tensor computations/contractions in a high-level language instead of low-level Fortran, and that then automatically generates parallel Fortran code. TCE is a symbolic manipulation program of second-quantized operators and a program generator, where an ordered list of binary tensor contractions, additions, and index permutations is translated into an optimized program. All TCE generated codes take advantage of spin and spatial symmetries for real Abelian point-group symmetry, and index permutation symmetry at every stage of the calculations to minimize the number of arithmetic operations and storage requirements. TCE also adjusts the peak local memory usage by index-range tiling, stores an operation tree as a data structure analogous to a directed acyclic graph, and supports parallel I/O interfaces and dynamic load balancing for parallel executions. Synchronization and load balancing is achieved through shared variables that are atomically updated using GA operations.[41,207] By examining the data dependencies in the memory blocks of each matrix, additional parallelism can be obtained even in cases were very good parallelism exists.

The data representation assumed by TCE is based on the partitioning of the entire spin-orbital domain into smaller subsets (tiles) containing spin-orbitals corresponding to the same spin and spatial symmetries. The occupied and unoccupied tiles (i.e., tiles containing occupied/unoccupied spin-orbital indices, respectively) are designated as `[i]`, `[j]`, `[k]`, ... and `[a]`, `[b]`, `[c]`,..., respectively. This division entails the partitioning of all tensors involved in the CC calculations, including cluster amplitudes, recursive intermediates, and integrals. For example, the tensor corresponding to doubly excited amplitudes is stored in the block form defined by smaller 4-dimensional tensors

$$\mathtt{t}^{\mathtt{[i][j]}}_{\mathtt{[a][b]}} \tag{2}$$

representing a subset of doubly excited amplitudes defined by the indices belonging to the `[i]`, `[j]`, `[a]`, and `[b]` tiles. This block structure of CC tensors also defines the granularity of the code. Parallelization and dynamic load balancing occurs over do-loops of occupied/unoccupied tiles. These units of parallel work contribute to a task pool – a collection of tasks that can be executed in parallel – where scalability of the calculation is directly related to the size of its task pool. The `tilesize` input parameter can be used to define the maximum size of a tile and to tune the granularity to given architecture specifications. The `tilesize` also defines the efficacy of `dgemm` calls and local memory requirements. While for the iterative CC singles and doubles (CCSD) and equation-of-motion (EOM) CCSD methods the local memory demand is proportional to $(\mathtt{tilesize})^4$, the analogous demand for the CCSD(T) perturbative triples part amounts to $2\times(\mathtt{tilesize})^6$. In order to overcome this bottleneck a version of the code where the 6-dimensional tensors can be dynamically decomposed along the first two dimensions to match available local memory have been developed.[208]

The TCE environment has been used to generate a number of canonical implementations of single-reference CC methods for ground- and excited-state calculations for arbitrary reference functions including: restricted, restricted open-shell, and unrestricted Hartree–Fock (RHF, ROHF, UHF) cases. Moreover, the TCE framework has also enabled various types of state-specific multi-reference CC methods. Examples of the type of scalability possible with TCE is given in Figure 1.

While TCE has been successful in abstracting the programming of tensor operations, generated codes are not necessarily ideal for enabling computation at the exascale. Especially with the increase in scale, complexity, and heterogeneity (CPUs, coprocessors, and accelerators) of modern platforms, traditional programming models fail to deliver the expected performance scalability. The main road-blocks precluding TCE from reaching the exascale regime can be attributed to two factors:

1. lack of an efficient execution model that would define and utilize interdependencies between particular tasks,

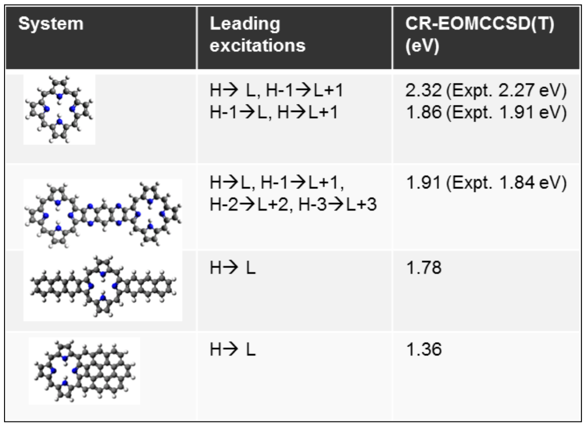
2. problems with data localization across the entire network - although a small portion of data can be replicated, the large-size tensors distribution may not be synchronized with a task pool and required data flow.

In many cases, these problems lead to insurmountable network congestion problems.

6.2 Tensor Algebra for Many-body Methods (TAMM)

The Tensor Algebra for Many-body Methods (TAMM) library provides one piece of infrastructure for NWChemEx to achieve a scalable performance–portable implementation of key modules on exascale supercomputing platforms. In addition, TAMM extends the capabilities needed for dense tensor contractions to contractions of block sparse tensors where the sparsity structure is not known until runtime. TAMM provides a flexible infrastructure to specify and manipulate data distribution, manage memory, and schedule tensor operations. This infrastructure is, in turn, implemented using GA and MPI for scalable parallelization on distributed-memory platforms and using optimized libraries for efficient intra-node execution of tensor operation kernels on CPUs and accelerators.

TAMM builds on the experience of NWChem's TCE and improves upon it in several important ways. TCE focused on the generation of parallel Fortran 77 programs to execute MO spin-orbital tensor contraction expressions. TAMM is designed to support a general



System	Leading excitations	CR-EOMCCSD(T) (eV)
	H→ L, H-1→L+1 H-1→L, H→L+1	2.32 (Expt. 2.27 eV) 1.86 (Expt. 1.91 eV)
	H→L, H-1→L+1, H-2→L+2, H-3→L+3	1.91 (Expt. 1.84 eV)
	H→ L	1.78
	H→ L	1.36

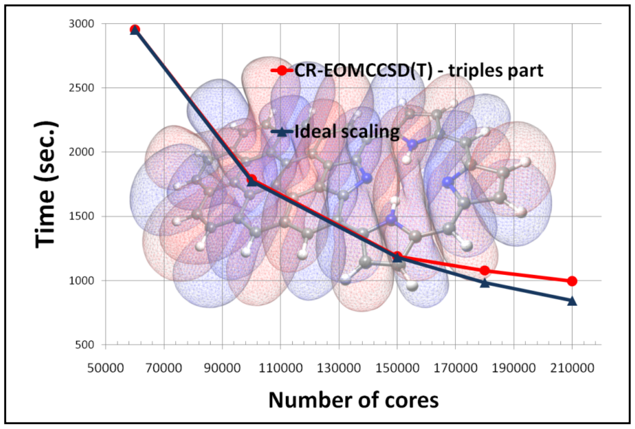


Figure 1: Benchmark systems considered in EOMCC calculations: free-base porphyrin (FBP), fused porphyrin dimer, FBP-fused-anthracene, and FBP fused coronene (left panel). Scalability of the triples part of the CR- EOMCCSD(T) approach for the coronene fused free-base porphyrin in the aug-cc-pVTZ basis set. Timings were determined from calculations on the Jaguar Cray XT5 computer system at the. National Center for Computational Sciences (right panel).

notion of *index spaces,* allowing the development of a much larger class of methods. A user can create multiple index spaces corresponding to, for example, atomic, molecular, or localized orbitals, and simultaneously use them in implementing a method[209].

A method is implemented as a sequence of operations. This design separates the specification of the operations from their implementation. The TAMM scheduler manages the execution of a collection of operations. It analyzes the dependencies between the operations and the computation-communication requirements to execute operations in the most load-balanced and communication-efficient way with the fewest number of global synchronizations. Whereas TCE generated code to employ a specific parallelization strategy, the TAMM code is schedule-independent. In particular, TAMM supports a variety of schedules to choose from based on the operations to be executed.

In addition to automated load-balanced execution of tensor operations, TAMM is designed to allow user control. The users can control data distribution, operation ordering, and the choice of parallelization strategy to construct custom *execution plans* for specific computing platforms or input classes. To enable such automation, TAMM implements multiple mechanisms that a user can choose from. Given the constraints imposed by the user, TAMM automates the rest of the execution plan to construct an efficient execution schedule. This allows the user to incrementally adapt the code using tailored execution strategies with minimal programming effort. With the code generation strategy employed by TCE, each operation's schedule is explicitly encoded in the generated code. TAMM's runtime interpretation approach reduces the resulting code size, making it more readable and optimizable.

Execution of operations using the TAMM library involves multi-granular dependence analysis and task-based execution. At the coarsest level, the dependencies between operations are analyzed to construct a macro operation-graph. When two operations share the same data structure with one of them writing to it, the operations are said to conflict and cannot be executed in parallel. The operation-graph is analyzed to identify and order the non-parallel operations to minimize the number of synchronizations required. The operations that can be scheduled in parallel are executed in a single program multiple data fashion. The execution is compatible with MPI, and the operations are collectively executed on a given MPI communicator.

Each operation is further partitioned into tasks. The tasks that constitute an operation are produced using task iterators. Each task computes a portion of the operation, typically a contribution to a block of data in the output tensor. Until it begins execution, a task is migrateable and can be scheduled for execution on any compute node or processor core. Once the execution of a task begins, the data required by the task are transferred to its location. At this point, the task is bound to the process in which it is executing and cannot be migrated.

This hierarchical parallelization enables the coordinated use of optimizations at different granularities. For example, the user might decide to replicate an often-required small tensor. This information can be used to automatically optimize locality for the remaining tensors and automate communication overlap. The computation and communication requirements of the tasks are profiled as they are executed. When an inefficiency (e.g., load imbalance or excessive remote communication) is detected, the tasks are remapped to address the inefficiency. This new schedule is then used when the computation is executed again – as is the case in an iterative algorithm.

Optimized execution of each task requires kernels optimized for the given platform. This includes the exploitation of faster memory, GPUs, and vectorization choices. TAMM is designed to allow the use of multiple external libraries that can provide these kernels. For GPUs, the kernels from TAL-SH[210,211] and TensorGen[212] are used to ensure efficient execution. In complementing the kernel execution, TAMM manages the data distribution and transfer to minimize the data movement costs.

6.3 TiledArray

TiledArray (TA) is a generic framework that provides efficient implementation of the algebra of distributed-memory dense and block-sparse tensors. The development of TA has been driven by the needs of the newly-reengineered MPQC package,[38] thus TA is designed to be sufficiently generic to provide most of the features expected from a tensor library in the electronic structure context (e.g., lazy evaluation of tensors for integral-direct electronic structure methods, irregular tiling, and general sparsity models, among others) by non-intrusive customization. Nevertheless, TA is fully domain-neutral, i.e., it is free of domain-specific concepts, such as spin, but it can be customized to support domain-specific features.

TA supports multiple composition styles. The most common is the high-level math-like "language" that allows non-experts to compose parallel tensor algebra in math-like form; e.g., the following expressions for the MP1 amplitude residual and the MP2 energy,

$$R^{ij}_{ab} = G^{ij}_{ab} + T^{ij}_{ac}F^{c}_{b} + T^{ij}_{cb}F^{c}_{a} - T^{ik}_{ab}F^{j}_{k} - T^{kj}_{ab}F^{i}_{k} \tag{3}$$

$$E^{(2)} = G^{ab}_{ij}(2T^{ij}_{ab} - T^{ji}_{ab}) \tag{4}$$

are written in TA C++ as

$$\begin{aligned}\texttt{R("i,a,j,b")} = & \texttt{G("i,a,j,b")} + \texttt{Fv("b,c")} * \texttt{T("i,a,j,c")} \\ & + \texttt{Fv("a,c")} * \texttt{T("i,c,j,b")} - \texttt{Fo("j,k")} * \texttt{T("i,a,k,b")} \\ & - \texttt{Fo("i,k")} * \texttt{T("k,a,j,b")};\end{aligned}$$

$$\texttt{double energy} = \texttt{G("i,a,j,b").dot(2*T("i,a,j,b")} - \texttt{T("i,b,j,a"))};$$

with the corresponding TA Python code looking nearly identical to this.[38] This style of programming also supports expert-level features such as complete programmatic control of sparsity, data distribution, and work partitioning. Other lower-level programming styles, such as functional-style iteration over tensor blocks to explicit loops over tile indices and direct byte-level access to the data, are also supported to provide experts with the ability to compose arbitrary algorithms over general sparse tensorial data structures.[213]

TA has been designed to support efficient execution on modern and future hardware of all scales, from a single multi-core machine to a cluster of multi-core, multi-GPU nodes, to leadership-class supercomputers. To maximize the concurrency and hide latency, which is crucial for alleviating the load imbalance and lower computation-to-communication ratio of the irregular sparse tensor algebra, TA has an asynchronous, dataflow-style core. Namely, the tensors in TA are collections of *futures* to data tiles, fulfilled by asynchronously executing tasks, whose scheduling is driven by the data flow between tasks and between memory spaces (e.g., between memories of different nodes, or between host and device memories in heterogeneous nodes). By increasing the exploitable concurrency, the fine-grained task composition allows one to overlap communication and computation, as well as overlap execution of tasks within individual and across multiple unary/binary tensor expressions, where possible. These low-level details of TA are implemented using the MADNESS parallel runtime.[113]

In addition to the MADNESS runtime (which requires a thread-safe MPI[214] implementation), TA also requires a BLAS library[215] for optimized implementation of vector and matrix algebra. Interfaces to linear algebra packages Eigen[216] and ScaLAPACK[217] and tensor algebra library BTAS[218] are provided. For execution on CUDA-capable accelerators, TA also requires the CUDA toolkit,[69] the cuTT tensor transpose library[219] and Umpire memory management library.[220]

TA is implemented in the standard C++ language; the most recent version obeys the 2017 C++ ISO standard for most of the code (the CUDA-specific code uses the older, 2014 standard). TA bindings to Python were also developed recently.

A tensor contraction in TA, implemented as an asynchronous formulation of the 2D SUMMA algorithm,[221] has been demonstrated to scale efficiently to up to 256K cores for large dense tensors; excellent strong scaling was also demonstrated for contractions of block-sparse and block-rank-sparse tensors.[201] Implementation of ground-state and excited-state CC methods in MPQC based on TA have demonstrated excellent efficiency and strong scaling on both conventional and heterogeneous clusters.[222–224] Pilot implementation of pair natural orbital CC methods for ground- and excited-states have also been demonstrated.[223,225]

7 Solvers

Numerical solvers for HF and DFT eigenvalue problems and CC non-linear equations play an important role in computational chemistry. In earlier days, many quantum chemistry packages implemented native solvers using algorithms provided in numerical recipes[226] or embedded subroutines from EISPACK[227]. As computer architectures have become more complex, developing solvers that can run efficiently on modern distributed-memory parallel architecture becomes a challenging task[228]. Fortunately, in the numerical analysis community, standards for basic linear algebra subroutines (BLAS) emerged. By expressing many algorithms in terms of these building blocks, one can benefit from vendor optimized BLAS to achieve high performance with a minimal amount of effort. Furthermore, the development of open-source LAPACK[229] and ScaLAPACK[217] libraries allows a new generation of stable and efficient numerical linear algebra algorithms to be utilized by the computational chemistry community.

The exascale effort is inextricably associated with enabling a new class of solvers that can significantly reduce the number of iterations required to meet convergence criteria. Another aspect of the applied math effort is associated with identifying a physically meaningful solution that describes the physics/chemistry of interest. These problems are additionally amplified by the non-linear character of electronic structure methods targeted in the NWChemEx project, which includes HF, DFT, and CC formalisms. To illustrate the scale of these problems, one should realize that in exascale applications of the canonical CC formalism, one will have to solve sets of non-linear polynomial equations for 10^{11}-10^{13} wave function parameters. In the following part of this Section, we will describe state-of-the-art iterative formulations to address this issue.

7.1 Eigensolver for HF and DFT

In each SCF iteration (for HF and DFT), we need to solve a generalized eigenvalue problem

$$HX = SX\Lambda, \tag{5}$$

where H is the HF or the Kohn–Sham (KS) Hamiltonian matrix, S is the overlap matrix, and X contains a subset of eigenvectors of the matrix pencil (H, S) associated with the n_e algebraically smallest eigenvalues. Here n_e is usually associated with the number of electrons in the chemical system.

In most quantum chemistry codes, the eigenvalue problem (5) is solved by a dense eigensolver implemented in LAPACK[229], ScaLAPACK[217] or more recently in ELPA[230,231]. Iterative eigensolver such as the Davidson's method[232], the locally optimal block preconditioned conjugate gradient method[233], and the Chebyshev polynomial filtering method[234] are used for planewave[83], finite difference[235] and finite element[236] discretization of the DFT problem. Although recent progress in dense solvers has made them very efficient, there are a number of drawbacks to using these solvers directly in the SCF iteration.

- It is difficult to make these solvers scalable beyond a few thousand cores due to sequential bottlenecks in the reduction to a tridiagonal matrix procedure.

- One cannot easily trade accuracy for efficiency in these solvers. Since we typically do not need highly accurate eigenvectors in early SCF cycles, a more efficient solver that can provide approximate eigenvectors is sufficient.

- These solvers typically do not make use of approximate eigenvectors produced from the previous SCF cycle.
- Partial diagonalization can only be performed in the tridiagonal eigensolver, even though more than two-thirds of the time is spent in reduction to tridiagonal form.

In NWChemEx, we adopt the spectrum slicing technique[237–240] to design an eigensolver that is more scalable. The basic idea is to divide the desired part of the spectrum into several spectrum slices, each containing roughly the same number of eigenvalues. On the ith slice, we apply a subspace iteration to the shifted and inverted operator $(H - \sigma_i S)^{-1}S$ to compute eigenvectors associated with eigenvalues closest to σ_i. Because computation associated with different slices can be carried out independently and the subspace iteration can also be easily parallelized, this approach has multiple levels of concurrency that can be fully exploited for exascale computing platforms.

In this approach, we need to perform an LDL^T factorization of the shifted Fock matrix (KS Hamiltonian) $H - \sigma_i S$, and solve a number of linear systems within each slice. The diagonal factor provides inertia counts that allow us to make sure all desired eigenpairs are computed. A Lanczos-based algorithm is used to estimate the distribution of eigenvalues and provide an initial partition of the spectrum[241]. The spectrum partition in subsequent SCF cycles is obtained by analyzing the distribution of eigenvalues obtained from the previous SCF cycle.

Within the SCF iterative process, there are similarities to the commonly used direct inversion of iterative subspace (DIIS) method[242,243]. In the first SCF cycle, the approximation to the desired eigenvectors are generated as linearly independent random vectors. In subsequent SCF cycles, approximate eigenvectors obtained from the previous SCF cycles are used as the starting guess. Typically, three to five subspace iterations are sufficient in each SCF cycle.

Another recent development in this field is the ELectronic Structure Infrastructure[244] (ELSI) library. ELSI provides a unified interface for a variety of eigensolvers and density-matrix-based solvers with either sparse or dense matrix representations. ELSI is already integrated into DFTB+, DGDFT, FHI-aims, and SIESTA codes.[245]

7.2 Newton–Krylov Solver for Coupled-Cluster Equations

The CC equations are traditionally solved by an inexact Newton algorithm of the form

$$T^{(k+1)} = T^{(k)} - \left[\hat{J}^{(k)}\right]^{-1} R(T(k)) \tag{6}$$

where $\hat{J}^{(k)}$ is an approximate Jacobian matrix evaluated at $T^{(k)}$.

In CCSD calculations, a common practice is to choose $\hat{J}$ as a diagonal matrix with orbital energy differences as the diagonal elements. This works well for problems in which the Jacobian is diagonally dominant, which happens typically when the system is near equilibrium. For systems far from equilibrium, the diagonal approximation may not be sufficient.

In NWChemEx, we have implemented a Newton–Krylov method[246] for solving the CC equations. Even though J is not explicitly available, it is possible to approximate the product of $J(T)$ with any tensor W that has the same dimension as T. This can be accomplished through a finite-difference calculation of the form

$$J(T)W \approx \frac{R(T + hW) - R(T)}{h}, \tag{7}$$

where h is a small constant.

The possibility to approximate $J(T)W$ by one extra function evaluation makes it possible to solve the Newton equation by a Krylov subspace-based iterative method such as the GMRES algorithm[247], even when $J(T)$ is not explicitly available. This approach is often referred to as the Newton–Krylov method.

An iterative procedure for computing the solution to the Newton correction equation

$$J^{(k)}\Delta = R(T^{(k)}) \tag{8}$$

can be accelerated by using a preconditioner P. Instead of solving (8), we solve

$$P^{-1}J^{(k)}\Delta = P^{-1}R(T^{(k)}), \tag{9}$$

with the hope that $P^{-1}J^{(k)}$ has a smaller condition number that leads to faster convergence.

DIIS is a commonly used technique in many codes to accelerate the convergence of iterative methods for solving the CC equation and can be combined with the Newton–Krylov method. At the k-th iteration, we form a new approximation as

$$\tilde{T}^{(k+1)} = \sum_{j=k-\ell}^{k} \omega_j \left[T^{(j)} + \Delta^{(j)}\right], \tag{10}$$

for some constant ℓ, where the ω_j's are chosen to be the solution to the following constrained minimization problem

$$\min_{\sum_j \omega_j = 1} \| \sum_j \omega_j \Delta^{(j)} \|.$$

The $k+1$ amplitude approximation is then computed from

$$T^{(k+1)} = \tilde{T}^{(k+1)} - \tilde{\Delta}^{(k+1)},$$

where $\tilde{\Delta}^{(k+1)}$ is the approximate solution to the Newton correction equation (8) or (9).

In a conventional DIIS procedure, both $\Delta^{(k)}$ and $\tilde{D}^{(k)}$ are taken to be $D^{-1}R(\tilde{T}^{(k)})$ and $D^{-1}R(T^{(k)})$, respectively. When combined with the Newton–Krylov procedure, we simply compute both of these corrections by a Krylov subspace iterative solver, such as the GMRES method.

Figure 2 shows that the Newton–Krylov method is much more stable than the DIIS method, as indicated by the monotonic reduction of the residual norm.[248] For the Cr_2 test problem, the Newton–Krylov method and the combined Newton–Krylov method use fewer number of residual function evaluations compared with that used in the DIIS method.

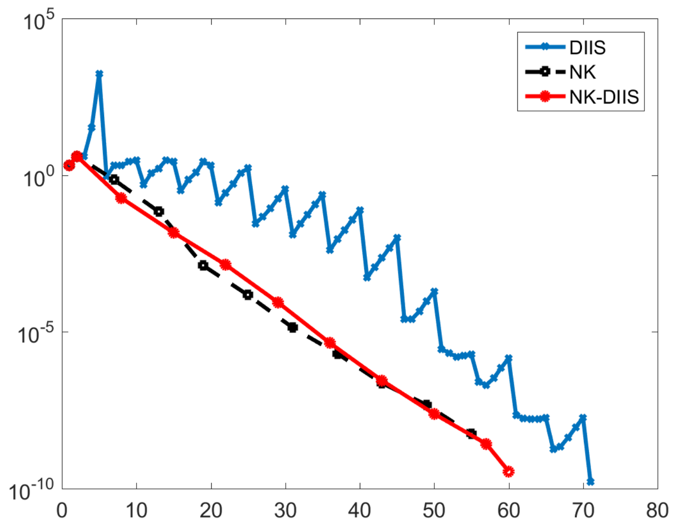


Figure 2: A comparison of the convergence of the Newton–Krylov, DIIS, and combined Newton–Krylov DIIS methods when they are applied to Cr_2 at equilibrium distance. For each method, we plot the residual norm at each iteration against the number of residual function evaluations up to that iteration. The DIIS acceleration was applied every 5 inexact Newton iterations and each Newton–Krylov iteration used a maximum of 5 GMRES iterations.

8 Gaussian Basis Hartree–Fock and Density Functional Theory

In many Gaussian basis set HF and Kohn–Sham (KS) implementations, both HF[249] and KS-DFT[250–252] are expressed as a set of matrix equations

$$\mathbf{FC} = \mathbf{SC}\epsilon \tag{11}$$

where $\mathbf{F}$, $\mathbf{C}$, $\mathbf{S}$ and ϵ represent the Fock, coefficient, overlap and diagonal orbital energy matrices, respectively. The problem of solving the HF/KS equations thus boils down to solving a non-linear, generalized eigenvalue problem. The Fock matrix for both HF and KS formalisms can be encompassed under a single framework as

$$F_{\mu\nu} = H_{\mu\nu}^{core} + G_{\mu\nu}^{j} + \alpha G_{\mu\nu}^{k} + \beta G_{\mu\nu}^{x-dft} + \gamma G_{\mu\nu}^{c-dft} \tag{12}$$

where H^{core} is the one-electron contribution (kinetic and ion-electron), G^j and G^k represents the two-electron (Coulomb and explicit exchange), G^{x-dft} and G^{c-dft} are the DFT exchange and correlation (XC) parts, respectively. The mixing coefficients α, β, and γ help span the HF and DFT limits. With $\alpha = 1$, $\beta = 0$, $\gamma = 0$ one gets the pure HF limit, while the pure DFT limit is obtained with $\alpha = 0$, $\beta = 1$, $\gamma = 0$ and $\alpha < 1$, $\beta < 1$, $\gamma = 1$ covers the phase space of non-local hybrid-DFT forms. Most local, non-local (gradient-corrected), global hybrid, range-separated hybrid and double hybrid and meta exchange-correlation functionals[253,254] have been implemented in NWChem.[11,202] As we enter the exascale era, the need to simulate larger systems with DFT requires reduced-scaling algorithms. Various strategies have been explored over the years in a number of established codes within different basis set formulations.[23,39,40,78,80,81,202,255–266] Since the parallel Gaussian basis HF and DFT implementations in NWChem have been described in detail in a recent review[202], we will focus on the implementation in NWChemEx in this paper.

Given that SCF computations are the starting point for most correlated computations, a large part of NWChemEx's initial focus is devoted to implementing a massively parallel, local, density-fit SCF algorithm. The resulting algorithm differs from the canonical algorithm in two key ways: first, density fitting is used to compress the order four electron-electron repulsion tensor into an order three variant; second, the use of localized MOs allows for the exploitation of sparsity in quantities involving MO indices (such quantities are typically dense given the delocalized nature of the canonical MOs).

The SCF module in NWChemEx is the first major electronic structure method implemented using the SDE. As a result, it is possible for the user to customize and extend almost every part of the SCF algorithm from the input file. In particular, it is possible to change the algorithms for building the: core Hamiltonian, Coulomb matrix, exchange matrix, Fock matrix, and initial guess. While our initial efforts have been primarily focused on implementing local variants of SCF, we anticipate that the flexibility provided by the SDE will make it easier to develop architecture-specific algorithms.

One of the most expensive steps in the HF method is the calculation of the exchange contribution to the Fock matrix. NWChemEx includes an implementation of a local, density-fit algorithm for building the exchange matrix based on the work of Köppl and Werner.[267]

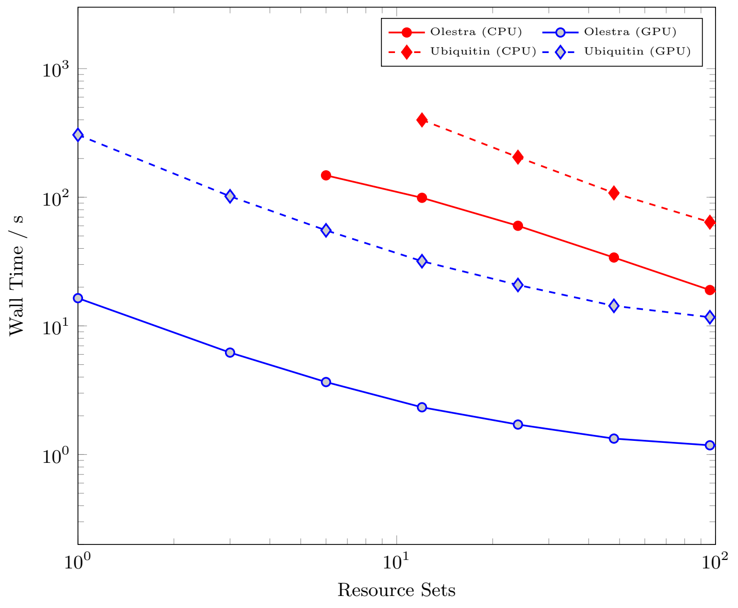


Figure 3: Strong scaling of the GPU accelerated XC integration in NWChemEx in comparison with the CPU implementation in NWChem for two large molecules in the 6-31G(d) basis set: Olestra (453 atoms / 3181 basis functions) and Ubiquitin (1231 atoms / 10,292 basis functions). Each Summit Resource Set consists of 7 CPUs and 1 GPU. NWChemEx exhibits nearly identical linear scaling behaviour as NWChem to hundreds of CPUs + GPUs while admitting O(5-10x) speedup over the CPU implementation.

Localized MOs are formed to introduce the necessary sparsity for reduced scaling. Subsets ("domains") of the atomic orbital (AO) basis and fitting functions are formed for each MO based on thresholds for the localized MO coefficients, three-center electron repulsion integral screening, and localized MO charges. The domains are defined such that all the functions on an atom center are either included or excluded in the domain. As the size of the system increases, domain sizes eventually become constant so that, asymptotically, the algorithm scales linearly with respect to the system size. The sparse tensors and contractions required by the method are implemented using the TA framework. In order to support larger tensor block sizes for efficient matrix multiplication, the NWChemEx implementation allows the domains to be defined as maps between groups of MOs to groups of atoms. This creates a trade-off between the efficiency of the tensor contractions and the amount of sparsity captured by the domains.

As a part of the SCF module, NWChemEx includes a scalable implementation of Kohn–Sham density functional theory (KS-DFT) for Gaussian basis sets. As the target of NWChemEx is performance on leadership class exascale computing architectures, the majority of development effort has been afforded the computation and assembly of KS Fock matrix components (equation (12)) while leveraging the computational power of GPU accelerators in a distributed environment. Over the last few decades, an immense research effort directed towards the development of GPU accelerated Gaussian basis set KS-DFT,[268–273] a majority of which has been focused on the development of highly efficient algorithms for the evaluation and digestion of the electron repulsion integrals (ERI) required for the formation of the Coulomb and explicit exchange components of the Fock matrix.[61,65,255,268,274–280] A major component of this effort in NWChemEx has been the development of an efficient and highly scalable algorithm for the numerical integration of the XC potential on clusters of GPU accelerators.[281] The key component of this algorithm is the reliance on highly tuned, microarchitecture optimized implementations of GPU accelerated batched level-3 BLAS operations such as matrix multiply (GEMM) and symmetric rank-2k updates (SYR2K) to achieve high computational efficiency across a wide range of GPU hardware. Example comparisons of the strong scaling of the XC integration in NWChem and NWChemEx for a set of large molecules are given in Fig. 3. Overall, the scalable GPU accelerated XC integration in NWChemEx has been demonstrated to exhibit O(5x-10x) speedups over the existing CPU implementation in NWChem.

9 Plane-Wave DFT

The pseudopotential plane-wave method and the related pseudopotential uniform grid real-space method is a popular, fast, and efficient way to implement KS-DFT[29,45,83,109,235,250,251,282–299]. In this approach, the fast varying parts inside the atomic core regions of the valence wavefunctions are removed and replaced by pseudopotentials and the related projector augmented wave potentials (PAW)[300–304], which are generated by requiring that the resulting pseudoatoms have the same scattering properties as the original atoms[305–311]. The justification for this approach is that the electronic structure of chemical bonding is in the interstitial region outside the atomic core regions. These methods over the years have become an important class of quantum chemistry methods that can be used to model chemistry and dynamics of molecular and condensed phase systems while retaining an electronic structure description of their interactions. Moreover, even with the difficulties in parallelizing certain kernels in these methods (i.e., fast Fourier transforms), highly efficient parallel algorithms have continued to be developed, which have allowed these methods to keep up over several generations in high-performance computing architectures.

Within the plane-wave approach the single-particle wavefunctions of a periodic system can be expanded, in general, as[292,293,295,312]:

$$\psi_{j\mathbf{k}}(\mathbf{r}) = e^{\imath\mathbf{k}\cdot\mathbf{r}} \sum_{\mathbf{G}} \tilde{\psi}_{j\mathbf{k}}(\mathbf{G}) e^{\imath\mathbf{G}\cdot\mathbf{r}} \tag{13}$$

where $\mathbf{k}$ is a vector in the first Brillouin zone, $\mathbf{G}$ is the reciprocal lattice vector, and j represents the orbital index. For molecules (isolated systems), the Brillouin zone sampling is limited to the Γ-point ($\mathbf{k} = \mathbf{0}$) which gives,

$$\psi_j(\mathbf{r}) = \sum_{\mathbf{G}} \tilde{\psi}_j(\mathbf{G}) e^{\imath\mathbf{G}\cdot\mathbf{r}} \tag{14}$$

The size of the plane-wave basis set expansion is determined by the maximum kinetic energy cutoff (E_{cut}) via the following relation:

$$\frac{1}{2}\left|\mathbf{G}\right|^2 < E_{cut} \tag{15}$$

Some favorable features of the plane-wave expansion include: being able to treat periodic systems in a seamless way, efficient calculation of the expansion coefficients using Fast Fourier Transform (FFT) techniques, independence of the basis with respect to nuclear positions which makes it immune to superposition and over-completeness problems which are critical issues in local basis set approaches.

Since plane-wave basis sets are typically much larger than a local basis, explicit Fock matrix construction and diagonalization is avoided in favor of direct optimization approaches like conjugate gradient minimization[286,313,314]. The most time-consuming steps of plane-wave-based algorithms are the evaluation of the pseudopotential (specifically the application of atom-based projectors in non-local pseudopotentials), wavefunction orthogonalization, and exact exchange (if needed) in the description of the exchange-correlation.

Several parallelization strategies have been explored by several groups[48,48,82,292,304,315–322] over the years and the different approaches have been implemented in NWCHEM. For an overview of the parallel plane-wave DFT implementation in NWCHEM, we refer the reader to a recent review.[202]

The very high amount of parallelism available on future exascale machines with many-core processors and/or separate GPU devices requires developers to carefully revisit the implementation of their programs in order to make use of this hardware efficiently. The exascale systems require the development of new hybrid algorithms to effectively utilize these machines – requiring an extensive redesign, and essentially complete rewrite, of the old codes. These new developments are currently being implemented as a part of NWChemEx. Over the last two years, we have been developing hybrid OpenMP-MPI and SYCL-MPI algorithms. The OpenMP-MPI algorithms were initially implemented in the NWCHEM program and are now being ported into NWCHEMEX. The SYCL-MPI algorithms are being directly implemented into NWChemEx.

Many of the details for the OpenMP-MPI hybrid algorithms can already be found in several recent papers al[48,322,323]. In Figure 4 the timing results for a full AIMD simulation of 256 water molecules on 16, 32, 64, 128, 256, and 1024 KNL nodes with 66 threads per node are shown. Leaving out two cores of each Xeon Phi processor during the computations is best for performance. The most likely explanation for the small degradation in performance when including the additional two cores is due to the overhead of the operating system and MPI processes running on the node. The "Cori" system at NERSC was used to run the benchmark, a Car–Parrinello simulations of 256 H_2O with an FFT grid of $N_g = 180^3$ (N_e=2056) using the plane-wave DFT module (PSPW) in NWChem. The size of this benchmark simulation is about 4 times larger than many mid-size AIMD simulations carried out in recent years, e.g., in recent work by Bylaska et al.[324–329]. The overall timings show strong scaling up to 1024 KNL nodes (69632 cores) and the timings of the major kernels, the pipelined 3D FFTs, non-local pseudopotential, and Lagrange multiplier kernels all displayed significant speedups.

As part of the NWCHEMEX development, we are also building an infrastructure for a

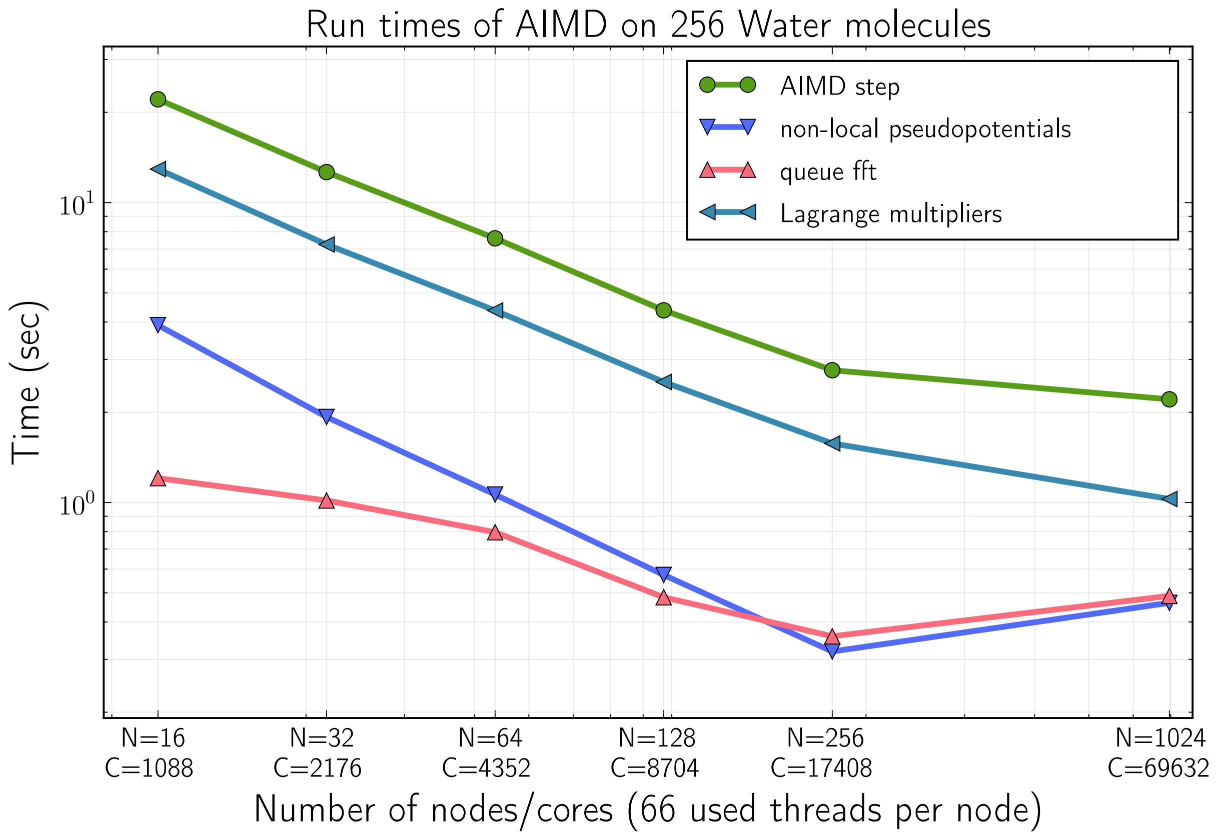


Figure 4: Scalability of major components of an AIMD step on the Xeon Phi partition for "water256". Figure from Bylaska et al.[48].

completely new $O(N)$ AIMD code. In this proposed development, we are generally following the strategy of Fattenbert et al.[297,330]. However, we are using several different design choices compared with this prior work. The main difference in our development is that we are making more extensive use of 3D FFTs for interpolation and efficiency. Most notably, the iterative multi-grid Poisson solver is being replaced by a large parallel 3D FFT.

The overall data layout of the $O(N)$ AIMD algorithm is shown in Fig. 5. The overall space of the simulation is described by the density grid. The ψ patches, depicted as yellow grids, are used to describe the space for each of the localized wavefunctions. For each simulation, there are N_e ψ patches, i.e., a patch for each localized wavefunctions. During a AIMD simulation the patches are moved to follow the motion of the localized ψ.

The key parts to the $O(N)$ AIMD algorithm are as follows.

- *Use penalty functions* to keep ψ's constrained to patches (yellow grids).

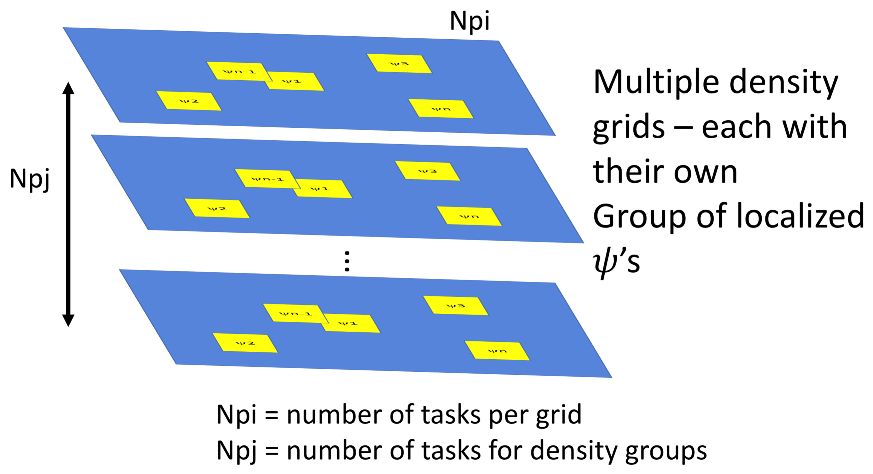

Npi
Npj
Multiple density grids – each with their own Group of localized ψ's
ψn-1
ψ1
ψ2
ψ3
ψn
Npi = number of tasks per grid
Npj = number of tasks for density groups

Figure 5: Multiple density grids where subgroups of $\{\psi\}$ are stored on different density grids.

- *The non-local pseudopotential, V_{NL}, and the kinetic energy operator, ∇^2*, are localized operators and are computed on the patches. These require FFTs on yellow grids for accurate interpolation or, alternatively, a real-space form can be used.

- *Enforcing orthogonality*, requires the evaluation of overlap matrices between different ψ and iteration algorithms that require sparse matrix multiply.

- *Density operations*, i.e., V_{xc} and V_H, and local pseudopotentials, V_{local}, are evaluated on the blue grids. Requires 3D FFTs across blue grids.

- *Exact exchange operator*, requires the evaluation of 3D FFTs over overlap densities. These algorithms are following the prior work which used an incomplete butterfly algorithm, except that the incomplete butterfly algorithm is being generalized to be carried out across multiple density grids.

- *Multiple density grids* are used. To handle load balancing for simulations where there are large regions of space that do not contain ψ, e.g., slab simulations, we generalize the computational space to have multiple density grids, where each grid has its own group of localized ψ (see Fig. 5).

During each iteration of an AIMD simulation in this scheme, N_e KS orbitals, $\psi(r, 1 : N_e)$, are converted from real space to reciprocal space and N_e orbital gradients (i.e. kinetic energy

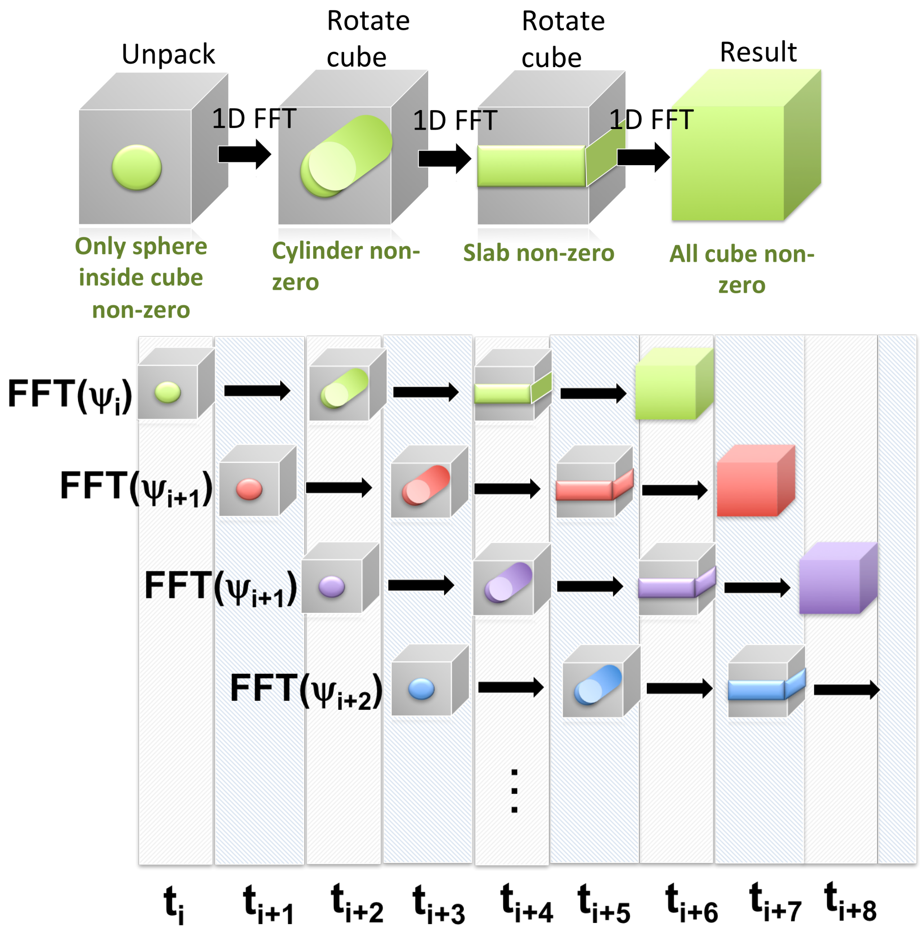


Figure 6: (Top) Illustration of computational steps in a specialized 3D FFT used to build a patch 3D FFT. (Bottom) Illustration of the pipelined 3D FFT algorithm used in the AIMD code.

and non-local pseudopotential operators) are transformed from real space to reciprocal space to real space. This corresponds to computing N_e reverse 3D FFTs and N_e forward 3D FFTs on the patch grids (i.e., yellow grids).

Each parallel 3D FFT [304,323,331–333] consists of six distinct steps, each of which is executed for each of the N_e KS orbitals in a pipelined fashion, as illustrated in Figure 6. For the forward 3D FFT, the steps are (in reverse order for backward FFTs):

1. Unpack the reciprocal space sphere into a 3D cube, where the leading dimension of the cube is the z-direction, second dimension is the x-direction, and the third dimension is the y-direction, i.e., cube stored as z, x, y.

2. Perform $nx \times ny$ 1D FFTs along the z-direction. Note that only the arrays that intersect the sphere need to be computed.

3. Rotate the cube so that the first dimension is the y-direction, $z, x, y \rightarrow y, z, x$.

4. Perform $nz \times nx$ 1D FFTs along the y-direction.

5. Rotate the cube so that the first dimension is the x-direction, $y, z, x \rightarrow x, y, z$.

6. Perform $ny \times nz$ 1D FFTs along the x-direction.

Note that the 1^{st} step in the above algorithm could be eliminated, however, reducing the reciprocal space to a sphere substantially improves the parallelization of the overall 3D FFT.

In developing the patch 3D FFTs, the 3D FFT algorithms contained in the NWChem plane-wave module (called NWPW) have been modified. These algorithms have already been generalized to use a hybrid MPI-OpenMP model where the planes of 1D FFTs in steps 2, 4, and 6 execute on multiple threads through an OpenMP `DO` directive so that a single 1D FFT is carried out on one thread. The data rearrangement in steps 1, 3, and 5 is threaded using a `DO` directive on the loops that perform the data-copying on the node. These algorithms will be generalized to use the GPU-MPI programming model in the near future. There have also been recent developments for 3D FFT algorithms using GPUs that show promise and may be end up being used in our developments[334]. A key aspect of these algorithms is that they are implemented using core point to point communication routines, i.e., non-blocking MPI primitives. More details on the implementation of these FFTs can be found in prior work by Bylaska et al.[92,304,320].

These 3D FFT algorithms are transformed to patch grids by using a mapping function for the patch grids to a standardized grid in which the point to point communications have been defined. Implementation of these algorithms is relatively straightforward, although the old code contained in NWChem was not usable for this development because it does not allow a dynamic number of instances (the overhead of Fortran 77 made this not practical).

10 Coupled-Cluster

Various CC approximations[335–341] have evolved into the working engines of high-accuracy computational chemistry and have been implemented in most of the electronic structure codes. The success of the CC theory in capturing electron correlation effects originates in:

- Size-consistency of the theory associated with the connected character of diagrams contributing to the CC equations. This property allows one to correctly describe chemical reactions and dissociation processes.
- Higher-order excitations can be approximated as products of lower-rank cluster operators.
- Systematic, and rapid, convergence of the properties with the excitation rank of the cluster operator.

The perturbative nature of the CC formalism also allows one to incorporate higher-order clusters using various perturbative techniques. Undoubtedly, the best-known example falling into this category is the ubiquitous CCSD(T) formalism[342].

Two key factors limit the usefulness of accurate CC methods to practical problems of interest to experimentalists:

- First, the rapid (high-order polynomial) growth of the computational cost with the molecular size N. Namely, CCSD,[338] CC with singles, doubles, and triples (CCSDT),[343–345] CC with singles, doubles, triples and quadruples (CCSDTQ),[346,347] have asymptotic operations costs proportional to $n_o^2 n_u^4$, $n_o^3 n_u^5$, and $n_o^4 n_u^6$, respectively (where n_o and n_u denote the number of correlated occupied and unoccupied orbitals, respectively, and grow in proportion to the system size N), whereas the cost of the non-iterative (T) contribution in CCSD(T) is proportional to $n_o^3 n_u^4$. Hence, doubling the system size increases the computational expense of even the simplest CC methods by more than two orders of magnitude;
- Second, the slow basis set convergence of the correlation energy. To reduce the basis set error of correlation energy to chemically-relevant levels large basis sets are required

(100+ basis functions/atom), and the asymptotic rate of convergence to the basis set limit is very slow: halving the basis set error increases the computational cost by more than one order of magnitude.

The high computational complexity of the modern wave function methods can be alleviated somewhat by taking advantage of modern distributed-memory parallel computers such as pioneered by the original NWChem project. Nevertheless, the development of parallel algorithms must be accompanied by fundamental improvements of these methods since even a million-fold concurrency would only allow an increase of the system size by a factor of ten or so. Thus to make the NWChemEx project transformative, rather than incremental, it is mandatory to address both of these key issues.

To reduce the cost and complexity of CC methods NWChemEx pursues several directions, namely by compressing the Hamiltonian via Cholesky-based tensor factorization (see Section 10.1) and by local compression of the cluster operators in the pair natural orbital framework (see Sections 10.1 and 10.2). Furthermore, the basis-set problem of CC methods is addressed by developing the explicitly correlated CC formulation therefore (see Section 10.3).

10.1 Cholesky-decomposition-based CC formulations

Since the CCSD(T) approach was a target of numerous development efforts, special attention has been paid to reducing the memory requirements associated with the storage of the most memory demanding tensor in the CCSD(T) workflow representing two-electron integrals. Among several tensor decomposition techniques, two of them – Cholesky decomposition[348–351] and Density Fitting[352–358]– have assumed special position in the development of CC implementations[224,358–367] based on the approximate representation of two-electron integrals. These methods also open alternative ways for optimizing the operation count, data representation, and communication of the parallel codes, which is especially true for the iterative CCSD formalism (being an intermediate step in the CCSD(T) calculation). In NWChemEx we enable a CCSD(T) implementation using Cholesky decomposition for atomic two-electron integrals $(\mu\nu|\rho\sigma)$

$$(\mu\nu|\rho\sigma) = \sum_K L^K_{\mu\nu} L^K_{\rho\sigma} \,, \tag{16}$$

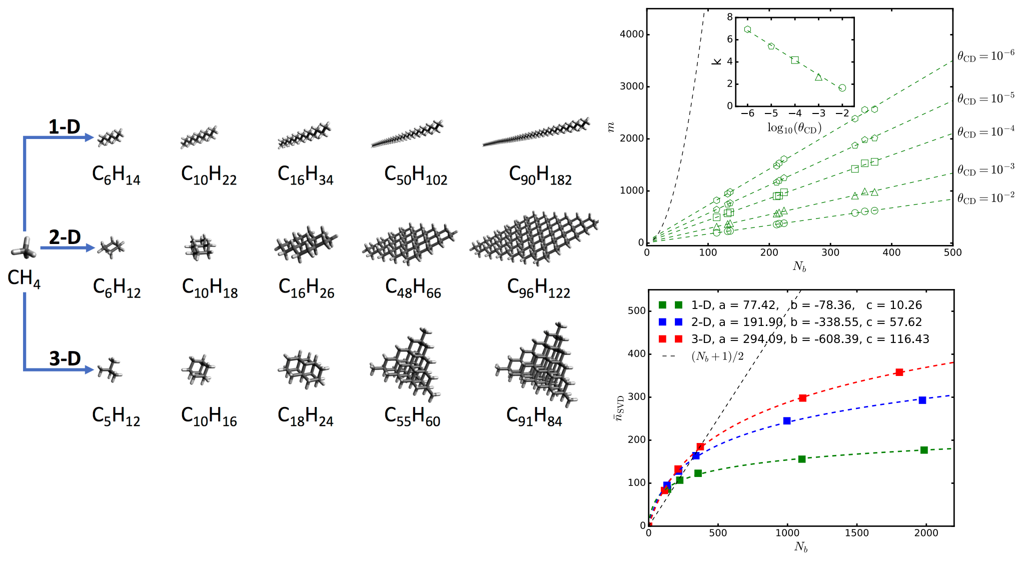


Figure 7: Structures of carbon–hydrogen systems chosen for the benchmark test of the low-rank compound decomposition procedure of the two-electron integral tensor, the linear relationship between the number of Cholesky vectors , m, and the number of basis functions, N_b, and the average number of singular vectors per Cholesky vector ($\bar{n}_{SVD}$).

where $L^K_{\mu\nu}$ (or L^K_{pq} in the molecular orbital basis) are the Cholesky vectors. To further compress two-electron integrals, we have also tested an approach where Cholesky vectors are further compressed using Singular Value Decomposition (SVD).[368] The combined CD-SVD decomposition was validated on 1-, 2-, and 3-dimensional systems showing a significant reduction of storage requirements for two-electron integrals from $O(N_b^4)$ to $N_b^2 log(N_b)$ (see Fig. 7) without a significant loss in accuracy for ground-state energies, excitation energies, and non-linear optical properties (see Ref.[368] and Fig. 8 for details) – where N_b is the number of basis functions.

A significant effort was expended to optimize the algebraic form of the CCSD-CD equations using TAMM functionalities. The main difference between canonical and Cholesky-decomposition-based equations is the fact that in the latter case a larger number of contractions between low-rank tensors are involved. In contrast to the canonical formulation, expressions in the CD-based parameterizations can be further factorized and parallelized due to the fact that large batches of CD or CD-SVD vectors can be stored locally, which significantly reduces the inter-node communications. Additionally, imposing a tile structure on all indices including orbital/spin-orbital (`[i]`,`[j]`,...,`[a]`,`[b]`,...) and Cholesky (`[K]`) leads to significant

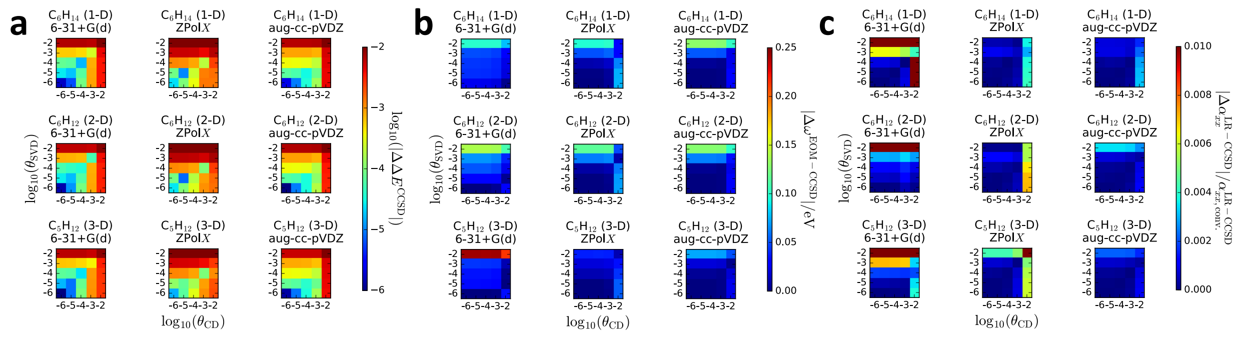


Figure 8: Deviations of ground-state correlation energies (a), excitation energies for first excited state (b), and static polarizabilities (c) calculated at the CCSD level for the selected 1-D, 2-D, 3-D carbon–hydrogen systems along with the change of θ_{CD} and θ_{SVD}.

improvements in the performance of the code (especially, when the size of the [K] tiles is significantly bigger than the sizes of molecular tiles).

10.2 Reduced-Scaling CC methods based on Pair Natural Orbitals

As shown in the previous Section, it is possible to reduce the cost of the CCSD method by compression of the Hamiltonian, however, reduction of the operation complexity requires compressing the basis used to express the cluster operator. Any such reduction of the computational complexity takes advantage of the rapid decay of many-electron correlations with distances in most chemical situations; truncation of these interactions can be done in several ways.

1. One group of approaches truncates the many-body expansion (MBE) of the energy, which is a sum of one-, two-, and higher-body components:

$$E = \sum_{A} E_A + \sum_{A<B} \Delta E_{AB} + \sum_{A<B<C} \Delta E_{ABC} + \ldots \tag{17}$$

   This series converges quickly if classical long-range interactions (e.g., electrostatic polarization) are renormalized into one-body energies by performing a self-consistent calculation on the entire system and if the fragments are chosen larger than the quantum correlation length of the system. The many approaches based on truncated variants of Eq. (17) differ in the manner of renormalization, fragment definition, and levels of theory; however, they all share an essential technical trait that (after renormalization) computations on sets of fragments are independent of one another (hence suitable for coarse-grain parallelization of the work) and utilize standard electronic structure approaches. We will only mention the fragment molecular orbital method,[369] the incremental correlation scheme[370] and the cluster-in-molecules approach.[371] The reader is referred to several reviews for more information.[372,373] Evaluation of properties with these approaches is also possible by differentiating the truncated MBE with respect to the perturbation parameter.

2. Another group of approaches truncates the operators and wave functions directly by representing them in a form that reveals their sparse structure. In molecular applications, this means using spatially localized basis sets (AOs, localized MOs, and finite/spectral elements) or grids. The oldest approaches of this kind were the "local correlation" methods of Pulay,[374] first implemented by Saebo and Pulay.[375,376] The original ansatz used the non-orthogonal projected atomic orbitals to represent unoccupied states. Its large-scale use has been made possible by the work of Werner, Schütz and co-workers[377,378] who have developed and refined an efficient implementation of the concept. Other unoccupied orbital bases have been considered: pair-natural orbitals (PNOs) investigated by Edmiston, Krauss, Meyer, Ahlrichs, Kutzelnigg, Neese and others,[379–386] orbital-specific virtuals (OSVs) of Chan and Manby,[387] and localized virtual orbitals of Jørgensen.[388,389] Another group of approaches expresses many-body methods exclusively or primarily in the AO basis and exploit the resulting sparsity to attain reduced complexity; these methods were pioneered by Almlöf[390] and developed further by Scuseria, Ochsenfeld, and others.[391,392] Lastly, an initial $\mathcal{O}(N^3)$ implementation of MP2 and CC2 based on adaptive spectral element-based representation was recently described by Bischoff, Valeev, and co-workers.[113,393]

Most important recent development of accurate reduced/linear-scaling CC methods have employed PNOs[385,386] and other closely related concepts.[387,394,395] Although PNOs date back to the 60s and 70s and the work of Edmiston and Krauss,[380,396] Meyer,[381,382,397] Ahlrichs and Kutzelnigg,[384,398–400] and others, their recent use was popularized by the work of Neese and co-workers,[385,386] who showed that they become competitive with canonical methods rather early. PNO-based compression combined with local ("domain") formalism (which alone is sufficient for linear scaling as shown by Werner and co-workers[359,401–403]) gives rise to reduced[404,405] and even linear scaling[394,395,406–411,411–413] variants of the PNO CC methods, becoming competitive with canonical CC implementations for systems with as few as 10-20 atoms. These dramatic developments have prompted us to make the domain-based PNO CC methods a major focus of the NWChemEx project.

The PNOs of pair ij are the eigenvectors of the pair-specific density $\mathbf{D}^{ij}$:

$$\mathbf{D}^{ij}\mathbf{U}^{ij} = \mathbf{U}^{ij}\mathbf{n}^{ij}, \tag{18}$$

with $(\mathbf{U}^{ij})_{ba} \equiv U_{ba_{ij}}$ the coefficient of MO b in PNO a_{ij}, and $(\mathbf{n}^{ij})_{ab} \equiv n_{a_{ij}}\delta_{a_{ij}b_{ij}}$ the corresponding occupancy.[1] The pair density matrix,

$$\mathbf{D}^{ij} = \frac{1}{1+\delta_{ij}}\left(\tilde{\mathbf{T}}^{ij\dagger}\mathbf{T}^{ij} + \tilde{\mathbf{T}}^{ij}\mathbf{T}^{ij\dagger}\right), \tag{19}$$

is defined from the 2-body amplitudes $\mathbf{T}^{(ij)}$, where $(\mathbf{T}^{ij})_{ab} \equiv T^{ij}_{ab}$ and $\tilde{T}^{ij}_{ab} = 2T^{ij}_{ab} - T^{ij}_{ba}$. To realize computational savings only those PNOs are kept for which $n_{a_{ij}} \geq \tau_{\text{PNO}}$, where the truncation threshold, τ_{PNO}, is a user-defined model parameter (setting $\tau_{\text{PNO}} = 0$ makes the PNO-based representation exact). For a finite (nonzero) τ_{PNO} the number of PNOs per pair is independent of the system size, thus using the *truncated* PNO basis reduces the number of amplitudes $\left(\bar{\mathbf{T}}^{ij}\right)_{a_{ij}b_{ij}}$ from $\mathcal{O}(N^4)$ to $\mathcal{O}(N^2)$. 1-body amplitudes $\bar{\mathbf{T}}^{i}$ are similarly expressed in a truncated basis of OSVs $\mathbf{U}^{i}$, typically taken to be PNOs of pair ii truncated with tighter threshold than the PNOs themselves.

The quality of the PNO representation is controlled by truncation parameter τ_{PNO} *and* the type of guess 2-body amplitudes used to define the PNOs. Several types of guess amplitudes have been tried originally;[380–382,384,396,397] the standard approach nowadays is to compute PNOs from first-order Møller–Plesset (MP1) amplitudes,[385] defined in the canonical MO

[1]Following convention, we have used i, j, . . . ; a, b, . . . ; and p, q, . . . for the occupied, virtual, and general orbitals in the HF basis, respectively.

basis as

$$T^{ij}_{ab} = \frac{g^{ij}_{ab}}{f^i_i + f^j_j - f^a_a - f^b_b}, \tag{20}$$

with $f^p_q = \langle p| \hat{f} |q\rangle$ the matrix elements of the Fock operator (often localized occupied orbitals are used directly in eq. (20), resulting in so-called as semicanonical MP1 amplitudes)[385]. This approach has also been generalized by Tew and co-workers in the context of explicitly correlated methods.[414] Recently it was shown that by iterative optimization of PNOs in coupled-cluster methods the PNO truncation error can be greatly reduced.[225] Similarly, PNOs can be computed purely numerically.[415]

The main challenge of domain-local pair natural orbital (DLPNO) methods is the need for high-performance algebra of block-sparse and hierarchical tensors. Whereas in conventional CC theory one computes with ordinary dense tensors like t^{ij}_{ab} and g^{ab}_{cd} and optionally with *static* block-sparse structure due to spin, geometric, and other symmetries, in PNO CC methods one computes with more complex tensorial data $t^{ij}_{a_{ij}b_{ij}}$ and $g^{a_{ij}b_{ij}}_{c_{ij}d_{ij}}$, which can be represented in a number of ways (e.g., as regular sparse tensors, or hierarchical block-sparse tensors), and, importantly, with their sparse structure determined by *dynamical* factors such as properties of the particular molecular system, variations between iterations, etc. Not only do DLPNO methods present novel tensorial data structures, but the equations themselves are different from those of the traditional CC and they involve new types of operations (e.g., Hadamard products). The NWChemEx framework will include support for efficient parallel implementation of reduced-scaling PNO MP2, CCSD, and CCSD(T) energies.

10.3 Explicitly correlated CC methods

The pursuit of predictive simulation of electronic structure involves controlling both the *accuracy* (e.g., the Hamiltonian approximation and the wave function ansatz) as well as *precision*, namely, the numerical errors that arise due to discretization of the CC equations. The numerical error of many-body methods like CCSD(T) in their standard formalism, unfortunately, suffers from the slow asymptotic decay due to the singularity of the Coulomb electron-electron interaction, and the resulting cusps in the electronic wave function whenever two electrons meet.[416–418] This translates into rapid asymptotic growth of the computational cost with the desired precision; most importantly, the use of small basis sets results in unacceptably large errors for key properties like reaction energetics, vibrational frequencies, and spin-spin couplings. To make reduced-scaling many-body methods a viable alternative to DFT it is mandatory to reduce their basis set errors. This is accomplished most robustly by building in the cusp-like structure into the wave function via terms dependent on the interelectronic distances. Such *explicitly correlated* formalisms, specifically in the form of the R12/F12 methods,[419–421] greatly reduce the basis set error by typically 2 cardinal numbers of the correlation consistent basis set family. The NWChemEx framework will include support for the F12 variants of the standard and the reduced-scaling (DLPNO) variants of the CC methods.

11 Classical Molecular Dynamics

The generation of extended molecular time-trajectories is a principal and general challenge for biomolecular modeling and simulation, and advanced simulation methodologies need to be used to implement algorithms that enable the capture of dynamical, energetic and thermodynamical features central to molecular research.[422] The molecular-level processes to be understood through the application of computer simulation include protein folding, protein docking, complex enzymatic reactions, and the association and function of large protein and protein-DNA complexes. The common challenge is the need for sufficiently long simulation times for sufficiently large ensembles of conformations to capture the relevant events. Using MD simulation codes on very large processor count heterogeneous and accelerated computer architectures requires an extensive analysis of the algorithms used and the formulation of an implementation strategy that leads to significant performance improvements.[423]

The most efficient implementation of MD is based on a domain decomposition approach. Domain decomposition has been shown to be the most efficient approach on distributed-memory systems and is being used by most major classical simulation packages. Physical space is decomposed into rectangular cells that are assigned to one of the available process ranks. A main difference among the multiple MD packages is in how the inter-cell interactions are communicated and computed and how the work is load-balanced while the chemical system is moving in real space. In NWChem, two methods are implemented to dynamically optimize load-balance between processes. First, the 'ownership' of a cell-pair can dynamically switch from the busiest to the less busy rank. Second, the physical space of the computationally most intensive cell can be dynamically made smaller, thereby increasing the size of neighboring, less computationally intensive cells. The second method typically requires redistribution of atoms between ranks with associated communication, and is only applied if the first method of cell-pair distribution no longer leads to improved load-balance.[424,425].

The MD implementation in NWChem requires each process rank used in a simulation to be assigned at least one cell in the domain decomposition. While this a good approach for large molecular systems, it limits the use of large numbers of processes for moderately sized chemical systems. The implications for parallel efficiency, communication requirements, and load-balancing options of a new approach based on the distribution of the cell-cell pair list was analyzed as the design target for NWChemEx.[426]

In NWChemEx, a new data structure has been designed to maximally exploit locality and reduction of communication and memory requirements.[427] The computational workload depends on the distribution based on relative orientation of cells within the physical decomposition. Duplication of cell-cell pairs in our current prototype allows the use of a number of processes up to 42 times the number of cells in the system. The most important challenge for scalability of classical MD is the need for synchronization after the evaluation of forces and after updating the coordinates. Large process counts make the use of explicit global synchronizations very inefficient, and redesigned kernels without explicit synchronization are required for computational efficiency. All synchronizations can be made implicit by having processes wait only for expected neighbor contributions before continuing. Such a local synchronization approach effectively improves scalability by avoiding the causes of time inefficiencies in global reduction and synchronization operations. This feature can be accomplished in part by using GA features in a recently developed put-notify mode allowing all communication to be expressed using a push-data model instead of the traditional pull-data model and was shown to result in an order of magnitude better scalability of the implemented MD simulation module in the ARGOS[428] code (a prototype for NWChemEx) compared to

NWChem.[426]

The chosen approach is being implemented in a MD kernel for use on modern heterogeneous accelerated architectures.[429] Work is ongoing to implement further improvements and optimizations of intra-node parallelism, topology-aware data structure assignment, and data communication protocols that reduce movement of data needed by multiple processes on a node only once to that node. Polling mechanisms for each of the processes will be used to determine the availability of the data. The implementation will make extensive use of features of the GA toolkit. In addition, data communication will be hidden behind computation through the use of a small number (typically three) dedicated processes to handle control tasks, such as global accumulation of kinetic energy and virial for constant temperature and pressure simulations as well as the collection of data for trajectory and property recording during simulations runs. A prototype MD kernel has been extensively instrumented for detailed wall clock time analysis of all components in MD time steps, including communication, computation, and implicit synchronization times. Such analysis details assist in identifying opportunities to reduce load imbalance, avoid communication cost, and improve parallelism. The utilization of the data-centric capabilities enabled by the GA/ARMCI (Aggregate Remote Memory Copy Interface) PGAS programming model provides unique opportunities to address the primary challenges for parallel scalability of MD time-stepping algorithms. The one-sided asynchronous communication operations provided by the ARMCI communication model and the global, data-centric view of GA allow for an intuitive way to avoid both global and local explicit synchronous operations and to effectively manage concurrent communication and computation.

12 Embedding Methods

When dealing with moderate to large chemical systems requiring chemical accuracy, an accurate theoretical treatment of the entire system with *ab initio* wavefunction approaches can become computationally prohibitive very quickly. Embedding or hybrid approaches, where a smaller chemically relevant active region is treated with a higher-level wavefunction approach in a background environment represented at a lower-level approximation, can mitigate this issue.

Over the years, several embedding strategies have been developed at different levels of complexity, such as: ONIOM[430], electrostatic embedding[431], quantum mechanics/molecular mechanics (QM/MM)[432,433], fragment methods[372], density-functional-based embedding[434–438], density embedding (DET)[439], density matrix embedding (DMET)[440], projector-based embedding[441–443], embedded mean-field theory[444], self-energy[445] and Green's function embedding.[446–448] For a general overview of embedding approaches we refer the reader to numerous reviews that have been published on the subject.[372,431,449–453]

All embedding approaches rely on partitioning the full system into subsystems and a definition of the energy. For two subsystems, A embedded in the environment of B, one can, within the language of DFT, write the formal DFT-in-DFT embedding energy expression as[435],

$$E^{full}_{DFT-in-DFT} = E^{full}_{DFT} + (\widetilde{E}^{A}_{DFT} - \widetilde{E}^{A}_{DFT}) \tag{21}$$

and, similarly, the wavefunction (WF)-in-DFT embedding energy as

$$E^{full}_{WF-in-DFT} = E^{full}_{DFT} + (\widetilde{E}^{A}_{WF} - \widetilde{E}^{A}_{DFT}) \tag{22}$$

where E^{full}_{DFT} is the DFT energy for the full system and $\widetilde{E}^{A}_{DFT}$ and $\widetilde{E}^{A}_{WF}$ are the embedded energies of subsystem A at the DFT and WF levels of theory, respectively. Clearly, the correction in parenthesis to the full DFT energy cancels for same, and is non-zero for different, exchange-correlation functionals in Eq.21, while the correction is almost always non-zero with a wavefunction method as in Eq.22.

Depending on the nature of the electronic structure of the system, partitioning can become strongly system dependent. This is especially the case if the subsystem partitioning involves strongly bonded, i.e., covalently bonded, environments. From a quantum embedding standpoint, this imposes another stringent requirement, namely, orthogonality between the subsystems.

The projector-based embedding approach, which has its roots in the pioneering work of Phillips and Kleinman[306] and Huzinaga and Cantu[454,455] on the formal separability of many-electron systems, has been explored with renewed interest recently.[442,443] Within this approach, the partitioning is achieved as projections between subspaces instead of physical partitioning, which ensures the orthogonality requirement. In addition, it also provides a natural way to generalize the embedding problem to multiple systems, represented at different levels of theory. This approach has been further explored and developed with great success in recent years both in molecular and periodic systems, with the main difference being the choice of the projection operator, namely, the Huzinaga or level-shift-based projection operators.[442,443,454–456] Another key component of the projection approach involves the decomposition of the orbital spaces, where different promising schemes have been developed and is an active area of research.[438,442,443,457–461]

In NWChemEx, we are developing a general and scalable embedding framework to support the projector-based embedding approach including orbital partitioning, so that different levels of electronic structure theories may be combined as well as other embedding variants like quantum-classical embedding.

13 Conclusions and Future Directions

As exascale machines get closer to existence, hardware is growing more complex (multiple layers of memory, accelerators in addition to CPUs, etc.) in a way that has a direct effect on programmers as they seek to efficiently use the resources on the machine (for example, fully taking advantage of any available high bandwidth memory). This review article addresses some of the challenges for addressing these complex hardware and associated software issues. The NWChemEx project, as an exemplar of the constantly changing computational landscape, has been designed for flexibility and performance to respond to these challenges through reduced-scaling methods, careful design, OOP conventions, modularity and abstractions, the SDE framework, and solid software engineering practices. Capabilities like unit testing, software review, and checkpoint/restart have been added into the framework and software engineering practices from the beginning to ensure that NWChemEx has a solid base from which to build upon.

The emergence of next-generation exascale architectures offers a unique opportunity to address outstanding and previously unobtainable chemical challenges with methodologies that describe electron correlation effects and reduce dependence on basis set choice. Novel reduced-scaling techniques that take advantage of the local character of correlation effects and efficiently utilize massively parallel architectures through innovative programming tools will significantly reshape the landscape of high-accuracy simulations. The design and implementation of such methods and algorithms requires a program suite under which these approaches and tools can be designed, and NWChemEx strives to be at the forefront of this development. NWChemEx will offer the possibility of performing simulations with an unprecedented level of accuracy for systems several orders of magnitude larger than systems tractable by canonical formulations of theoretical methods. This transition requires redefinition, redesign, and extension of parallel tensor libraries to cope with challenges posed by efficient exascale implementations of sparse tensors contractions that underlie reduced-scaling CC formulations. To achieve this goal, TA and TAMM are used to enable facile development of the algorithms and to provide performance on CPUs and GPUs for local versions of HF, DFT, and CC methods.

The arrival of exascale systems like the Argonne Leadership Computing Facility (ALCF) Aurora computer or the Oak Ridge Leadership Computing Facility (OLCF) Frontier machine, which are expected to arrive in the 2021-2023 time frame will allow for the performance analysis and validation of current and future progress of NWChemEx. Moving forward, NWChemEx will include continued development of reduced-scaling methods coupled with potential energy surface sampling methods to enable highly accurate calculations of both enthalpy and entropy effects. In addition, future work will also involve incorporating community standards and interfacing with other software development projects for property type APIs and additional functionality into the code.

NWChemEx will significantly shift the envelope for systems-size limits tractable by high-accuracy methods. In the near-time perspective, NWChemEx will focus on select classes of methodologies mainly related to the reduced-scale approaches. Once the sparse infrastructure matures, deploying new formulations and extension to excited-state problems, strongly correlated methods, properties calculations, and molecular dynamics should be much easier and less time consuming. We envision and anticipate that the full transition to NWChemEx and emplacement of a rich environment of electronic structure methods in NWChem will occur over the next decade. In this intermediate period, we envisage the co-existence of NWChem and NWChemEx according to the equation:

$$(1-\lambda)\text{NWChem} + \lambda\text{NWChemEx} \; . \tag{23}$$

The extent of the transition period is contingent upon the external support, engagement of a broad computational chemistry community, and availability of exascale architectures.

Acknowledgement

The authors thank Edoardo Aprà for his extensive conversations about DFT design and implementation issues. This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. This work used resources of the OLCF at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. This research used resources of the ALCF, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. This research also used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231. Funding for the development of NWChem and parallel software tools over the lifetime of the project include Environmental and Molecular Sciences Laboratory (EMSL), the Construction Project and Operations and the Office of Biological and Environmental Research. This research also benefited from computational resources at EMSL, a DOE Office of Science User Facility sponsored by the Office of Biological and Environmental Research and located at the Pacific Northwest National Laboratory (PNNL). PNNL is operated by Battelle Memorial Institute for the United States Department of Energy under DOE contract DE-AC05-76RL1830.

References

1. The National Strategic Computing Initiative. `https://www.nitrd.gov/nsci/`, 2015; [Online; accessed 6-August-2020].

2. Kothe, D.; Lee, S.; Qualters, I. Exascale Computing in the United States. *Comput. Sci. Eng.* **2019**, *21*, 17–29.

3. Alexander, F.; Almgren, A.; Bell, J.; Bhattacharjee, A.; Chen, J.; Colella, P.; Daniel, D.; DeSlippe, J.; Diachin, L.; Draeger, E. et al. Exascale applications: skin in the game. *Phil. Trans. R. Soc. A* **2020**, *378*, 20190056.

4. Bernholdt, D.; Apra, E.; Früchtl, H.; Guest, M.; Harrison, R.; Kendall, R.; Kutteh, R.; Long, X.; Nicholas, J.; Nichols, J. et al. Parallel Computational Chemistry Made Easier: The Development of NWChem. *Int. J. Quant. Chem.* **1995**, *56*, 475–483.

5. Anchell, J.; Apra, E.; Bernholdt, D.; Borowski, P.; Clark, T.; Clerc, D.; Dachsel, H.; Deegan, M.; Dupuis, M.; Dyall, K. et al. NWChem, Version 3.2. High Performance Computational Chemistry Group, Pacific Northwestern National Laboratory, Richland, WA, 1998.

6. Kendall, R. A.; Apra, E.; Bernholdt, D. E.; Bylaska, E. J.; Dupuis, M.; Fann, G. I.; Harrison, R. J.; Ju, J.; Nichols, J. A.; Nieplocha, J. et al. High Performance Computational Chemistry: An Overview of NWChem a Distributed Parallel Application. *Comput. Phys. Commun.* **2000**, *128*, 260–283.

7. Harrison, R.; Nichols, J.; Straatsma, T.; Dupuis, M.; Bylaska, E.; Fann, G.; Windus, T.; Apra, E.; Anchell, J.; Bernholdt, D. et al. NWChem, A Computational Chemistry Package for Parallel Computers, version 4.1. Pacific Northwest National Laboratory, Richland, Washington, 2000.

8. Straatsma, T. P.; Apra, E.; Windus, T.; Bylaska, E.; de Jong, W.; Hirata, S.; Valiev, M.; Hackler, M.; Pollack, L.; Harrison, R. et al. NWChem, A Computational Chemistry Package for Parallel Computers, Version 4.6 (2004). Pacific Northwest National Laboratory, Richland, Washington.

9. Apra, E.; Windus, T.; Straatsma, T. P.; Bylaska, E.; de Jong, W.; Hirata, S.; Valiev, M.; Hackler, M.; Pollack, L.; Kowalski, K. et al. NWChem, A Computational Chemistry Package for Parallel Computers, Version 4.7. Pacific Northwest National Laboratory, Richland, Washington, 2005.

10. Bylaska, E.; De Jong, W.; Govind, N.; Kowalski, K.; Straatsma, T. P.; Valiev, M.; Wang, D.; Apra, E.; Windus, T.; Hammond, J. et al. NWChem, A Computational Chemistry Package for Parallel Computers, Version 5.1. Pacific Northwest National Laboratory, Richland, Washington, 2007.

11. Valiev, M.; Bylaska, E.; Govind, N.; Kowalski, K.; Straatsma, T. P.; Van Dam, H.; Wang, D.; Nieplocha, J.; Apra, E.; Windus, T. et al. NWChem: A Comprehensive and Scalable Open-Source Solution for Large Scale Molecular Simulations. *Comput. Phys. Commun.* **2010**, *181*, 1477–1489.

12. Schmidt, M.; Baldridge, K.; Boatz, J.; Elbert, S.; Gordon, M.; Jensen, J.; Koseki, S.; Matsunaga, N.; Nguyen, K.; Su, S. et al. General Atomic and Molecular Electronic Structure System GAMESS. *J. Comp. Chem.* **1993**, *11*, 1347–1363.

13. Plimpton, S. Fast Parallel Algorithms for Short-Range Molecular Dynamics. *J. Comput. Phys.* **1995**, *117*, 1–19.

14. Kim, J.; Baczewski, A. D.; Beaudet, T. D.; Benali, A.; Bennett, M. C.; Berrill, M. A.; Blunt, N. S.; Borda, E. J. L.; Casula, M.; Ceperley, D. M. et al. QMCPACK: An Open Source ab initio Quantum Monte Carlo Package for the Electronic Structure of Atoms, Molecules and Solids. *J. Phys.: Condens. Matter* **2018**, *30*, 195901.

15. McKone, T. E.; Nazaroff, W. W.; Berck, P.; Auffhammer, M.; Lipman, T.; Torn, M. S.; Masanet, E.; Lobscheid, A.; Santero, N.; Mishra, U. et al. Grand Challenges for Life-Cycle Assessment of Biofuels. *Environ. Sci. Technol.* **2011**, *45*, 1751–1756.

16. Goddard, W. A.; Dunning, T. H.; Hunt, W. J.; Hay, P. J. Generalized Valence Bond Description of Bonding in Low-Lying States of Molecules. *Acc. Chem. Res.* **1973**, *6*, 368–376.

17. Lischka, H.; Müller, T.; Szalay, P. G.; Shavitt, I.; Pitzer, R. M.; Shepard, R. Columbus – A Program System for Advanced Multireference Theory Calculations. *WIREs Comput. Mol. Sci.* **2011**, *1*, 191–199.

18. Guest, M. F.; Bush, I. J.; Dam, H. J. J. V.; Sherwood, P.; Thomas, J. M. H.; Lenthe, J. H. V.; Havenith, R. W. A.; Kendrick, J. The GAMESS-UK Electronic Structure Package: Algorithms, Developments and Applications. *Mol. Phys.* **2005**, *103*, 719–747.

19. Dupuis, M.; Watts, J.; Villar, H.; Hurst, G. The General Atomic and Molecular Electronic Structure System Hondo: Version 7.0. *Comput. Phys. Commun.* **1989**, *52*, 415–425.

20. Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Scalmani, G.; Barone, V.; Petersson, G. A.; Nakatsuji, H. et al. Gaussian 16 Revision B.01. 2016; Gaussian Inc. Wallingford CT.

21. Aquilante, F.; Autschbach, J.; Carlson, R. K.; Chibotaru, L. F.; Delcey, M. G.; De Vico, L.; Fdez. Galván, I.; Ferré, N.; Frutos, L. M.; Gagliardi, L. et al. Molcas 8: New Capabilities for Multiconfigurational Quantum Chemical Calculations Across the Periodic Table. *J. Comput. Chem.* **2016**, *37*, 506–541.

22. Werner, H.-J.; Knowles, P. J.; Knizia, G.; Manby, F. R.; Schütz, M. Molpro: A General-Purpose Quantum Chemistry Program Package. *WIREs Comput. Mol. Sci.* **2012**, *2*, 242–253.

23. Balasubramani, S. G.; Chen, G. P.; Coriani, S.; Diedenhofen, M.; Frank, M. S.; Franzke, Y. J.; Furche, F.; Grotjahn, R.; Harding, M. E.; Hättig, C. et al. TURBOMOLE: Modular Program Suite for ab initio Quantum-Chemical and Condensed-Matter Simulations. *J. Chem. Phys.* **2020**, *152*, 184107.

24. Perera, A.; Bartlett, R. J.; Sanders, B. A.; Lotrich, V. F.; Byrd, J. N. Advanced Concepts in Electronic Structure (ACES) Software Programs. *J. Chem. Phys.* **2020**, *152*, 184105.

25. te Velde, G.; Bickelhaupt, F. M.; Baerends, E. J.; Fonseca Guerra, C.; van Gisbergen, S. J. A.; Snijders, J. G.; Ziegler, T. Chemistry with ADF. *J. Comput. Chem.* **2001**, *22*, 931–967.

26. Clark, S. J.; Segall, M. D.; Pickard, C. J.; Hasnip, P. J.; Probert, M. I. J.; Refson, K.; Payne, M. C. First Principles Methods Using CASTEP. *Z. Kristallogr. Cryst. Mater.* **2005**, *220*, 567–570.

27. CPMD. `https://www.cpmd.org/wordpress/`, 2020; [Online; accessed 23-July-2020].

28. Salahub, D.; Goursot, A.; Weber, J.; Koster, A.; Vela, A. In *Theory and Applications of Computational Chemistry: The First 40 Years. A Volume of Technical and Historical Perspectives*; Dykstra, C. E., Frenking, G., Kim, K., Scuseria, G. E., Eds.; Elsevier, 2005; Chapter Applied Density Functional Theory and the deMon Codes: 1964-2004.

29. Kresse, G.; Furthmüller, J. Efficient Iterative Schemes for ab initio Total-Energy Calculations Using a Plane-wave Basis Set. *Phys. Rev. B* **1996**, *54*, 11169–11186.

30. Case, D. A.; Cheatham III, T. E.; Darden, T.; Gohlke, H.; Luo, R.; Merz Jr., K. M.; Onufriev, A.; Simmerling, C.; Wang, B.; Woods, R. J. The Amber Biomolecular Simulation Programs. *J. Comput. Chem.* **2005**, *26*, 1668–1688.

31. Brooks, B. R.; Bruccoleri, R. E.; Olafson, B. D.; States, D. J.; Swaminathan, S.; Karplus, M. CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations. *J. Comput. Chem.* **1983**, *4*, 187–217.

32. Rackers, J. A.; Wang, Z.; Lu, C.; Laury, M. L.; Lagardère, L.; Schnieders, M. J.; Piquemal, J.-P.; Ren, P.; Ponder, J. W. Tinker 8: Software Tools for Molecular Design. *J. Chem. Theory Comput.* **2018**, *14*, 5273–5289.

33. MolSSI CMS Software Database. `https://molssi.org/software-search/`, 2020; [Online; accessed 23-July-2020].

34. Wikipedia Category: Computational chemistry software. `https://en.wikipedia.org/wiki/Category:Computational_chemistry_software`, 2020; [Online; accessed 23-July-2020].

35. Hutter, J.; Iannuzzi, M.; Schiffmann, F.; VandeVondele, J. CP2K: Atomistic Simulations of Condensed Matter Systems. *WIREs Comput. Mol. Sci.* **2014**, *4*, 15–25.

36. Bowers, K. J.; Chow, D. E.; Xu, H.; Dror, R. O.; Eastwood, M. P.; Gregersen, B. A.; Klepeis, J. L.; Kolossvary, I.; Moraes, M. A.; Sacerdoti, F. D. et al. Scalable Algorithms for Molecular Dynamics Simulations on Commodity Clusters. SC '06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing. 2006; pp 43–43.

37. Berendsen, H.; van der Spoel, D.; van Drunen, R. GROMACS: A Message-Passing Parallel Molecular Dynamics Implementation. *Comput. Phys. Commun.* **1995**, *91*, 43–56.

38. Peng, C.; Lewis, C. A.; Wang, X.; Clement, M. C.; Pierce, K.; Rishi, V.; Pavošević, F.; Slattery, S.; Zhang, J.; Teke, N. et al. Massively Parallel Quantum Chemistry: A High-Performance Research Platform for Electronic Structure. *J. Chem. Phys.* **2020**, *153*, 044120.

39. Shao, Y.; Gan, Z.; Epifanovsky, E.; Gilbert, A. T.; Wormit, M.; Kussmann, J.; Lange, A. W.; Behn, A.; Deng, J.; Feng, X. et al. Advances in Molecular Quantum Chemistry Contained in the Q-Chem 4 Program Package. *Mol. Phys.* **2015**, *113*, 184–215.

40. García, A.; Papior, N.; Akhtar, A.; Artacho, E.; Blum, V.; Bosoni, E.; Brandimarte, P.; Brandbyge, M.; Cerdá, J. I.; Corsetti, F. et al. Siesta: Recent Developments and Applications. *J. Chem. Phys.* **2020**, *152*, 204108.

41. Nieplocha, J.; Harrison, R. J.; Littlefield, R. J. Global Arrays: A Portable "Shared-memory" Programming Model for Distributed Memory Computers. Proceedings of the 1994 ACM/IEEE Conference on Supercomputing. Los Alamitos, CA, USA, 1994; pp 340–349.

42. Bachega, L.; Siddhartha Chatterjee,; Dockser, K. A.; Gunnels, J. A.; Manish Gupta,; Gustavson, F. G.; Lapkowski, C. A.; Liu, G. K.; Mendell, M. P.; Wait, C. D. et al. A High-Performance SIMD Floating Point Unit for BlueGene/L: Architecture, Compilation, and Algorithm Design. Proceedings. 13th International Conference on Parallel Architecture and Compilation Techniques, 2004. PACT 2004. 2004; pp 85–96.

43. Gschwind, M.; Hofstee, H. P.; Flachs, B.; Hopkins, M.; Watanabe, Y.; Yamazaki, T. Synergistic Processing in Cell's Multicore Architecture. *IEEE Micro* **2006**, *26*, 10–24.

44. Streitz, F. H.; Glosli, J. N.; Patel, M. V.; Chan, B.; Yates, R. K.; de Supinski, B. R.; Sexton, J.; Gunnels, J. A. 100 + TFlop Solidification Simulations on BlueGene / L. Proceedings of IEEE/ACM Supercomputing. 2005.

45. Gygi, F.; Yates, R. K.; Lorenz, J.; Draeger, E. W.; Franchetti, F.; Ueberhuber, C. W.; de Supinski, B. R.; Kral, S.; Gunnels, J. A.; Sexton, J. C. Large-Scale First-Principles Molecular Dynamics Simulations on the BlueGene/L Platform Using the Qbox Code. SC '05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing. 2005; pp 24–24.

46. Sodani, A.; Gramunt, R.; Corbal, J.; Kim, H.; Vinod, K.; Chinthamani, S.; Hutsell, S.; Agarwal, R.; Liu, Y. Knights Landing: Second-Generation Intel Xeon Phi Product. *IEEE Micro* **2016**, *36*, 34–46.

47. Dagum, L.; Menon, R. OpenMP: An Industry Standard API for Shared-Memory Programming. *CSE-M, IEEE* **1998**, *5*, 46–55.

48. Bylaska, E. J.; Jacquelin, M.; de Jong, W. A.; Hammond, J. R.; Klemm, M. Performance Evaluation of NWChem Ab-Initio Molecular Dynamics (AIMD) Simulations on the Intel© Xeon Phi™ Processor. High Performance Computing. Cham, 2017; pp 404–418.

49. Shan, H.; Williams, S.; de Jong, W.; Oliker, L. Thread-Level Parallelization and Optimization of NWChem for the Intel MIC Architecture. Proceedings of the Sixth International Workshop on Programming Models and Applications for Multicores and Manycores. New York, NY, USA, 2015; pp 58–67.

50. Apra, E.; Klemm, M.; Kowalski, K. Efficient Implementation of Many-body Quantum Chemical Methods on the Intel Xeon Phi Coprocessor. SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2014; pp 674–684.

51. Huang, H.; Chow, E. Accelerating Quantum Chemistry with Vectorized and Batched Integrals. Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis. 2018; p 1–14.

52. Mironov, V.; Moskovsky, A.; D'Mello, M.; Alexeev, Y. An Efficient MPI/OpenMP Parallelization of the Hartree–Fock–Roothaan Method for the First Generation of Intel© Xeon Phi™Processor Architecture. *Int. J. High Perform. Comput. Appl.* **2019**, *33*, 212–224.

53. Leang, S. S.; Rendell, A. P.; Gordon, M. S. Quantum Chemical Calculations Using Accelerators: Migrating Matrix Operations to the NVIDIA Kepler GPU and the Intel Xeon Phi. *J. Chem. Theory Comput.* **2014**, *10*, 908–912.

54. Owens, J. D.; Houston, M.; Luebke, D.; Green, S.; Stone, J. E.; Phillips, J. C. GPU Computing. *Proceedings of the IEEE* **2008**, *96*, 879–899.

55. Aamodt, T. M.; Fung, W. W. L.; Rogers, T. G.; Martonosi, M. *General-Purpose Graphics Processor Architecture*; 2018.

56. Walker, R. C.; W.Götz, A. *Electronic Structure Calculations on Graphics Processing Units: From Quantum Chemistry to Condensed Matter Physics*; 2016.

57. Tornai, G. J.; Ladjánszki, I.; Rák, A.; Kis, G.; Cserey, G. Calculation of Quantum Chemical Two-Electron Integrals by Applying Compiler Technology on GPU. *J. Chem. Theory Comput.* **2019**, *15*, 5319–5331.

58. Ma, W.; Krishnamoorthy, S.; Villa, O.; Kowalski, K. GPU-Based Implementations of the Noniterative Regularized-CCSD(T) Corrections: Applications to Strongly Correlated Systems. *J. Chem. Theory Comput.* **2011**, *7*, 1316–1327.

59. Kim, J.; Sukumaran-Rajam, A.; Hong, C.; Panyala, A.; Srivastava, R. K.; Krishnamoorthy, S.; Sadayappan, P. Optimizing Tensor Contractions in CCSD(T) for Efficient Execution on GPUs. Proceedings of the 2018 International Conference on Supercomputing. New York, NY, USA, 2018; pp 96–106.

60. DePrince, A. E.; Hammond, J. R.; Gray, S. K. Many-Body Quantum Chemistry on Graphics Processing Units. 2011; `https://www.mcs.anl.gov/uploads/cels/papers/scidac11/final/deprince_eugene.pdf`, [Online; accessed 30-January-2021].

61. Asadchev, A.; Gordon, M. S. New Multithreaded Hybrid CPU/GPU Approach to Hartree–Fock. *J. Chem. Theory Comput.* **2012**, *8*, 4166–4176.

62. Yasuda, K. Accelerating Density Functional Calculations with Graphics Processing Unit. *J. Chem. Theory Comput.* **2008**, *4*, 1230–1236.

63. Yoshikawa, T.; Komoto, N.; Nishimura, Y.; Nakai, H. GPU-Accelerated Large-Scale Excited-State Simulation Based on Divide-and-Conquer Time-Dependent Density-Functional Tight-Binding. *J. Comput. Chem.* **2019**, *40*, 2778–2786.

64. Genovese, L.; Ospici, M.; Deutsch, T.; Méhaut, J.-F.; Neelov, A.; Goedecker, S. Density Functional Theory Calculation on Many-Cores Hybrid Central Processing Unit-Graphic Processing Unit Architectures. *J. Chem. Phys.* **2009**, *131*, 034103.

65. Asadchev, A.; Allada, V.; Felder, J.; Bode, B. M.; Gordon, M. S.; Windus, T. L. Uncontracted Rys Quadrature Implementation of up to G Functions on Graphical Processing Units. *J. Chem. Theory Comput.* **2010**, *6*, 696–704.

66. DePrince, A. E.; Hammond, J. R. Coupled Cluster Theory on Graphics Processing Units I. The Coupled Cluster Doubles Method. *J. Chem. Theory Comput.* **2011**, *7*, 1287–1295.

67. Snyder, J. W.; Fales, B. S.; Hohenstein, E. G.; Levine, B. G.; Martínez, T. J. A Direct-Compatible Formulation of the Coupled Perturbed Complete Active Space Self-Consistent Field Equations on Graphical Processing Units. *J. Chem. Phys.* **2017**, *146*, 174113.

68. Ufimtsev, I. S.; Martínez, T. J. Graphical Processing Units for Quantum Chemistry. *Comput. Sci. Eng.* **2008**, *10*, 26–34.

69. CUDA Toolkit. `https://developer.nvidia.com/cuda-toolkit`, [Online; accessed 30-August-2020].

70. OpenCL. `https://www.khronos.org/opencl/`, 2020; [Online; accessed 22-July-2020].

71. HIP Programming Guide. `https://rocmdocs.amd.com/en/latest/Programming_Guides/HIP-GUIDE.html`, 2020; [Online; accessed 22-July-2020].

72. Edwards, H. C.; Trott, C. R.; Sunderland, D. Kokkos: Enabling Manycore Performance Portability Through Polymorphic Memory Access Patterns. *J. Parallel Distrib. Comput.* **2014**, *74*, 3202–3216.

73. RAJA Performance Portability Layer. `https://github.com/LLNL/RAJA`, 2020; [Online; accessed 22-July-2020].

74. Shiozaki, T. BAGEL: Brilliantly Advanced General Electronic-structure Library. *WIREs Comput. Mol. Sci.* **2018**, *8*, e1331.

75. Anderson, J. A.; Glaser, J.; Glotzer, S. C. HOOMD-blue: A Python Package for High-Performance Molecular Dynamics and Hard Particle Monte Carlo Simulations. *Comput. Mater. Sci.* **2020**, *173*, 109363.

76. Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kalé, L.; Schulten, K. Scalable Molecular Dynamics with NAMD. *J. Comput. Chem.* **2005**, *26*, 1781–1802.

77. Eastman, P.; Swails, J.; Chodera, J.; McGibbon, R.; Zhao, Y.; Beauchamp, K.; Wang, L.-P.; Simmonett, A.; Harrigan, M.; Stern, C. et al. OpenMM 7: Rapid Development of High Performance Algorithms for Molecular Dynamics. *PLOS Comp. Biol.* **2017**, *13*, e1005659.

78. Neese, F. The ORCA Program System. *WIREs Comput. Mol. Sci.* **2012**, *2*, 73–78.

79. Bonomi, M.; Branduardi, D.; Bussi, G.; Camilloni, C.; Provasi, D.; Raiteri, P.; Donadio, D.; Marinelli, F.; Pietrucci, F.; Broglia, R. A. et al. PLUMED: A Portable Plugin

for Free-Energy Calculations with Molecular Dynamics. *Comput. Phys. Commun.* **2009**, *180*, 1961–1972.

80. Parrish, R. M.; Burns, L. A.; Smith, D. G. A.; Simmonett, A. C.; DePrince, A. E.; Hohenstein, E. G.; Bozkaya, U.; Sokolov, A. Y.; Di Remigio, R.; Richard, R. M. et al. Psi4 1.1: An Open-Source Electronic Structure Program Emphasizing Automation, Advanced Libraries, and Interoperability. *J. Chem. Theory Comput.* **2017**, *13*, 3185–3197.

81. Sun, Q.; Berkelbach, T. C.; Blunt, N. S.; Booth, G. H.; Guo, S.; Li, Z.; Liu, J.; McClain, J. D.; Sayfutyarova, E. R.; Sharma, S. et al. PySCF: The Python-Based Simulations of Chemistry Framework. *WIREs Comput. Mol. Sci.* **2018**, *8*, e1340.

82. Gygi, F. Architecture of Qbox: A Scalable First-Principles Molecular Dynamics Code. *IBM J. Res. Dev.* **2008**, *52*, 137–144.

83. Giannozzi, P.; Baroni, S.; Bonini, N.; Calandra, M.; Car, R.; Cavazzoni, C.; Ceresoli, D.; Chiarotti, G. L.; Cococcioni, M.; Dabo, I. et al. QUANTUM ESPRESSO: A Modular and Open-Source Software Project for Quantum Simulations of Materials. *J. Phys.: Condens. Matter* **2009**, *21*, 395502.

84. Dongarra, J.; Beckman, P.; Moore, T.; Aerts, P.; Aloisio, G.; Andre, J.-C.; Barkai, D.; Berthou, J.-Y.; Boku, T.; Braunschweig, B. et al. The International Exascale Software Project Roadmap. *Int. J. High Perform. Comput. Appl.* **2011**, *25*, 3–60.

85. Cociorva, D.; Wilkins, J.; Baumgartner, G.; Sadayappan, P.; Ramanujam, J.; Nooijen, M.; Bernholdt, D.; Harrison, R. Towards Automatic Synthesis of High-Performance Codes for Electronic Structure Calculations: Data Locality Optimization. High Performance Computing — HiPC 2001. Berlin, Heidelberg, 2001; pp 237–248.

86. Kerbyson, D.; Vishnu, A.; Barker, K.; Hoisie, A. Codesign Challenges for Exascale Systems: Performance, Power, and Reliability. *Computer* **2011**, *44*, 37–43.

87. Lee, J.; Sun, J.; Peterson, G.; Harrison, R.; Hinde, R. Power-aware Performance of Mixed Precision Linear Solvers for FPGAs and GPGPUs. Application Accelerators in High Performance Computing, 2010 Symposium, Papers. 2010.

88. Fought, E. L.; Sundriyal, V.; Sosonkina, M.; Windus, T. L. Saving Time and Energy with Oversubscription and Semi-direct Møller–Plesset Second Order Perturbation Methods. *J. Comput. Chem.* **2017**, *38*, 830–841.

89. Fought, E. L.; Sundriyal, V.; Sosonkina, M.; Windus, T. L. Improving Efficiency of Semi-direct Møller–Plesset Second-order Perturbation Methods Through Oversubscription on Multiple Nodes. *J. Comput. Chem.* **2019**, *40*, 2146–2157.

90. Phillips, J. C.; Hardy, D. J.; Maia, J. D. C.; Stone, J. E.; Ribeiro, J. V.; Bernardi, R. C.; Buch, R.; Fiorin, G.; HÃ©nin, J.; Jiang, W. et al. Scalable Molecular Dynamics on CPU and GPU Architectures with NAMD. *J. Chem. Phys.* **2020**, *153*, 044130.

91. Jain, N.; Bohm, E.; Mikida, E.; Mandal, S.; Kim, M.; Jindal, P.; Li, Q.; Ismail-Beigi, S.; Martyna, G.; Kale, L. OpenAtom: Scalable Ab-Initio Molecular Dynamics with Diverse Capabilities. International Supercomputing Conference. 2016.

92. de Jong, W. A.; Bylaska, E.; Govind, N.; Janssen, C. L.; Kowalski, K.; Müller, T.; Nielsen, I. M. B.; van Dam, H. J. J.; Veryazov, V.; Lindh, R. Utilizing High Performance Computing for Chemistry: Parallel Computational Chemistry. *Phys. Chem. Chem. Phys.* **2010**, *12*, 6896–6920.

93. Schüler, M.; Kovar, T.; Lischka, H.; Shepard, R.; Harrison, R. J. A Parallel Implementation of the COLUMBUS Multireference Configuration Interaction Program. *Theor. Chim. Acta* **1993**, *84*, 489 – 509.

94. Lischka, H.; Dachsel, H.; Shepard, R.; Harrison, R. J. Parallel Computing in Quantum Chemistry — Message Passing and Beyond for a General ab initio Program System. *Future Gener. Comput. Syst.* **1995**, *11*, 445 – 450.

95. Wong, A. T.; Harrison, R. J. Approaches to Large-Scale Parallel Self-Consistent Field Calculations. *J. Comput. Chem.* **1995**, *16*, 1291–1300.

96. Richard, R. M.; Bertoni, C.; Boschen, J. S.; Keipert, K.; Pritchard, B.; Valeev, E. F.; Harrison, R. J.; de Jong, W. A.; Windus, T. L. Developing a Computational Chemistry Framework for the Exascale Era. *Comput. Sci. Eng.* **2019**, *21*, 48–58.

97. van Dam, H. J. J.; Vishnu, A.; de Jong, W. A. A Case for Soft Error Detection and Correction in Computational Chemistry. *J. Chem. Theory Comput.* **2013**, *9*, 3995–4005.

98. Vishnu, A.; Van Dam, H.; De Jong, W.; Balaji, P.; Song, S. Fault-Tolerant Communication Runtime Support for Data-Centric Programming Models. 2010 International Conference on High Performance Computing. 2010; pp 1–9.

99. van Dam, H. J. J.; Vishnu, A.; de Jong, W. A. Designing a Scalable Fault Tolerance Model for High Performance Computational Chemistry: A Case Study with Coupled Cluster Perturbative Triples. *J. Chem. Theory Comput.* **2011**, *7*, 66–75.

100. Meneses, E.; Ni, X.; Zheng, G.; Mendes, C. L.; Kalé, L. V. Using Migratable Objects to Enhance Fault Tolerance Schemes in Supercomputers. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *26*, 2061–2074.

101. Hogan, S.; Hammond, J. R.; Chien, A. A. An Evaluation of Difference and Threshold Techniques for Efficient Checkpoints. Proceedings of IEEE Dependable Systems and Networks Workshops (DSN-W). Boston, MA, 2012; pp 1–6.

102. VELOC. `https://github.com/ECP-VeloC/VELOC`, 2018; [Online; accessed 29-August-2020].

103. Nicolae, B.; Moody, A.; Gonsiorowski, E.; Mohror, K.; Cappello, F. Veloc: Towards High Performance Adaptive Asynchronous Checkpointing at Large Scale. *Proceedings - 2019 IPDPS* **2019**, 911–920.

104. Bochevarov, A. D.; Harder, E.; Hughes, T. F.; Greenwood, J. R.; Braden, D. A.; Philipp, D. M.; Rinaldo, D.; Halls, M. D.; Zhang, J.; Friesner, R. A. Jaguar: A High-Performance Quantum Chemistry Software Program with Strengths in Life and Materials Sciences. *Int. J. Quant. Chem.* **2013**, *113*, 2110–2142.

105. Baker, J.; Wolinski, K.; Malagoli, M.; Kinghorn, D.; Wolinski, P.; Magyarfalvi, G.; Saebo, S.; Janowski, T.; Pulay, P. Quantum Chemistry in Parallel with PQS. *J. Comput. Chem.* **2009**, *30*, 317–335.

106. Seritan, S.; Bannwarth, C.; Fales, B. S.; Hohenstein, E. G.; Isborn, C. M.; Kokkila-Schumacher, S. I. L.; Li, X.; Liu, F.; Luehr, N.; Snyder Jr., J. W. et al. TeraChem: A Graphical Processing Unit-Accelerated Electronic Structure Package for Large-Scale ab initio Molecular Dynamics. *WIREs Comput. Mol. Sci.* e1494.

107. Geudtner, G.; Calaminici, P.; Carmona-Espíndola, J.; del Campo, J. M.; Domínguez-Soria, V. D.; Moreno, R. F.; Gamboa, G. U.; Goursot, A.; Köster, A. M.; Reveles, J. U. et al. deMon2k. *WIREs Comput. Mol. Sci.* **2012**, *2*, 548–555.

108. Muller, R. P. PyQuante: Python Quantum Chemistry. https://sourceforge.net/projects/pyquante/, [Online; accessed 27-January-2021].

109. Gonze, X.; Amadon, B.; Anglade, P.-M.; Beuken, J.-M.; Bottin, F.; Boulanger, P.; Bruneval, F.; Caliste, D.; Caracas, R.; Côté, M. et al. ABINIT: First-Principles Approach to Material and Nanosystem Properties. *Comput. Phys. Commun.* **2009**, *180*, 2582–2615.

110. Deumens, E.; Lotrich, V. F.; Perera, A.; Ponton, M. J.; Sanders, B. A.; Bartlett, R. J. Software Design of Aces III with the Super Instruction Architecture. *WIREs Comput. Mol. Sci.* **2011**, *1*, 895–901.

111. Ratcliff, L. E.; Dawson, W.; Fisicaro, G.; Caliste, D.; Mohr, S.; Degomme, A.; Videau, B.; Cristiglio, V.; Stella, M.; D'Alessandro, M. et al. Flexibilities of Wavelets as a Computational Basis Set for Large-Scale Electronic Structure Calculations. *J. Chem. Phys.* **2020**, *152*, 194110.

112. Aidas, K.; Angeli, C.; Bak, K. L.; Bakken, V.; Bast, R.; Boman, L.; Christiansen, O.; Cimiraglia, R.; Coriani, S.; Dahle, P. et al. The Dalton Quantum Chemistry Program System. *WIREs Comput. Mol. Sci.* **2014**, *4*, 269–284.

113. Harrison, R. J.; Beylkin, G.; Bischoff, F. A.; Calvin, J. A.; Fann, G. I.; Fosso-Tande, J.; Galindo, D.; Hammond, J. R.; Hartman-Baker, R.; Hill, J. C. et al. MADNESS: A Multiresolution, Adaptive Numerical Environment for Scientific Simulation. *SIAM J. Sci. Comput.* **2016**, *38*, S123–S142.

114. Tancogne-Dejean, N.; Oliveira, M. J. T.; Andrade, X.; Appel, H.; Borca, C. H.; Le Breton, G.; Buchholz, F.; Castro, A.; Corni, S.; Correa, A. A. et al. Octopus, a Computational Framework for Exploring Light-Driven Phenomena and Quantum Dynamics in Extended and Finite Systems. *J. Chem. Phys.* **2020**, *152*, 124119.

115. Tichy, W. F. Design, Implementation, and Evaluation of a Revision Control System. Proceedings of the 6th International Conference on Software Engineering. Washington, DC, USA, 1982; p 58–67.

116. Grune, D. Concurrent Versions System, A Method for Independent Cooperation. Online, accessed 27-January-2021, 1986; `https://dickgrune.com/Books/Publications/Concurrent_Versions_System,_a_method_for_independent_cooperation.pdf`.

117. Collins-Sussman, B.; Fitzpatrick, B. W.; Pilato, C. M. *Version Control with Subversion*; O'Reilly Media, 2011.

118. Torvalds, L. Tech Talk: Linus Torvalds on git. Online, accessed 27-January-2021, 2007; `https://www.youtube.com/watch?v=4XpnKHJAok8&t=1m30s`.

119. Pool, M. Bazaar. Online, accessed 27-January-2021, 2005; `https://bazaar.canonical.com/en/`.

120. SourceForge. Online, accessed 27-January-2021, 1999; `https://sourceforge.net/`.

121. Preston-Werner, T.; Wanstrath, C.; Hyett, P. J.; Chacon, S. GitHub. Online, accessed 27-January-2021, 2008; `https://github.com/`.

122. Sijbrandij, S.; Zaporozhets, D. GitLab. Online, accessed 28-January-2021, 2014; `https://gitlab.com/`.

123. Launchpad. Online, accessed 28-January-2021, 2004; `https://launchpad.net/`.

124. CDash. Online, accessed 28-January-2021, `https://cdash.org`.

125. Travis CI. Online, accessed 28-January-2021, 2011; `https://travis-ci.com`.

126. CircleCI. Online, accessed 28-January-2021, 2011; `https://circleci.com`.

127. Sijbrandij, S.; Zaporozhets, D. GitLabCI. Online, accessed 28-January-2021, 2016; `https://gitlab.com`.

128. Preston-Werner, T.; Wanstrath, C.; Hyett, P. J.; Chacon, S. GitHub Actions CI. Online, accessed 28-January-2021, 2018; `https://github.com`.

129. Codecov – Develop healthier code. 2016; `https://www.codecov.io/`, [Online; accessed 7-May-2020].

130. Looks Good To Me (LGTM). Online, accessed 28-January-2021, 2007; `https://lgtm.com`.

131. d. Moor, O.; Verbaere, M.; Hajiyev, E.; Avgustinov, P.; Ekman, T.; Ongkingco, N.; Sereni, D.; Tibble, J. Keynote Address: .QL for Source Code Analysis. Seventh IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM 2007). 2007; pp 3–16.

132. van Heesch, D. Doxygen. 1997; `https://www.doxygen.nl`, [Online; accessed 7-May-2020].

133. Brandl, G. Sphinx – Python Documentation Generator. 2008; `https://www.sphinx-doc.org/`, [Online; accessed 7-May-2020].

134. Loper, E. Epydoc. Online, accessed 28-January-2021, 2002; `http://epydoc.sourceforge.net/`.

135. Holscher, E.; Grace, B.; Leifer, C. Read the Docs. Online, accessed 28-January-2021, 2010; `https://readthedocs.org/`.

136. van Weert, J. ROBODoc. Online, accessed 28-January-2021, 1994; `https://rfsber.home.xs4all.nl/Robo/index.html`.

137. Apache 2.0. `https://www.apache.org/licenses/LICENSE-2.0`, 2019; [Online; accessed 30-August-2020].

138. CMake. 2000; `https://cmake.org/`, [Online; accessed 7-May-2020].

139. CMakePP – Automating and Simplifying CMake Build Systems. 2020; `https://github.com/cmakepp`, [Online; accessed 7-May-2020].

140. Catch2. `https://github.com/catchorg/Catch2`, 2020; [Online; accessed 22-July-2020].

141. gcov – a Test Coverage Program. `https://gcc.gnu.org/onlinedocs/gcc/Gcov.html`, [Online; accessed 7-May-2020].

142. Hart, W.; Atkinson, L. Gcovr – generate GCC code coverage reports. 2013; `https://github.com/gcovr/gcovr/`, [Online; accessed 7-May-2020].

143. McDonald, A. R.; Nash, J. A.; Nerenberg, P. S.; Ball, K. A.; Sode, O.; Foley IV, J. J.; Windus, T. L.; Crawford, T. D. Building Capacity for Undergraduate Education and Training in Computational Molecular Science: A Collaboration Between the MERCURY Consortium and the Molecular Sciences Software Institute. *Int. J. Quant. Chem.* **2020**, *120*, e26359.

144. Michie, D. "Memo" Functions and Machine Learning. *Nature* **1968**, *218*, 19–22.

145. Ahuja,; Carriero,; Gelernter, Linda and Friends. *Computer* **1986**, *19*, 26–34.

146. Fletcher, G. D.; Schmidt, M. W.; Bode, B. M.; Gordon, M. S. The Distributed Data Interface in GAMESS. *Comput. Phys. Commun.* **2000**, *128*, 190 – 200.

147. Kalé, L. V. *Parallel Programming with CHARM: An Overview*; 1993.

148. Kalé, L. V.; Krishnan, S. CHARM++: A Portable Concurrent Object Oriented System Based on C++. Proceedings of the Eighth Annual Conference on Object-oriented Programming Systems, Languages, and Applications. 1993; pp 91–108.

149. Kalé, L. V.; Ramkumar, B.; Sinha, A. B.; Saletore, V. A. *The CHARM Parallel Programming Language and System: Part II – The Runtime system*; 1994.

150. Kim, M.; Mandal, S.; Mikida, E.; Chandrasekar, K.; Bohm, E.; Jain, N.; Li, Q.; Kanakagiri, R.; Martyna, G. J.; Kale, L. et al. Scalable GW Software for Quasiparticle Properties Using OpenAtom. *Comput. Phys. Commun.* **2019**, *244*, 427–441.

151. Borštnik, U.; VandeVondele, J.; Weber, V.; Hutter, J. Sparse Matrix Multiplication: The Distributed Block-Compressed Sparse Row Library. *Parallel Comput.* **2014**, *40*, 47 – 58.

152. Nieplocha, J.; Harrison, R.; Littlefield, R. The Global Array Programming Model for High Performance Scientific Computing. *SIAM News* **1995**, *28*, 12–14.

153. Nieplocha, J.; Harrison, R. J.; Littlefield, R. J. Global Arrays: A Nonuniform Memory Access Programming Model for High-Performance Computers. *J. Supercomput.* **1996**, *10*, 169–189.

154. Nieplocha, J.; Harrison, R. J. Shared Memory NUMA programming on I-WAY. Proceedings of 5th IEEE International Symposium on High Performance Distributed Computing. 1996; pp 432–441.

155. Dachsel, H.; Nieplocha, J.; Harrison, R. An Out-Of-Core Implementation of the COLUMBUS Massively-Parallel Multireference Configuration Interaction Program. SC'98: Proceedings of the 1998 ACM/IEEE Conference on Supercomputing. 1998; pp 41–41.

156. Nieplocha, J.; Tipparaju, V.; Saify, A.; Panda, D. Protocols and Strategies for Optimizing Performance of Remote Memory Operations on Clusters. Proceedings of the 16th International Parallel and Distributed Processing Symposium. p 275.

157. Dinan, J.; Balaji, P.; Hammond, J. R.; Krishnamoorthy, S.; Tipparaju, V. Supporting the Global Arrays PGAS Model Using MPI One-Sided Communication. 2012 IEEE 26th International Parallel and Distributed Processing Symposium. 2012; pp 739–750.

158. Shah, G.; Nieplocha, J.; Mirza, J.; Kim, C.; Harrison, R.; Govindaraju, R. K.; Gildea, K.; DiNicola, P.; Bender, C. Performance and Experience With LAPI-A New High-Performance Communication Library for the IBM RS/6000 SP. Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing. 1998; pp 260–266.

159. Fedorov, D. G.; Olson, R. M.; Kitaura, K.; Gordon, M. S.; Koseki, S. A New Hierarchical Parallelization Scheme: Generalized Distributed Data Interface (GDDI), and

an Application to the Fragment Molecular Orbital Method (FMO). *J. Comput. Chem.* **2004**, *25*, 872–880.

160. Wang, M.; May, A. J.; Knowles, P. J. Parallel Programming Interface for Distributed Data. *Comput. Phys. Commun.* **2009**, *180*, 2673 – 2679.

161. Vancoillie, S.; Delcey, M. G.; Lindh, R.; Vysotskiy, V.; Malmqvist, P.-A.; Veryazov, V. Parallelization of a Multiconfigurational Perturbation Theory. *J. Comput. Chem.* **2013**, *34*, 1937–1948.

162. Alpert, B.; Beylkin, G.; Gines, D.; Vozovoi, L. Adaptive Solution of Partial Differential Equations in Multiwavelet Bases. *J. Comput. Phys.* **2002**, *182*, 149–190.

163. Harrison, R.; Fann, G.; Yanai, T.; Beylkin, G. Multiresolution Quantum Chemistry in Multiwavelet Bases. International Conference on Computational Science. 2003; pp 103–110.

164. Pei, J. C.; Fann, G. I.; Harrison, R. J.; Nazarewicz, W.; Shi, Y.; Thornton, S. Adaptive Multi-resolution 3D Hartree-Fock-Bogoliubov Solver for Nuclear Structure. *Phys. Rev. C* **2014**, *90*, 024317.

165. Pei, J.; Stoitsov, M.; Fann, G.; Nazarewicz, W.; Schunck, N.; Xu, F. Deformed Coordinate-Space Hartree-Fock-Bogoliubov Approach to Weakly Bound Nuclei and Large Deformations. *Phy. Rev. C* **2008**, *78*, 064306.

166. Fann, G.; Pei, J.; Harrison, R. J.; Jia, J.; Ou, M.; Nazarewciz, W.; Schunck, N.; Shelton, W. A. Fast Multiresolution Methods for Density Functional Theory in Nuclear Physics. *Physics: Conference Series 180* **2009**, 012080.

167. Pei, J.; Fann, G.; Nazarewicz, W.; Harrison, R.; Hill, J.; Galindo, D.; Jia, J. Coordinate-Space Hartree-Fock-Bogoliubov for Superfluid Fermi Systems in Large Boxes. *J. Phys. Conf. Ser. 402* **2012**, 012035.

168. Reuter, M.; Hill, J.; Harrison, R. Solving PDEs in Irregular Geometries with Multiresolution Methods I: Embedded Dirichlet Boundary Conditions. *Comput. Phys. Commun.* **2012**, *183*, 1–7.

169. Harrison, R.; Fann, G.; Yanai, T.; Gan, Z.; Beylkin, G. Multiresolution Quantum Chemistry: Basic Theory and Initial Applications. *J. Chem. Phys.* **2004**, *121*, 11587–11598.

170. Yanai, T.; Fann, G.; Gan, Z.; Harrison, R.; Beylkin, G. Multiresolution Quantum Chemistry: Hartree-Fock Exchange. *J. Chem. Phys.* **2004**, *121*, 6680–6688.

171. Sekino, H.; Maeda, Y.; Yanai, T.; Harrison, R. J. Basis Set Limit Hartree-Fock and Density Functional Theory Response Property Evaluation by Multiresolution Multiwavelet Basis. *J. Chem. Phys.* **2008**, *129*, 18647020.

172. Thornton, W. Electronic Excitations in YTiO3 using TDDFT and Electronic Structure Using a Multiresolution Framework. Ph.D. thesis, University of Tennessee, Knoxville, 2011.

173. Vence, N.; Harrison, R.; Krstić, P. Attosecond Electron Dynamics: A Multiresolution Approach. *Phys. Rev. A* **2012**, *85*, 033403.

174. Baker Jr., H. G.; Hewitt, C. The Incremental Garbage Collection of Processes. Proceedings of the 1977 Symposium on Artificial Intelligence and Programming Languages. 1977; pp 55–59.

175. Blumofe, R. D.; Joerg, C. F.; Kuszmaul, B. C.; Leiserson, C. E.; Randall, K. H.; Zhou, Y. Cilk: An Efficient Multithreaded Runtime System. Proceedings of the Fifth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP). Santa Barbara, California, 1995; pp 207–216.

176. Blumofe, R. D.; Joerg, C. F.; Kuszmaul, B. C.; Leiserson, C. E.; Randall, K. H.; Zhou, Y. L. Cilk: An Efficient Multithreaded Runtime System. *J. Parallel Distrib. Comput.* **1996**, *37*, 55–69.

177. Threading Building Blocks. `https://www.threadingbuildingblocks.org`, [Online; accessed 30-August-2020].

178. Schmidt, D. C. The ADAPTIVE Communication Environment (ACE). `http://www.dre.vanderbilt.edu/~schmidt/ACE.html`, [Online; accessed 30-August-2020].

179. Schmidt, D. C.; Huston, S. D. *C++ Network Programming: Systematic Reuse with ACE and Frameworks, Vol. 2*; Pearson Education, 2002.

180. Yelick, K.; Bonachea, D.; Chen, W.-Y.; Colella, P.; Datta, K.; Duell, J.; Graham, S. L.; Hargrove, P.; Hilfinger, P.; Husbands, P. et al. Productivity and Performance Using Partitioned Global Address Space Languages. Proceedings of the 2007 international workshop on Parallel symbolic computation. 2007; pp 24–32.

181. Smith, D. G. A.; Burns, L. A.; Simmonett, A. C.; Parrish, R. M.; Schieber, M. C.; Galvelis, R.; Kraus, P.; Kruse, H.; Di Remigio, R.; Alenaizan, A. et al. PSI4 1.4: Open-source Software for High-Throughput Quantum Chemistry. *J. Chem. Phys.* **2020**, *152*, 184108.

182. A schema for Quantum Chemistry. `https://molssi-qc-schema.readthedocs.io/en/latest/`.

183. Grant, W. S.; Voorhies, R. Cereal. `https://github.com/USCiLab/cereal`, 2013; [Online; accessed 29-August-2020].

184. Pezoa, F.; Reutter, J. L.; Suarez, F.; Ugarte, M.; Vrgoč, D. Foundations of JSON Schema. Proceedings of the 25th International Conference on World Wide Web. 2016; pp 263–273.

185. XML standards for simulation data. `http://www.quantum-simulation.org`.

186. Murray-Rust, P.; Townsend, J. A.; Adams, S. E.; Phadungsukanan, W.; Thomas, J. The Semantics of Chemical Markup Language (CML): Dictionaries and Conventions. *J. Cheminf.* **2011**, *3*, 43.

187. Hanwell, M. D.; de Jong, W. A.; Harris, C. J. Open Chemistry: RESTful Web APIs, JSON, NWChem and the Modern Web Application. *J. Cheminf.* **2017**, *9*, 55.

188. Windus, T. L.; Pople, J. A. Pinnacle: An Approach Toward Object Oriented Quantum Chemistry. *Int. J. Quantum Chem., Symposium Proceedings* **1995**, *29*, 485–495.

189. Janssen, C. L.; Schaefer, H. F. The Automated Solution of Second Quantization Equations with Applications to the Coupled Cluster Approach. *Theor. Chim. Acta* **1991**, *79*, 1–42.

190. Li, X.; Paldus, J. Automation of the Implementation of Spin-Adapted Open-Shell Coupled-Cluster Theories Relying on the Unitary Group Formalism. *J. Chem. Phys.* **1994**, *101*, 8812–8826.

191. Kállay, M.; Surján, P. R. Higher Excitations in Coupled-Cluster Theory. *J. Chem. Phys.* **2001**, *115*, 2945–2954.

192. Nooijen, M.; Lotrich, V. Towards a General Multireference Coupled Cluster Method: Automated Implementation of Open-Shell CCSD Method for Doublet States. *J. Mol. Struct. THEOCHEM* **2001**, *547*, 253–267.

193. Nooijen, M. State Selective Equation of Motion Coupled Cluster Theory: Some Preliminary Results. *Int. J. Mol. Sci.* **2002**, *3*, 656–675.

194. Hirata, S. Tensor Contraction Engine: Abstraction and Automated Parallel Implementation of Configuration-Interaction, Coupled-Cluster, and Many-Body Perturbation Theories. *J. Phys. Chem. A* **2003**, *107*, 9887–9897.

195. Hirata, S. Symbolic Algebra in Quantum Chemistry. *Theor. Chem. Acc.* **2006**, *116*, 2–17.

196. Deumens, E.; Lotrich, V. F.; Perera, A. S.; Bartlett, R. J.; Jindal, N.; Sanders, B. A. *Annual Reports in Computational Chemistry*; Elsevier, 2011; Vol. 7; Chapter The Super Instruction Architecture: A Framework for High-Productivity Parallel Implementation of Coupled-Cluster Methods on Petascale Computers, pp 179–191.

197. Epifanovsky, E.; Wormit, M.; Kuś, T.; Landau, A.; Zuev, D.; Khistyaev, K.; Manohar, P.; Kaliman, I.; Dreuw, A.; Krylov, A. I. New Implementation of High-Level Correlated

Methods Using a General Block Tensor Library for High-Performance Electronic Structure Calculations. *J. Comput. Chem.* **2013**, *34*, 2293–2309.

198. Solomonik, E.; Matthews, D.; Hammond, J.; Demmel, J. Cyclops Tensor Framework: Reducing Communication and Eliminating Load Imbalance in Massively Parallel Contractions. 2013 IEEE 27th International Symposium on Parallel and Distributed Processing. 2013; pp 813–824.

199. Solomonik, E.; Matthews, D.; Hammond, J. R.; Stanton, J. F.; Demmel, J. A Massively Parallel Tensor Contraction Framework for Coupled-Cluster Computations. *J. Parallel Distrib. Comput.* **2014**, *74*, 3176–3190.

200. Matthews, D. A.; Stanton, J. F. Non-Orthogonal Spin-Adaptation of Coupled Cluster Methods: A New Implementation of Methods Including Quadruple Excitations. *J. Chem. Phys.* **2015**, *142*, 064108.

201. Calvin, J. A.; Lewis, C. A.; Valeev, E. F. Scalable Task-Based Algorithm for Multiplication of Block-Rank-Sparse Matrices. IA3 '15, The 5th Workshop on Irregular Applications: Architectures and Algorithms. New York, New York, USA, 2015; pp 1–8.

202. Aprà, E.; Bylaska, E. J.; de Jong, W. A.; Govind, N.; Kowalski, K.; Straatsma, T. P.; Valiev, M.; van Dam, H. J. J.; Alexeev, Y.; Anchell, J. et al. NWChem: Past, Present, and Future. *J. Chem. Phys.* **2020**, *152*, 184102.

203. Matthews, D. Aquarius: A Parallel Quantum Chemistry Package Built on the Cyclops Tensor Framework. `https://github.com/devinamatthews/aquarius`, [Online; accessed 30-August-2020].

204. Matthews, D. A.; Cheng, L.; Harding, M. E.; Lipparini, F.; Stopkowicz, S.; Jagau, T.-C.; Szalay, P. G.; Gauss, J.; Stanton, J. F. Coupled-Cluster Techniques for Computational Chemistry: The CFOUR Program Package. *J. Chem. Phys.* **2020**, *152*, 214108.

205. Kállay, M.; Nagy, P. R.; Mester, D.; Rolik, Z.; Samu, G.; Csontos, J.; Csóka, J.; Szabó, P. B.; Gyevi-Nagy, L.; Hégely, B. et al. The MRCC Program System: Accurate Quantum Chemistry From Water to Proteins. *J. Chem. Phys.* **2020**, *152*, 074107.

206. Baumgartner, G.; Auer, A.; Bernholdt, D. E.; Bibireata, A.; Choppella, V.; Cociorva, D.; Xiaoyang Gao,; Harrison, R. J.; Hirata, S.; Krishnamoorthy, S. et al. Synthesis of High-Performance Parallel Programs for a Class of Ab Initio Quantum Chemistry Models. *Proceedings of the IEEE* **2005**, *93*, 276–292.

207. Nieplocha, J.; Harrison, R. Shared Memory Programming in Metacomputing Environments: The Global Array Approach. *J. Supercomput.* **1997**, *11*, 119–136.

208. Kowalski, K.; Krishnamoorthy, S.; Olson, R. M.; Tipparaju, V.; Apra, E. Scalable Implementations of Accurate Excited-State Coupled Cluster Theories: Application of High-Level Methods to Porphyrin-Based Systems. Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. 2011; p 72.

209. Mutlu, E.; Kowalski, K.; Krishnamoorthy, S. Toward Generalized Tensor Algebra for ab initio Quantum Chemistry Methods. Proceedings of the 6th ACM SIGPLAN International Workshop on Libraries, Languages and Compilers for Array Programming. 2019; pp 46–56.

210. TAL-SH: Tensor Algebra Library Routines for Shared Memory Systems. `https://github.com/DmitryLyakh/TAL_SH`, 2020; [Online; accessed 21-June-2020].

211. Lyakh, D. I. An Efficient Tensor Transpose Algorithm for Multicore CPU, Intel Xeon Phi, and NVidia Tesla GPU. *Comput. Phys. Commun.* **2015**, *189*, 84–91.

212. Kim, J.; Sukumaran-Rajam, A.; Thumma, V.; Krishnamoorthy, S.; Panyala, A.; Pouchet, L.; Rountev, A.; Sadayappan, P. A Code Generator for High-Performance Tensor Contractions on GPUs. IEEE/ACM International Symposium on Code Generation and Optimization, CGO 2019, Washington, DC, USA, February 16-20, 2019. 2019; pp 85–95.

213. Lewis, C. A.; Calvin, J. A.; Valeev, E. F. Clustered Low-Rank Tensor Format: Introduction and Application to Fast Construction of Hartree–Fock Exchange. *J. Chem. Theory Comput.* **2016**, *12*, 5868–5880.

214. MPI Forum. `https://github.com/mpi-forum/`, 2020; [Online; accessed 22-July-2020].

215. Blackford, L. S.; Demmel, J.; Dongarra, J.; Duff, I.; Hammarling, S.; Henry, G.; Heroux, M.; Kaufman, L.; Lumsdaine, A.; Petitet, A. et al. An Updated Set of Basic Linear Algebra Subprograms (BLAS). *ACM Trans. Math. Softw.* **2002**, *28*, 135–151.

216. Guennebaud, G.; Jacob, B. Eigen v3. http://eigen.tuxfamily.org, 2010; [Online; accessed 30-August-2020].

217. Blackford, L. S.; Choi, J.; Cleary, A.; D'Azeuedo, E.; Demmel, J.; Dhillon, I.; Hammarling, S.; Henry, G.; Petitet, A.; Stanley, K. et al. In *ScaLAPACK User's Guide*; Dongarra, J. J., Ed.; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 1997.

218. BTAS. `https://github.com/BTAS/BTAS`, [Online; accessed 30-August-2020].

219. Hynninen, A.-P.; Lyakh, D. I. cuTT: A High-Performance Tensor Transpose Library for CUDA Compatible GPUs. 2017.

220. Beckingsale, D.; Mcfadden, M.; Dahm, J.; Pankajakshan, R.; Hornung, R. Umpire: Application-Focused Management and Coordination of Complex Hierarchical Memory. *IBM J. Res. & Dev.* **2020**, *64*, 1–10.

221. Van De Geijn, R. A.; Watts, J. SUMMA: Scalable Universal Matrix Multiplication algorithm. *Concurrency: Pract. Exper.* **1997**, *9*, 255–274.

222. Peng, C.; Calvin, J. A.; Pavošević, F.; Zhang, J.; Valeev, E. F. Massively Parallel Implementation of Explicitly Correlated Coupled-Cluster Singles and Doubles Using TiledArray Framework. *J. Phys. Chem. A* **2016**, *120*, 10231–10244.

223. Peng, C.; Clement, M. C.; Valeev, E. F. State-Averaged Pair Natural Orbitals for Excited States: A Route toward Efficient Equation of Motion Coupled-Cluster. *J. Chem. Theory Comput.* **2018**, *14*, 5597–5607.

224. Peng, C.; Calvin, J. A.; Valeev, E. F. Coupled-Cluster Singles, Doubles and Perturbative Triples with Density Fitting Approximation for Massively Parallel Heterogeneous Platforms. *Int. J. Quant. Chem.* **2019**, *119*, e25894.

225. Clement, M. C.; Zhang, J.; Lewis, C. A.; Yang, C.; Valeev, E. F. Optimized Pair Natural Orbitals for the Coupled Cluster Methods. *J. Chem. Theory Comput.* **2018**, *14*, 4581–4589.

226. Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; Flannery, B. P. *Numerical Recipes: The Art of Scientific Computing*, 3rd ed.; Cambridge University Press: New York, 2007.

227. Smith, B. T.; Boyle, J. M.; Dongarra, J. J.; Garbow, B. S.; Ikebe, Y.; Klema, V. C.; Moler, C. B. *Matrix Eigensystem Routines: EISPACK Guide*; pub-SV, 1976; pp vii + 551.

228. Fann, G.; Littlefield, R.; Elwood, D. Performance of a Fully Parallel Dense Real Symmetric Eigensolver in Quantum Chemistry Applications. Proc. High Performance Computing '95, Simulation MultiConference. San Diego, CA, 1995.

229. Anderson, E.; Bai, Z.; Bischof, C.; Blackford, L. S.; Demmel, J.; Dongarra, J. J.; Du Croz, J.; Hammarling, S.; Greenbaum, A.; McKenney, A. et al. *LAPACK Users' Guide (Third Ed.)*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 1999.

230. Auckenthaler, T.; Blum, V.; Bungartz, H.-J.; Huckle, T.; Johanni, R.; Krämer, L.; Lang, B.; Lederer, H.; Willems, P. R. Parallel Solution of Partial Symmetric Eigenvalue Problems from Electronic Structure Calculations. *Parallel Comput.* **2011**, *37*, 783–794.

231. Marek, A.; Blum, V.; Johanni, R.; Havu, V.; Lang, B.; Auckenthaler, T.; Heinecke, A.; Bungartz, H.-J.; Lederer, H. The ELPA Library - Scalable Parallel Eigenvalue Solutions for Electronic Structure Theory and Computational Science. *J. Phys.: Condens. Matter* **2014**, *26*, 213201.

232. Davidson, E. R. The Iterative Calculation of a Few of the Lowest Eigenvalues and

Corresponding Eigenvectors Large Real-Symmetric Matrices. *J. Comput. Phys.* **1975**, *17*, 87–94.

233. Knyazev, A. V. Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method. *SIAM J. Sci. Comput.* **2001**, *23*, 517–541.

234. Zhou, Y.; Saad, Y.; Tiago, M. L.; Chelikowsky, J. R. Self-Consistent-Field Calculations Using Chebyshev-Filtered Subspace Iteration. *J. Comput. Phys.* **2006**, *219*, 172–184.

235. Kronik, L.; Makmal, A.; Tiago, M. L.; Alemany, M.; Jain, M.; Huang, X.; Saad, Y.; Chelikowsky, J. R. PARSEC–The Pseudopotential Algorithm for Real-Space Electronic Structure Calculations: Recent Advances and Novel Applications to Nano-Structures. *Phys. Status Solidi B* **2006**, *243*, 1063–1079.

236. Pask, J.; Sterne, P. Finite Element Methods in ab initio Electronic Structure Calculations. *Model. Simul. Mat. Sci. Eng.* **2005**, *13*, R71.

237. Zhang, H.; Smith, B.; Sternberg, M.; Zapol, P. SIPs: Shift-and-Invert Parallel Spectral Transformations. *ACM Trans. Math. Software* **2007**, *33*, 9–40.

238. Keçeli, M.; Zhang, H.; Zapol, P.; Dixon, D. A.; Wagner, A. F. Shift-and-Invert Parallel Spectral Transformation Eigensolver: Massively Parallel Performance for Density-Functional Based Tight-Binding. *J. Comput. Chem.* **2016**, *37*, 448–459.

239. Williams-Young, D. B.; Beckman, P. G.; Yang, C. A Shift Selection Strategy for Parallel Shift-Invert Spectrum Slicing in Symmetric Self-Consistent Eigenvalue Computation. *ACM Trans. Math. Softw.* **2020**, *46*.

240. Williams-Young, D. B.; Yang, C. Parallel Shift-Invert Spectrum Slicing on Distributed Architectures with GPU Accelerators. 49th International Conference on Parallel Processing - ICPP. New York, NY, USA, 2020.

241. Lin, L.; Saad, Y.; Yang, C. Approximating Spectral Densities of Large Matrices. *SIAM Rev.* **2016**, *58*, 34–65.

242. Pulay, P. Convergence Acceleration of Iterative Sequences. The Case of SCF Iteration. *Chem. Phys. Lett.* **1980**, *73*, 393–398.

243. Rohwedder, T.; Schneider, R. J. An Analysis for the DIIS Acceleration Method Used in Quantum Chemistry Calculations. *J. Math. Chem.* **2011**, *49*, 1889–1914.

244. Yu, V. W.-z.; Corsetti, F.; García, A.; Huhn, W. P.; Jacquelin, M.; Jia, W.; Lange, B.; Lin, L.; Lu, J.; Mi, W. et al. ELSI: A Unified Software Interface for Kohn–Sham Electronic Structure Solvers. *Comput. Phys. Commun.* **2018**, *222*, 267–285.

245. Yu, V. W.-z.; Campos, C.; Dawson, W.; García, A.; Havu, V.; Hourahine, B.; Huhn, W. P.; Jacquelin, M.; Jia, W.; Keçeli, M. et al. ELSI — An Open Infrastructure for Electronic Structure Solvers. *Comput. Phys. Commun.* **2020**, *256*, 107459.

246. Knoll, D.; Keyes, D. Jacobian-Free Newton–Krylov Methods: A Survey of Approaches and Applications. *J. Comput. Phys.* **2004**, *193*, 357–397.

247. Saad, Y.; Schultz, M. H. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM J. Sci. Stat. Comput.* **1986**, *7*, 856–869.

248. Yang, C.; Brabec, J.; Veis, L.; Williams-Young, D. B.; Kowalski, K. Solving Coupled Cluster Equations by the Newton Krylov Method. *Front. Chem.* **2020**, *8*, 987.

249. Ostlund, N. S.; Szabo, A. *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*; Dover Publications Inc New edition edn, 1996.

250. Kohn, W.; Sham, L. J. Self-Consistent Equations Including Exchange and Correlation Effects. *Phys. Rev.* **1965**, *140*, A1133.

251. Parr, R. G.; Yang, W. *Density-Functional Theory of Atoms and Molecules (International Series of Monographs on Chemistry)*; Oxford University Press, USA, 1994.

252. Dreizler, R. M.; Gross, E. K. U. *Density Functional Theory: An Approach to the Quantum Many-Body Problem*; Springer: Berlin, 1990.

253. Perdew, J. P.; Schmidt, K. In *Density Functional Theory and Its Application to Materials*; Van Doren, V., Ed.; Melville, NY: American Institute of Physics, 2011; Chapter Jacob's Ladder of Density Functional Approximations for the Exchange-Correlation Energy Density Functional Theory and Its Application to Materials, pp 1–20.

254. Mardirossian, N.; Head-Gordon, M. Thirty years of Density Functional Theory in Computational Chemistry: An Overview and Extensive Assessment of 200 Density Functionals. *Mol. Phys.* **2017**, *115*, 2315–2372.

255. Ufimtsev, I. S.; Martinez, T. J. Quantum Chemistry on Graphical Processing Units. 2. Direct Self-Consistent-Field Implementation. *J. Chem. Theory Comput.* **2009**, *5*, 1004–1015.

256. Rudberg, E.; Rubensson, E. H.; Sałek, P. Kohn- Sham Density Functional Theory Electronic Structure Calculations with Linearly Scaling Computational Time and Memory Usage. *J. Chem. Theory Comput.* **2011**, *7*, 340–350.

257. Bock, N.; Challacombe, M.; Gan, C. K.; Henkelman, G.; Nemeth, K.; Niklasson, A. M.; Odell, A.; Schwegler, E.; Tymczak, C.; Weber, V. FreeON: A Suite of Programs for Linear Scaling Quantum Chemistry, 2011. `https://github.com/FreeON/freeon`, [Online; accessed 30-August-2020].

258. Skylaris, C.-K.; Haynes, P. D.; Mostofi, A. A.; Payne, M. C. Introducing ONETEP: Linear-Scaling Density Functional Simulations on Parallel Computers. *J. Chem. Phys.* **2005**, *122*, 084119.

259. Bowler, D. R.; Miyazaki, T. Calculations for Millions of Atoms with Density Functional Theory: Linear Scaling Shows Its Potential. *J. Phys.: Condens. Matter* **2010**, *22*, 074207.

260. VandeVondele, J.; Krack, M.; Mohamed, F.; Parrinello, M.; Chassaing, T.; Hutter, J. Quickstep: Fast and Accurate Density Functional Calculations Using a Mixed Gaussian and Plane Waves Approach. *Comput. Phys. Commun.* **2005**, *167*, 103–128.

261. Mohr, S.; Ratcliff, L. E.; Genovese, L.; Caliste, D.; Boulanger, P.; Goedecker, S.; Deutsch, T. Accurate and Efficient Linear Scaling DFT Calculations with Universal Applicability. *Phys. Chem. Chem. Phys.* **2015**, *17*, 31360–31370.

262. Ozaki, T. O (N) Krylov-Subspace Method for Large-Scale ab initio Electronic Structure Calculations. *Phys. Rev. B* **2006**, *74*, 245101.

263. Blum, V.; Gehrke, R.; Hanke, F.; Havu, P.; Havu, V.; Ren, X.; Reuter, K.; Scheffler, M. Ab Initio Molecular Simulations with Numeric Atom-Centered Orbitals. *Comput. Phys. Commun.* **2009**, *180*, 2175–2196.

264. Ren, X.; Rinke, P.; Blum, V.; Wieferink, J.; Tkatchenko, A.; Sanfilippo, A.; Reuter, K.; Scheffler, M. Resolution-of-Identity Approach to Hartree–Fock, Hybrid Density Functionals, RPA, MP2 and GW with Numeric Atom-Centered Orbital Basis Functions. *New J. Phys.* **2012**, *14*, 053020.

265. Ratcliff, L. E.; Mohr, S.; Huhs, G.; Deutsch, T.; Masella, M.; Genovese, L. Challenges in Large Scale Quantum Mechanical Calculations. *WIREs Comput. Mol. Sci.* **2017**, *7*, e1290.

266. Chow, E.; Liu, X.; Smelyanskiy, M.; Hammond, J. R. Parallel Scalability of Hartree–Fock Calculations. *J. Chem. Phys.* **2015**, *142*, 104103.

267. Köppl, C.; Werner, H.-J. Parallel and Low-Order Scaling Implementation of Hartree–Fock Exchange Using Local Density Fitting. *J. Chem. Theory Comput.* **2016**, *12*, 3122–3134.

268. Kussmann, J.; Ochsenfeld, C. Employing OpenCL to Accelerate Ab Initio Calculations on Graphics Processing Units. *J. Chem. Theory Comput.* **2017**, *13*, 2712–2716.

269. Manathunga, M.; Miao, Y.; Mu, D.; Götz, A. W.; Merz, K. M. Parallel Implementation of Density Functional Theory Methods in the Quantum Interaction Computational Kernel Program. *J. Chem. Theory Comput.* **2020**, *16*, 4315–4326.

270. Luehr, N.; Sisto, A.; Martínez, T. J. *Electronic Structure Calculations on Graphics Processing Units*; John Wiley and Sons, Ltd, 2016; Chapter Gaussian Basis Set Hartree–Fock, Density Functional Theory, and Beyond on GPUs, pp 67–100.

271. Yasuda, K. Accelerating Density Functional Calculations with Graphics Processing Unit. *J. Chem. Theory Comput.* **2008**, *4*, 1230–1236.

272. Titov, A. V.; Ufimtsev, I. S.; Luehr, N.; Martinez, T. J. Generating Efficient Quantum Chemistry Codes for Novel Architectures. *J. Chem. Theory Comput.* **2013**, *9*, 213–221.

273. Brown, P.; Woods, C. J.; McIntosh-Smith, S.; Manby, F. R. A Massively Multicore Parallelization of the Kohn-Sham Energy Gradients. *J. Comput. Chem.* **2010**, *31*, 2008–2013.

274. Kalinowski, J.; Wennmohs, F.; Neese, F. Arbitrary Angular Momentum Electron Repulsion Integrals with Graphical Processing Units: Application to the Resolution of Identity Hartree–Fock Method. *J. Chem. Theory Comput.* **2017**, *13*, 3160–3170.

275. Ufimtsev, I. S.; Martínez, T. J. Quantum Chemistry on Graphical Processing Units. 1. Strategies for Two-Electron Integral Evaluation. *J. Chem. Theory Comput.* **2008**, *4*, 222–231.

276. Ufimtsev, I. S.; Martinez, T. J. Quantum Chemistry on Graphical Processing Units. 3. Analytical Energy Gradients, Geometry Optimization, and First Principles Molecular Dynamics. *J. Chem. Theory Comput.* **2009**, *5*, 2619–2628.

277. Miao, Y.; Merz, K. M. Acceleration of Electron Repulsion Integral Evaluation on Graphics Processing Units via Use of Recurrence Relations. *J. Chem. Theory Comput.* **2013**, *9*, 965–976.

278. Laqua, H.; Thompson, T. H.; Kussmann, J.; Ochsenfeld, C. Highly Efficient, Linear-Scaling Seminumerical Exact-Exchange Method for Graphic Processing Units. *J. Chem. Theory Comput.* **2020**, *16*, 1456–1468.

279. Barca, G. M. J.; Galvez-Vallejo, J. L.; Poole, D. L.; Rendell, A. P.; Gordon, M. S. High-Performance, Graphics Processing Unit-Accelerated Fock Build Algorithm. *J. Chem. Theory Comput.* **2020**, *16*, 7232–7238.

280. Barca, G. M. J.; Poole, D. L.; Vallejo, J. L. G.; Alkan, M.; Bertoni, C.; Rendell, A. P.; Gordon, M. S. Scaling the Hartree-Fock Matrix Build on Summit. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2020.

281. Williams-Young, D. B.; de Jong, W. A.; van Dam, H. J. J.; Yang, C. On the Efficient Evaluation of the Exchange Correlation Potential on Graphics Processing Unit Clusters. *Front. Chem.* **2020**, *8*, 951.

282. Hohenberg, P.; Kohn, W. Inhomogeneous Electron Gas. *Physical review* **1964**, *136*, B864.

283. Pickett, W. E. Electronic Structure of the High-Temperature Oxide Superconductors. *Rev. Mod. Phys.* **1989**, *61*, 433–512.

284. Ihm, J.; Zunger, A.; Cohen, M. L. Momentum-Space Formalism for the Total Energy of Solids. *J. Phys. C: Solid State Phys.* **1979**, *12*, 4409–4422.

285. Car, R.; Parrinello, M. Unified Approach for Molecular Dynamics and Density-Functional Theory. *Phys. Rev. Lett.* **1985**, *55*, 2471–2474.

286. Payne, M. C.; Teter, M. P.; Allan, D. C.; Arias, T.; Joannopoulos, J. Iterative Minimization Techniques for Ab Initio Total-Energy Calculations: Molecular Dynamics and Conjugate Gradients. *Rev. Mod. Phys.* **1992**, *64*, 1045–1097.

287. Remler, D. K.; Madden, P. A. Molecular Dynamics Without Effective Potentials via the Car-Parrinello Approach. *Mol. Phys.* **1990**, *70*, 921–966.

288. Dal Corso, A. *Quantum-Mechanical Ab-initio Calculation of the Properties of Crystalline Materials*; Springer, 1996; pp 155–178.

289. Pask, J.; Klein, B.; Fong, C.; Sterne, P. Real-Space Local Polynomial Basis for Solid-State Electronic-Structure Calculations: A Finite-Element Approach. *Phys. Rev. B* **1999**, *59*, 12352–12358.

290. Andreoni, W.; Curioni, A. New Advances in Chemistry and Materials Science with CPMD and Parallel Computing. *Parallel Comput.* **2000**, *26*, 819–842.

291. Fattebert, J.-L.; Bernholc, J. Towards Grid-Based O(N) Density-Functional Theory Methods: Optimized Nonorthogonal Orbitals and Multigrid Acceleration. *Phys. Rev. B* **2000**, *62*, 1713–1722.

292. Marx, D.; Hutter, J. *Ab Initio Molecular Dynamics: Basic Theory and Advanced Methods*; Cambridge University Press, 2009.

293. Martin, R. M. *Electronic Structure: Basic Theory and Practical Methods*; Cambridge University Press, 2004.

294. Valiev, M.; Bylaska, E. J.; Gramada, A.; Weare, J. H. First Principles Molecular Dynamics Simulations Using Density-Functional Theory. Reviews in Modern Quantum Chemistry: A Celebration of the Contributions of Robert G. Parr. 2002.

295. Bylaska, E. J.; Tsemekhman, K.; Govind, N.; Valiev, M. Large-Scale Plane-Wave-Based Density Functional Theory: Formalism, Parallelization, and Applications. Computational Methods for Large Systems: Electronic Structure Approaches for Biotechnology and Nanotechnology. 2011; pp 77–116.

296. Moore, S.; Briggs, E.; Hodak, M.; Lu, W.; Bernholc, J.; Lee, C.-W. Scaling the RMG Quantum Mechanics Code. Proceedings of the Extreme Scaling Workshop. 2012; pp 1–6.

297. Osei-Kuffuor, D.; Fattebert, J.-L. Accurate and scalable O(N) Algorithm for First-Principles Molecular-Dynamics Computations on Large Parallel Computers. *Phys. Rev. Lett.* **2014**, *112*, 046401.

298. Chen, Y.; Bylaska, E. J.; Weare, J. H. First Principles Estimation of Geochemically Important Transition Metal Oxide Properties. Molecular Modeling of Geochemical Reactions: An Introduction. 2016; pp 107–149.

299. Sundararaman, R.; Letchworth-Weaver, K.; Schwarz, K. A.; Gunceler, D.; Ozhabes, Y.; Arias, T. JDFTx: Software for Joint Density-Functional Theory. *SoftwareX* **2017**, *6*, 278–284.

300. Blöchl, P. E. Projector Augmented-Wave Method. *Phys. Rev. B* **1994**, *50*, 17953–17979.

301. Holzwarth, N. A. W.; Matthews, G. E.; Dunning, R. B.; Tackett, A. R.; Zeng, Y. Comparison of the Projector Augmented-Wave, Pseudopotential, and Linearized Augmented-Plane-Wave Formalisms for Density-Functional Calculations of Solids. *Phys. Rev. B* **1997**, *55*, 2005–2017.

302. Kresse, G.; Joubert, D. From Ultrasoft Pseudopotentials to the Projector Augmented-Wave Method. *Phys. Rev. B* **1999**, *59*, 1758–1775.

303. Valiev, M.; Weare, J. H. The Projector-Augmented Plane Wave Method Applied to Molecular Bonding. *J. Phys. Chem. A* **1999**, *103*, 10588–10601.

304. Bylaska, E. J.; Valiev, M.; Kawai, R.; Weare, J. H. Parallel Implementation of the Projector Augmented Plane Wave Method for Charged Systems. *Comput. Phys. Commun.* **2002**, *143*, 11–28.

305. Phillips, J. C. Energy-Band Interpolation Scheme Based on a Pseudopotential. *Phys. Rev.* **1958**, *112*, 685–695.

306. Phillips, J. C.; Kleinman, L. New Method for Calculating Wave Functions in Crystals and Molecules. *Phys. Rev.* **1959**, *116*, 287–294.

307. Austin, B. J.; Heine, V.; Sham, L. J. General Theory of Pseudopotentials. *Phys. Rev.* **1962**, *127*, 276–282.

308. Yin, M.; Cohen, M. L. Theory of ab initio Pseudopotential Calculations. *Phys. Rev. B* **1982**, *25*, 7403–7412.

309. Bachelet, G. B.; Hamann, D. R.; Schlüter, M. Pseudopotentials That Work: From H to Pu. *Phys. Rev. B* **1982**, *26*, 4199–4228.

310. Hamann, D. R. Generalized Norm-Conserving Pseudopotentials. *Phys. Rev. B* **1989**, *40*, 2980–2987.

311. Troullier, N.; Martins, J. L. Efficient Pseudopotentials for Plane-Wave Calculations. *Phys. Rev. B* **1991**, *43*, 1993–2006.

312. Bylaska, E. J. *Annual Reports in Computational Chemistry*; Elsevier, 2017; Vol. 13; Chapter Plane-Wave DFT Methods for Chemistry, pp 185–228.

313. Štich, I.; Car, R.; Parrinello, M.; Baroni, S. Conjugate Gradient Minimization of the Energy Functional: A New Method for Electronic Structure Calculation. *Phys. Rev. B* **1989**, *39*, 4997–5004.

314. Kresse, G.; Hafner, J. Ab initio Molecular Dynamics for Liquid Metals. *Phys. Rev. B* **1993**, *47*, 558–561.

315. Clarke, L. J.; Štich, I.; Payne, M. C. Large-Scale Ab Initio Total Energy Calculations on Parallel Computers. *Comput. Phys. Commun.* **1992**, *72*, 14–28.

316. Nelson, J.; Plimpton, S.; Sears, M. Plane-Wave Electronic-Structure Calculations on a Parallel Supercomputer. *Phys. Rev. B* **1993**, *47*, 1765–1774.

317. Wiggs, J.; Jonsson, H. A Hybrid Decomposition Parallel Implementation of the Car-Parrinello Method. *Comput. Phys. Commun.* **1995**, *87*, 319–340.

318. Canning, A.; Wang, L.; Williamson, A.; Zunger, A. Parallel Empirical Pseudopotential Electronic Structure Calculations for Million Atom Systems. *J. Comput. Phys.* **2000**, *160*, 29–41.

319. Canning, A.; Raczkowski, D. Scaling First-Principles Plane-Wave Codes to Thousands of Processors. *Comput. Phys. Commun.* **2005**, *169*, 449–453.

320. Bylaska, E. J.; Glass, K.; Baxter, D.; Baden, S. B.; Weare, J. H. Hard Scaling Challenges for Ab Initio Molecular Dynamics Capabilities in NWChem: Using 100,000 CPUs per second. Journal of Physics: Conference Series. 2009; p 012028.

321. Bylaska, E. J.; Tsemekhman, K.; Baden, S. B.; Weare, J. H.; Jonsson, H. Parallel Implementation of Γ-Point Pseudopotential Plane-Wave DFT with Exact Exchange. *J. Comput. Chem.* **2011**, *32*, 54–69.

322. Bylaska, E. J.; Aprà, E.; Kowalski, K.; Jacquelin, M.; De Jong, W. A.; Vishnu, A.; Palmer, B.; Daily, J.; Straatsma, T. P.; Hammond, J. R. Transitioning NWChem to the Next Generation of Manycore Machines. *Exascale Scientific Applications: Scalability and Performance Portability* **2017**, 165–186.

323. Canning, A.; Shalf, J.; Wright, N.; Anderson, S.; Gajbe, M. A Hybrid MPI/OpenMP 3D FFT for Plane Wave First-Principles Materials Science Codes. Proceedings of the International Conference on Scientific Computing (CSC). 2012; pp 1–6.

324. Swaddle, T. W.; Rosenqvist, J.; Yu, P.; Bylaska, E. J.; Phillips, B. L.; Casey, W. H. Kinetic Evidence for Five-Coordination in $AlOH^{2+}_{(aq)}$ Ion. *Science* **2005**, *308*, 1450–1453.

325. Rustad, J. R.; Bylaska, E. J. Ab Initio Calculation of Isotopic Fractionation in $B(OH)_3$(aq) and BOH_4^-(aq). *J. Am. Chem. Soc.* **2007**, *129*, 2222–2223.

326. Atta-Fynn, R.; Johnson, D. F.; Bylaska, E. J.; Ilton, E. S.; Schenter, G. K.; De Jong, W. A. Structure and Hydrolysis of the U(IV), U(V), and U(VI) Aqua Ions from Ab Initio Molecular Dynamics Simulations. *Inorg. Chem.* **2012**, *51*, 3016–3024.

327. Fulton, J. L.; Bylaska, E. J.; Bogatko, S.; Balasubramanian, M.; Cauët, E.; Schenter, G. K.; Weare, J. H. Near-Quantitative Agreement of Model-Free DFT-MD Predictions with XAFS Observations of the Hydration Structure of Highly Charged Transition-Metal Ions. *J. Phys. Chem. Lett.* **2012**, *3*, 2588–2593.

328. Odoh, S. O.; Bylaska, E. J.; de Jong, W. A. Coordination and Hydrolysis of Plutonium Ions in Aqueous Solution Using Car–Parrinello Molecular Dynamics Free Energy Simulations. *J. Phys. Chem. A* **2013**, *117*, 12256–12267.

329. Atta-Fynn, R.; Bylaska, E. J.; de Jong, W. A. Importance of Counteranions on the Hydration Structure of the Curium Ion. *J. Phys. Chem. Lett.* **2013**, *4*, 2166–2170.

330. Fattebert, J.-L.; Osei-Kuffuor, D.; Draeger, E. W.; Ogitsu, T.; Krauss, W. D. Modeling Dilute Solutions Using First-Principles Molecular Dynamics: Computing More Than A Million Atoms With Over a Million Cores. High Performance Computing, Networking, Storage and Analysis, SC16: International Conference. 2016; pp 12–22.

331. Canning, A. Scalable Parallel 3D FFTs for Electronic Structure Codes. International Conference on High Performance Computing for Computational Science. 2008; pp 280–286.

332. Canning, A.; Shalf, J.; Wang, L.-W.; Wasserman, H.; Gajbe, M. A Comparison of Different Communication Structures for Scalable Parallel Three Dimensional FFTs in First Principle Codes. Parallel Computing: From Multicores and GPU's to Petascale. 2010; pp 107–16.

333. Ayala, O.; Wang, L.-P. Parallel Implementation and Scalability Analysis of 3D Fast Fourier Transform Using 2D Domain Decomposition. *Parallel Comput.* **2013**, *39*, 58 – 77.

334. Franchetti, F.; Spampinato, D. G.; Kulkarni, A.; Popovici, D. T.; Low, T. M.; Franusich, M.; Canning, A.; McCorquodale, P.; Van Straalen, B.; Colella, P. FFTX and SpectralPack: A First Look. 2018 IEEE 25th International Conference on High Performance Computing Workshops (HiPCW). 2018; pp 18–27.

335. Coester, F. Bound States of a Many-Particle System. *Nucl. Phys.* **1958**, *7*, 421–424.

336. Coester, F.; Kummel, H. Short-Range Correlations in Nuclear Wave Functions. *Nucl. Phys.* **1960**, *17*, 477–485.

337. Čížek, J. On the Correlation Problem in Atomic and Molecular Systems. Calculation of Wavefunction Components in Ursell-Type Expansion Using Quantum-Field Theoretical Methods. *J. Chem. Phys.* **1966**, *45*, 4256–4266.

338. Purvis, G.; Bartlett, R. A Full Coupled-Cluster Singles and Doubles Model: The Inclusion of Disconnected Triples. *J. Chem. Phys.* **1982**, *76*, 1910–1918.

339. Paldus, J.; Li, X. A Critical Assessment of Coupled Cluster Method in Quantum Chemistry. *Adv. Chem. Phys.* **1999**, *110*, 1–175.

340. Crawford, T. D.; Schaefer, H. F. An Introduction to Coupled Cluster Theory for Computational Chemists. *Rev. Comput. Chem.* **2000**, *14*, 33–136.

341. Bartlett, R. J.; Musiał, M. Coupled-Cluster Theory in Quantum Chemistry. *Rev. Mod. Phys.* **2007**, *79*, 291–352.

342. Raghavachari, K.; Trucks, G. W.; Pople, J. A.; Head-Gordon, M. A Fifth-Order Perturbation Comparison of Electron Correlation Theories. *Chem. Phys. Lett.* **1989**, *157*, 479–483.

343. Noga, J.; Bartlett, R. J. The Full CCSDT Model for Molecular Electronic Structure. *J. Chem. Phys.* **1987**, *86*, 7041–7050.

344. Noga, J.; Bartlett, R. J. Erratum: The Full CCSDT Model for Molecular Electronic Structure [JCP 86, 7041 (1987)]. *J. Chem. Phys.* **1988**, *89*, 3401–3401.

345. Scuseria, G. E.; Schaefer, H. F. A New Implementation of the Full CCSDT Model for Molecular Electronic Structure. *Chem. Phys. Lett.* **1988**, *152*, 382–386.

346. Oliphant, N.; Adamowicz, L. Coupled-Cluster Method Truncated at Quadruples. *J. Chem. Phys.* **1991**, *95*, 6645–6651.

347. Kucharski, S. A.; Bartlett, R. J. Recursive Intermediate Factorization and Complete Computational Linearization of the Coupled-Cluster Single, Double, Triple, and Quadruple Excitation Equations. *Theor. Chem. Acc.* **1991**, *80*, 387–405.

348. Beebe, N. H.; Linderberg, J. Simplifications in the Generation and Transformation of Two-Electron Integrals in Molecular Calculations. *Int. J. Quant. Chem.* **1977**, *12*, 683–705.

349. Røeggen, I.; Wisløff-Nilssen, E. On the Beebe-Linderberg Two-Electron Integral Approximation. *Chem. Phys. Lett.* **1986**, *132*, 154–160.

350. Aquilante, F.; Pedersen, T. B.; Lindh, R. Low-Cost Evaluation of the Exchange Fock Matrix from Cholesky and Density Fitting Representations of the Electron Repulsion Integrals. *J. Chem. Phys.* **2007**, *126*, 194106.

351. Koch, H.; Sánchez de Merás, A.; Pedersen, T. B. Reduced Scaling in Electronic Structure Calculations Using Cholesky Decompositions. *J. Chem. Phys.* **2003**, *118*, 9481–9484.

352. Dunlap, B. I.; Connolly, J. W. D.; Sabin, J. R. On Some Approximations in Applications of Xα Theory. *J. Chem. Phys.* **1979**, *71*, 3396–3402.

353. Vahtras, O.; Almlöf, J.; Feyereisen, M. Integral Approximations for LCAO-SCF Calculations. *Chem. Phys. Lett.* **1993**, *213*, 514–518.

354. Feyereisen, M.; Fitzgerald, G.; Komornicki, A. Use of Approximate Integrals in ab initio Theory. An Application in MP2 Energy Calculations. *Chem. Phys. Lett.* **1993**, *208*, 359–363.

355. Weigend, F. A Fully Direct RI-HF Algorithm: Implementation, Optimised Auxiliary Basis Sets, Demonstration of Accuracy and Efficiency. *Phys. Chem. Chem. Phys.* **2002**, *4*, 4285–4291.

356. Werner, H.-J.; Manby, F. R.; Knowles, P. J. Fast Linear Scaling Second-Order Møller-Plesset Perturbation Theory (MP2) Using Local and Density Fitting Approximations. *J. Chem. Phys.* **2003**, *118*, 8149–8160.

357. Sodt, A.; Subotnik, J. E.; Head-Gordon, M. Linear Scaling Density Fitting. *J. Chem. Phys.* **2006**, *125*, 194109.

358. Boström, J.; Pitonak, M.; Aquilante, F.; Neogrady, P.; Pedersen, T. B.; Lindh, R. Coupled Cluster and Møller–Plesset Perturbation Theory Calculations of Noncovalent Intermolecular Interactions Using Density Fitting With Auxiliary Basis Sets from Cholesky Decompositions. *J. Chem. Theory Comput.* **2012**, *8*, 1921–1928.

359. Schütz, M.; Manby, F. R. Linear Scaling Local Coupled Cluster Theory with Density Fitting. Part I: 4-External Integrals. *Phys. Chem. Chem. Phys.* **2003**, *5*, 3349–3358.

360. Werner, H.-J.; Knizia, G.; Manby, F. R. Explicitly Correlated Coupled Cluster Methods with Pair-Specific Geminals. *Mol. Phys.* **2011**, *109*, 407–417.

361. DePrince III, A. E.; Kennedy, M. R.; Sumpter, B. G.; Sherrill, C. D. Density-Fitted Singles and Doubles Coupled Cluster on Graphical Processing Units. *Mol. Phys.* **2014**, *112*, 844–852.

362. Parrish, R. M.; Sherrill, C. D.; Hohenstein, E. G.; Kokkila, S. I. L.; Martínez, T. J. Communication: Acceleration of Coupled Cluster Singles and Doubles via Orbital-Weighted Least-Squares Tensor Hypercontraction. *J. Chem. Phys.* **2014**, *140*, 181102.

363. Bozkaya, U.; Sherrill, C. D. Analytic Energy Gradients for the Coupled-Cluster Singles and Doubles Method with the Density-Fitting Approximation. *J. Chem. Phys.* **2016**, *144*, 174103.

364. Pedersen, T. B.; Sánchez de Merás, A. M.; Koch, H. Polarizability and Optical Rotation Calculated from the Approximate Coupled Cluster Singles and Doubles CC2 Linear Response Theory Using Cholesky Decompositions. *J. Chem. Phys.* **2004**, *120*, 8887–8897.

365. Epifanovsky, E.; Zuev, D.; Feng, X.; Khistyaev, K.; Shao, Y.; Krylov, A. I. General Implementation of the Resolution-of-the-Identity and Cholesky Representations of Electron Repulsion Integrals within Coupled-Cluster and Equation-of-Motion Methods: Theory and Benchmarks. *J. Chem. Phys.* **2013**, *139*, 134105.

366. Feng, X.; Epifanovsky, E.; Gauss, J.; Krylov, A. I. Implementation of Analytic Gradients for CCSD and EOM-CCSD Using Cholesky Decomposition of the Electron-Repulsion Integrals and Their Derivatives: Theory and Benchmarks. *J. Chem. Phys.* **2019**, *151*, 014110.

367. Folkestad, S. D.; Kjønstad, E. F.; Koch, H. An Efficient Algorithm for Cholesky Decomposition of Electron Repulsion Integrals. *J. Chem. Phys.* **2019**, *150*, 194112.

368. Peng, B.; Kowalski, K. Highly Efficient and Scalable Compound Decomposition of Two-Electron Integral Tensor and its Application in Coupled Cluster Calculations. *J. Chem. Theory Comput.* **2017**, *13*, 4179–4192.

369. Fedorov, D. G.; Kitaura, K. Extending the Power of Quantum Chemistry to Large Systems with the Fragment Molecular Orbital Method. *J. Phys. Chem. A* **2007**, *111*, 6904–6914.

370. Stoll, H. Correlation Energy of Diamond. *Phys. Rev. B* **1992**, *46*, 6700–6704.

371. Li, W.; Piecuch, P.; Gour, J. R.; Li, S. Local Correlation Calculations Using Standard and Renormalized Coupled-Cluster Approaches. *J. Chem. Phys.* **2009**, *131*, 114109.

372. Gordon, M. S.; Fedorov, D. G.; Pruitt, S. R.; Slipchenko, L. V. Fragmentation Methods: A Route to Accurate Calculations on Large Systems. *Chem. Rev.* **2012**, *112*, 632–672.

373. Herbert, J. M. Fantasy Versus Reality in Fragment-Based Quantum Chemistry. *J. Chem. Phys.* **2019**, *151*, 170901.

374. Pulay, P. Localizability of Dynamic Electron Correlation. *Chem. Phys. Lett.* **1983**, *100*, 151–154.

375. Sæbø, S.; Pulay, P. Local Configuration Interaction: An Efficient Approach for Larger Molecules. *Chem. Phys. Lett.* **1985**, *113*, 13–18.

376. Saebo, S.; Pulay, P. The Local Correlation Treatment. II. Implementation and Tests. *J. Chem. Phys.* **1988**, *88*, 1884–1890.

377. Schütz, M.; Hetzer, G.; Werner, H.-J. Low-Order Scaling Local Electron Correlation Methods. I. Linear Scaling Local MP2. *J. Chem. Phys.* **1999**, *111*, 5691–5705.

378. Schütz, M.; Werner, H.-J. Local Perturbative Triples Correction (T) with Linear Cost Scaling. *Chem. Phys. Lett.* **2000**, *318*, 370–378.

379. Edmiston, C.; Krauss, M. Configuration-Interaction Calculation of H3 and H2. *J. Chem. Phys.* **1965**, *42*, 1119–1120.

380. Edmiston, C.; Krauss, M. Pseudonatural Orbitals as a Basis for the Superposition of Configurations. I. He2+. *J. Chem. Phys.* **1966**, *45*, 1833–1839.

381. Meyer, W. Ionization Energies of Water from PNO-CI Calculations. *Int. J. Quant. Chem.* **1971**, *5*, 341–348.

382. Meyer, W. PNO–CI Studies of Electron Correlation Effects. I. Configuration Expansion by Means of Nonorthogonal Orbitals, and Application to the Ground State and Ionized States of Methane. *J. Chem. Phys.* **1973**, *58*, 1017–1035.

383. Meyer, W. PNO-CI and CEPA Studies of Electron Correlation Effects. *Theor. Chim. Acta* **1974**, *35*, 277–292.

384. Ahlrichs, R.; Lischka, H.; Staemmler, V.; Kutzelnigg, W. PNO-CL (Pair-Natural-Orbital Configuration Interaction) AND CEPA-PNO (Coupled Electron Pair Approximation with Pair Natural Orbitals) Calculations of Molecular Systems. 1. Outline of Method for Closed-Shell States. *J. Chem. Phys.* **1975**, *62*, 1225–1234.

385. Neese, F.; Wennmohs, F.; Hansen, A. Efficient and Accurate Local Approximations to Coupled-Electron Pair Approaches: An Attempt to Revive the Pair Natural Orbital Method. *J. Chem. Phys.* **2009**, *130*, 114108.

386. Neese, F.; Hansen, A.; Liakos, D. G. Efficient and Accurate Approximations to the Local Coupled Cluster Singles Doubles Method Using a Truncated Pair Natural Orbital Basis. *J. Chem. Phys.* **2009**, *131*, 064103.

387. Yang, J.; Chan, G. K.-L.; Manby, F. R.; Schütz, M.; Werner, H.-J. The Orbital-Specific-Virtual Local Coupled Cluster Singles and Doubles Method. *J. Chem. Phys.* **2012**, *136*, 144105.

388. Eriksen, J. J.; Baudin, P.; Ettenhuber, P.; Kristensen, K.; Kjærgaard, T.; Jørgensen, P. Linear-Scaling Coupled Cluster with Perturbative Triple Excitations: The Divide–Expand–Consolidate CCSD (T) Model. *J. Chem. Theory Comput.* **2015**, *11*, 2984–2993.

389. Høyvik, I.-M.; Kristensen, K.; Jansik, B.; Jørgensen, P. The Divide-Expand-Consolidate Family of Coupled Cluster Methods: Numerical Illustrations Using Second Order Møller-Plesset Perturbation Theory. *J. Chem. Phys.* **2012**, *136*, 014105.

390. Häser, M.; Almlöf, J. Laplace Transform Techniques in Møller–Plesset Perturbation Theory. *J. Chem. Phys.* **1992**, *96*, 489–494.

391. Ayala, P. Y.; Scuseria, G. E. Linear Scaling Second-Order Møller–Plesset Theory in the Atomic Orbital Basis for Large Molecular Systems. *J. Chem. Phys.* **1999**, *110*, 3660–3671.

392. Doser, B.; Lambrecht, D. S.; Ochsenfeld, C. Tighter Multipole-Based Integral Estimates and Parallel Implementation of Linear-Scaling AO–MP2 theory. *Phys. Chem. Chem. Phys.* **2008**, *10*, 3335–3344.

393. Kottmann, J. S.; Bischoff, F. A. Coupled-Cluster in Real Space. 1. CC2 Ground State Energies Using Multiresolution Analysis. *J. Chem. Theory Comput.* **2017**, *13*, 5945–5955.

394. Riplinger, C.; Sandhoefer, B.; Hansen, A.; Neese, F. Natural Triple Excitations in Local Coupled Cluster Calculations with Pair Natural Orbitals. *J. Chem. Phys.* **2013**, *139*, 134101.

395. Riplinger, C.; Neese, F. An Efficient and Near Linear Scaling Pair Natural Orbital Based Local Coupled Cluster Method. *J. Chem. Phys.* **2013**, *138*, 034106.

396. Edmiston, C.; Krauss, M. Pseudonatural Orbitals as a Basis for the Superposition of Configurations. II. Energy Surface for Linear H3. *J. Chem. Phys.* **1968**, *49*, 192–205.

397. Meyer, W.; Rosmus, P. PNO-CL and CEPA Studies of Electron Correlation Effects. 3. Spectroscopic Constants and Dipole-Moment Functions for Ground-States of First-Row and Second-Row Diatomic Hydrides. *J. Chem. Phys.* **1975**, *63*, 2356–2375.

398. Ahlrichs, R.; ; Driessler, F.; Lischka, H.; Staemmler, V.; Kutzelnigg, W. PNO-CL (Pair-Natural-Orbital Configuration Interaction) AND CEPA-PNO (Coupled Electron Pair Approximation with Pair Natural Orbitals) Calculations of Molecular Systems.2. Molecules BeH2, BH, BH3, CH4, CH-3, NH3 (Planar and Pyramidal), H2O, OH+3, HF and Ne Atom. *J. Chem. Phys.* **1975**, *62*, 1235–1247.

399. Ahlrichs, R.; Keil, F.; Lischka, H.; ; Kutzelnigg, W.; Staemmler, V. PNO-CL (Pair-Natural-Orbital Configuration Interaction) AND CEPA-PNO (Coupled Electron Pair Approximation with Pair Natural Orbitals) Calculations of Molecular Systems.3. Molecules MgH2, AlH3, SiH4, PH3 (Planar and Pyramidal), H2S, HCl, and Ar Atom. *J. Chem. Phys.* **1975**, *63*, 455–463.

400. Ahlrichs, R.; Lischka, H.; B, Z.; Kutzelnigg, W. PNO-CL (Pair-Natural-Orbital Configuration Interaction) and CEPA-PNO (Coupled Electron Pair Approximation with Pair Natural Orbitals) Calculations of Molecular Systems.4. Molecules N2, F2, C2H2, C2H4, and C2H6. *J. Chem. Phys.* **1975**, *63*, 4685–4694.

401. Schütz, M.; Werner, H.-J. Low-Order Scaling Local Electron Correlation Methods. IV. Linear Scaling Local Coupled-Cluster (LCCSD). *J. Chem. Phys.* **2001**, *114*, 661–681.

402. Schütz, M. A New, Fast, Semi-Direct Implementation of Linear Scaling Local Coupled Cluster Theory. *Phys. Chem. Chem. Phys.* **2002**, *4*, 3941–3947.

403. Schütz, M. Low-Order Scaling Local Electron Correlation Methods. V. Connected Triples Beyond (T): Linear Scaling Local CCSDT-1b. *J. Chem. Phys.* **2002**, *116*, 8772–8785.

404. Schmitz, G.; Hättig, C.; Tew, D. P. Explicitly Correlated PNO-MP2 and PNO-CCSD and Their Application to the S66 Set and Large Molecular Systems. *Phys. Chem. Chem. Phys.* **2014**, *16*, 22167–22178.

405. Schmitz, G.; Hättig, C. Perturbative Triples Correction for Local Pair Natural Orbital Based Explicitly Correlated CCSD (F12*) Using Laplace Transformation Techniques. *J. Chem. Phys.* **2016**, *145*, 234107.

406. Riplinger, C.; Pinski, P.; Becker, U.; Valeev, E. F.; Neese, F. Sparse Maps-A Systematic Infrastructure for Reduced-Scaling Electronic Structure Methods. II. Linear Scaling Domain Based Pair Natural Orbital Coupled Cluster Theory. *J. Chem. Phys.* **2016**, *144*, 024109.

407. Pavosevic, F.; Pinski, P.; Riplinger, C.; Neese, F.; Valeev, E. F. SparseMaps-A Systematic Infrastructure for Reduced-Scaling Electronic Structure Methods. IV. Linear-Scaling

Second-Order Explicitly Correlated Energy with Pair Natural Orbitals. *J. Chem. Phys.* **2016**, *144*, 144109.

408. Pavosevic, F.; Peng, C.; Pinski, P.; Riplinger, C.; Neese, F.; Valeev, E. F. SparseMaps-A Systematic Infrastructure for Reduced Scaling Electronic Structure Methods. V. Linear Scaling Explicitly Correlated Coupled-Cluster Method with Pair Natural Orbitals. *J. Chem. Phys.* **2017**, *146*, 174108.

409. Saitow, M.; Becker, U.; Riplinger, C.; Valeev, E. F.; Neese, F. A New Near-Linear Scaling, Efficient and Accurate, Open-Shell Domain-Based Local Pair Natural Orbital Coupled Cluster Singles and Doubles Theory. *J. Chem. Phys.* **2017**, *146*, 164105.

410. Schwilk, M.; Ma, Q.; Köppl, C.; Werner, H.-J. Scalable Electron Correlation Methods. 3. Efficient and Accurate Parallel Local Coupled Cluster with Pair Natural Orbitals (PNO-LCCSD). *J. Chem. Theory Comput.* **2017**, *13*, 3650–3675.

411. Ma, Q.; Schwilk, M.; Köppl, C.; Werner, H.-J. Scalable Electron Correlation Methods. 4. Parallel Explicitly Correlated Local Coupled Cluster with Pair Natural Orbitals (PNO-LCCSD-F12). *J. Chem. Theory Comput.* **2017**, *13*, 4871–4896.

412. Ma, Q.; Werner, H.-J. Scalable Electron Correlation Methods. 5. Parallel Perturbative Triples Correction for Explicitly Correlated Local Coupled Cluster with Pair Natural Orbitals. *J. Chem. Theory Comput.* **2017**, *14*, 198–215.

413. Ma, Q.; Werner, H.-J. Explicitly Correlated Local Coupled-Cluster Methods Using Pair Natural Orbitals. *WIREs Comput. Mol. Sci.* **2018**, *8*, e1371.

414. Tew, D. P.; Helmich, B.; Hättig, C. Local Explicitly Correlated Second-Order Møller–Plesset Perturbation Theory with Pair Natural Orbitals. *J. Chem. Phys.* **2011**, *135*, 074107.

415. Kottmann, J. S.; Bischoff, F. A.; Valeev, E. F. Direct Determination of Optimal Pair-Natural Orbitals in a Real-Space Representation: The Second-Order Moller–Plesset Energy. *J. Chem. Phys.* **2020**, *152*, 074105.

416. Kato, T. On the Eigenfunctions of Many-Particle Systems in Quantum Mechanics. *Commun. Pure Appl. Math.* **1957**, *10*, 151–177.

417. Pack, R. T.; Brown, W. B. Cusp Conditions for Molecular Wavefunctions. *J. Chem. Phys.* **1966**, *45*, 556–559.

418. Kutzelnigg, W.; Morgan, J. D. Rates of Convergence of the Partial-Wave Expansions of Atomic Correlation Energies. *J. Chem. Phys.* **1992**, *96*, 4484–4508.

419. Kutzelnigg, W. R12-Dependent Terms in the Wave-Function as Closed Sums of Partial-Wave Amplitudes for Large-L. *Theoret. Chim. Acta* **1985**, *68*, 445–469.

420. Kong, L.; Bischoff, F. A.; Valeev, E. F. Explicitly Correlated R12/F12 Methods for Electronic Structure. *Chem. Rev.* **2012**, *112*, 75–107.

421. Hättig, C.; Klopper, W.; Köhn, A.; Tew, D. P. Explicitly Correlated Electrons in Molecules. *Chem. Rev.* **2012**, *112*, 4–74.

422. Straatsma, T. P.; McCammon, J. A. *Molecular Design and Modeling: Concepts and Applications Part A: Proteins, Peptides, and Enzymes*; Methods in Enzymology; Academic Press, 1991; Vol. 202; Chapter Theoretical Calculations of Relative Affinities of Binding, pp 497–511.

423. Straatsma, T. P.; Bylaska, E. J.; van Dam, H. H. J.; Govind, N.; de Jong, W. A.; Kowalski, K.; Valiev, M. In *Annual Reports in Computational Chemistry*; Wheeler, R. A., Ed.; Annual Reports in Computational Chemistry; Elsevier, 2011; Vol. 7; Chapter Advances in Scalable Computational Chemistry: NWChem, pp 151–177.

424. Straatsma, T. P.; Philippopoulos, M.; McCammon, J. A. NWChem: Exploiting Parallelism in Molecular Simulations. *Comput. Phys. Commun.* **2000**, *128*, 377–385.

425. Straatsma, T. P.; McCammon, J. A. Load Balancing of Molecular Dynamics Simulation with NWChem. *IBM Syst. J.* **2001**, *40*, 328–341.

426. Straatsma, T. P.; Chavarría-Miranda, D. G. On Eliminating Synchronous Communication in Molecular Simulations to Improve Scalability. *Comput. Phys. Commun.* **2013**, *184*, 2634–2640.

427. Straatsma, T. P.; McCammon, J. A. Load Balancing of Molecular Dynamics Simulation with NWChem. *IBM Syst. J.* **2001**, *40*, 328–341.

428. ARGOS Molecular Dynamics. `http://www.argoscode.org`, [Online; accessed 30-August-2020].

429. Straatsma, T. P.; McCammon, J. A. ARGOS, A Vectorized General Molecular Dynamics Program. *J. Comput. Chem.* **1990**, *11*, 943–951.

430. Svensson, M.; Humbel, S.; Froese, R. D.; Matsubara, T.; Sieber, S.; Morokuma, K. ONIOM: A Multilayered Integrated MO+ MM Method for Geometry Optimizations and Single Point Energy Predictions. A Test for Diels-Alder Reactions and Pt (P (t-Bu) 3) 2+ H2 Oxidative Addition. *J. Phys. Chem.* **1996**, *100*, 19357–19363.

431. Sushko, P. V.; Huang, C.; Govind, N.; Kowalski, K. *Computational Materials Discovery*; 2018; Chapter Embedding Methods in Materials Discovery, pp 87–116.

432. Warshel, A.; Levitt, M. Theoretical Studies of Enzymic Reactions: Dielectric, Electrostatic and Steric Stabilization of the Carbonium Ion in the Reaction of Lysozyme. *J. Mol. Biol.* **1976**, *103*, 227–249.

433. Valiev, M.; Garrett, B. C.; Tsai, M.-K.; Kowalski, K.; Kathmann, S. M.; Schenter, G. K.; Dupuis, M. Hybrid Approach for Free Energy Calculations with High-Level Methods: Application to the SN2 Reaction of CHCl3 and OH- in Water. *J. Chem. Phys.* **2007**, *127*, 051102.

434. Wesolowski, T. A.; Warshel, A. Frozen Density Functional Approach for ab initio Calculations of Solvated Molecules. *J. Phys. Chem.* **1993**, *97*, 8050–8053.

435. Govind, N.; Wang, Y. A.; Carter, E. A. Electronic-Structure Calculations by First-Principles Density-Based Embedding of Explicitly Correlated Systems. *J. Chem. Phys.* **1999**, *110*, 7677–7688.

436. Elliott, P.; Burke, K.; Cohen, M. H.; Wasserman, A. Partition Density-Functional Theory. *Phys. Rev. A* **2010**, *82*, 024501.

437. Huang, C.; Pavone, M.; Carter, E. A. Quantum Mechanical Embedding Theory Based on a Unique Embedding Potential. *J. Chem. Phys.* **2011**, *134*, 154110.

438. Khait, Y. G.; Hoffmann, M. R. *Annual Reports in Computational Chemistry*; Elsevier, 2012; Vol. 8; Chapter On the orthogonality of orbitals in subsystem Kohn–Sham density functional theory, pp 53–70.

439. Bulik, I. W.; Chen, W.; Scuseria, G. E. Electron Correlation in Solids via Density Embedding Theory. *J. Chem. Phys.* **2014**, *141*, 054113.

440. Knizia, G.; Chan, G. K.-L. Density Matrix Embedding: A Strong-Coupling Quantum Embedding Theory. *J. Chem. Theory Comput.* **2013**, *9*, 1428–1432.

441. Seijo, L.; Barandiarán, Z. *Computational Chemistry: Reviews of Current Trends*; 1999; Chapter The Ab Initio Model Potential Method: A Common Strategy for Effective Core Potential and Embedded Cluster Calculations, pp 55–152.

442. Manby, F. R.; Stella, M.; Goodpaster, J. D.; Miller III, T. F. A Simple, Exact Density-Functional-Theory Embedding Scheme. *J. Chem. Theory Comput.* **2012**, *8*, 2564–2568.

443. Hégely, B.; Nagy, P. R.; Ferenczy, G. G.; Kállay, M. Exact Density Functional and Wave Function Embedding Schemes Based on Orbital Localization. *J. Chem. Phys.* **2016**, *145*, 064107.

444. Fornace, M. E.; Lee, J.; Miyamoto, K.; Manby, F. R.; Miller III, T. F. Embedded Mean-Field Theory. *J. Chem. Theory Comput.* **2015**, *11*, 568–580.

445. Lan, T. N.; Kananenka, A. A.; Zgid, D. Communication: Towards ab initio Self-Energy Embedding Theory in Quantum Chemistry. *J. Chem. Phys.* **2015**, *143*, 241102.

446. Inglesfield, J. A Method of Embedding. *J. Phys. C: Solid State Phys.* **1981**, *14*, 3795.

447. Pisani, C.; Dovesi, R.; Nada, R.; Kantorovich, L. Ab initio Hartree–Fock Perturbed-Cluster Treatment of Local Defects in Crystals. *J. Chem. Phys.* **1990**, *92*, 7448–7460.

448. Chibani, W.; Ren, X.; Scheffler, M.; Rinke, P. Self-Consistent Green's Function Embedding for Advanced Electronic Structure Methods Based on a Dynamical Mean-Field Concept. *Phys. Rev. B* **2016**, *93*, 165106.

449. Neugebauer, J. Chromophore-Specific Theoretical Spectroscopy: From Subsystem Density Functional Theory to Mode-Specific Vibrational Spectroscopy. *Phys. Rep.* **2010**, *489*, 1–87.

450. Gomes, A. S. P.; Jacob, C. R. Quantum-Chemical Embedding Methods for Treating Local Electronic Excitations in Complex Chemical Systems. *Annu. Rep. Prog. Chem., Sect. C: Phys. Chem.* **2012**, *108*, 222–277.

451. Jacob, C. R.; Neugebauer, J. Subsystem Density-Functional Theory. *WIREs Comput. Mol. Sci.* **2014**, *4*, 325–362.

452. Wesolowski, T. A.; Shedge, S.; Zhou, X. Frozen-Density Embedding Strategy for Multilevel Simulations of Electronic Structure. *Chem. Rev.* **2015**, *115*, 5891–5928.

453. Sun, Q.; Chan, G. K.-L. Quantum Embedding Theories. *Acc. Chem. Res.* **2016**, *49*, 2705–2712.

454. Huzinaga, S.; Cantu, A. Theory of Separability of Many-Electron Systems. *J. Chem. Phys.* **1971**, *55*, 5543–5549.

455. Francisco, E.; Martín Pendás, A.; Adams, W. Generalized Huzinaga Building-Block Equations for Nonorthogonal Electronic Groups: Relation to the Adams–Gilbert Theory. *J. Chem. Phys.* **1992**, *97*, 6504–6508.

456. Graham, D. S.; Wen, X.; Chulhai, D. V.; Goodpaster, J. D. Robust, Accurate, and Efficient: Quantum Embedding Using the Huzinaga Level-Shift Projection Operator for Complex Systems. *J. Chem. Theory Comput.* **2020**, *16*, 2284–2295.

457. Head, J. D.; Silva, S. J. A Localized Orbitals Based Embedded Cluster Procedure for Modeling Chemisorption on Large Finite Clusters and Infinitely Extended Surfaces. *J. Chem. Phys.* **1996**, *104*, 3244–3259.

458. Whitten, J.; Pakkanen, T. A. Chemisorption Theory for Metallic Surfaces: Electron Localization and the Description of Surface Interactions. *Phys. Rev. B* **1980**, *21*, 4357–4367.

459. Tamukong, P. K.; Khait, Y. G.; Hoffmann, M. R. Density Differences in Embedding Theory with External Orbital Orthogonality. *J. Phys. Chem. A* **2014**, *118*, 9182–9200.

460. Welborn, M.; Manby, F. R.; Miller III, T. F. Even-Handed Subsystem Selection in Projection-Based Embedding. *J. Chem. Phys.* **2018**, *149*, 144101.

461. Culpitt, T.; Brorsen, K. R.; Hammes-Schiffer, S. Communication: Density Functional Theory Embedding with the Orthogonality Constrained Basis Set Expansion Procedure. *J. Chem. Phys.* **2017**, *146*, 211101.

Biographies

Karol Kowalski is a Laboratory Fellow at Pacific Northwest National Laboratory. Karol has received his Ph.D. in theoretical physics from Nicolaus Copernicus University. He has contributed to several computational chemistry packages, including GAMESS and NWChem. He is currently involved in developing a scalable implementation of local coupled-cluster (CC) formulations in NWChemEx. Karol is also an author of Green's function formulations and equation-of-motion CC methods used in the high-accuracy characterization of ionization, transport, and excited-state processes in molecular systems.

Raymond Bair is the Chief Computational Scientist for Applications in the Computing, Environment and Life Sciences Directorate (CELS) and the Computational Science Division (CSD) at Argonne National Laboratory. He is also Senior Scientist at Large in the University ofnArgonne's Laboratory Computing Resource Center (LCRC). Dr. Bair received his B.S. in Chemistry and Mathematics from Westminster College, PA, and his Ph.D. in Theoretical Chemistry from the California Institute of Technology. His research interests span high performance computational methods, new computer architectures, and computational facilities to meet emerging research needs.

Nicholas P. Bauman is a Postdoctoral Research Associate at the Pacific Northwest National Laboratory. Nicholas received his B.S. degree in Chemistry from Michigan Technological University. He completed his Ph.D. in Theoretical/Computational Chemistry at Michigan State University, focusing on developing and applying high-level many-body methods. He then was a Postdoctoral Associate at the University of Florida. He continues to develop and implement many-body methods in the NWChem and NWChemEx packages and apply those methods to a variety of problems.

Jeffery S. Boschen is a Postdoctoral Research Associate with the Ames Laboratory. He received a B.S. in chemistry from Truman State University and a Ph.D. in physical chemistry from Iowa State University. His research interests include highly accurate electronic structure methods, non-adiabatic dynamics, and high performance computing.

Eric J. Bylaska is a research scientist at the Pacific Northwest National Laboratory. Eric received a B.S. degree in applied physics and a B.S. degree in computer science from Michigan Technological University. He then received a Ph.D. in physical chemistry from the University of California, San Diego. Eric's research involves several areas of chemical, geochemical, and environmental research. Eric is the primary author of the ab initio molecular dynamics and band structure modules in the NWChem and NWChemEx program packages. He has also developed several new methods for chemistry and materials simulation including adaptive unstructured finite element electronic structure codes, parallel in time algorithms, efficient implementations of the exact exchange operator, and the EMSL Arrows web application.

Jeff Daily is a Senior Member of Technical Staff at Advanced Micro Devices (AMD). Jeff received his B.S. and Ph.D. degrees in Computer Science from Washington State University where he focused on high performance computing runtimes and applications. While at Pacific Northwest National Laboratory (PNNL), Jeff contributed to the Global Arrays project, the parallel runtime library for the NWChemEx project. Jeff is currently a member of the Machine Learning Software Engineering team at AMD, supporting the development of machine learning frameworks for AMD GPUs. Note: Work on NWChemEx and GlobalArrays was performed while at PNNL.

Wibe A. de Jong is a Senior Scientist at Lawrence Berkeley National Laboratory (LBNL), and a Fellow of the American Association for the Advancement of Science. de Jong received his M.S. in Chemical Physics and Ph.D. in Theoretical Chemistry at the University of Groningen (Netherlands), where he focused on actinide chemistry and the development of parallel relativistic quantum chemistry software. Before moving to LBNL, he led the team developing the NWChem computational chemistry code. de Jong has made contributions to multiple software packages, continues to develop new methods and scalable and fault-tolerant algorithms for, and integrating data science approaches in, NWChem and NWChemEx.

Thom H. Dunning, Jr. is a Research Professor of Chemistry at the University of Washington, a Battelle Fellow at the Pacific Northwest National Laboratory, and a Professor Emeritus of Chemistry at the University of Illinois Champaign-Urbana. Dr. Dunning is a Fellow of the American Physical Society, the American Association for the Advancement of Science, and the American Chemical Society. Dr. Dunning received his B.S. degree from the Missouri University of Science and Technology in 1965 and his Ph.D. degree from the California Institute of Technology in 1970. He is known for his contributions to the development of computational techniques for molecular electronic structure calculations as well as the application of quantum chemistry to the characterization of lasers, combustion chemistry, aqueous clusters, and organic and inorganic chemistry.

Niranjan Govind earned a Ph.D. in Physics from McGill University in Montreal, Canada, followed by postdoctoral work at the Department of Chemistry at the University of California, Los Angeles. He was a staff scientist at Accelrys, Inc. in San Diego before joining his present position at the Pacific Northwest National Laboratory (PNNL), where he is currently a Chief Scientist. His research interests focus on the development and application of time-dependent electronic structure methods for molecules and materials, embedding approaches, relativistic electronic structure methods, X-ray spectroscopies, and ultrafast dynamics including excited-state non-adiabatic molecular dynamics. In addition to his position at PNNL, he also serves on the Editorial Board of Electronic Structure, IOPScience, UK.

Robert Harrison is a Professor of Applied Mathematics and Statistics at Stony Brook University where he also directs the Institute for Advanced Computational Science. He received his Ph.D. from the University of Cambridge in Theoretical Chemistry. He has held staff and leadership positions at Brookhaven National Laboratory, Oak Ridge National Laboratory, Pacific Northwest National Laboratory, and Argonne National Laboratory, and has helped lead the Global Arrays, NWChem, MADNESS, and NWChemEx projects. His research interests are focused on scientific computing and the development of computational chemistry methods for the world's most technologically advanced supercomputers.

Murat Keçeli is an Assistant Computational Scientist at Argonne National Laboratory. He received B.S. and M.S. degrees in physics from Bilkent University in Turkey and a Ph.D. degree in chemical physics from the University of Illinois at Urbana-Champaign. His thesis was focused on size-extensive many-body methods for the vibrational structure of molecules and extended systems. Murat is currently working on memoization and checkpointing infrastructure of NWChemEx. He is also involved in data-driven projects that makes use of leadership class supercomputers.

Kristopher Keipert is a solutions architect at NVIDIA corporation. Previously at Argonne National Laboratory, Kristopher helped design and develop the data resiliency infrastructure in NWChemEx. Kristopher received his Ph.D. in physical chemistry at Iowa State University, where he conducted novel research in dynamics simulations of photochemical systems and high performance computational chemistry algorithms.

Sriram Krishnamoorthy is a computer scientist and Laboratory Fellow in PNNL's High Performance Computing group, and a Research Professor at the School of Electrical Engineering and Computer Science in Washington State University. He earned his B.E. from the College of Engineering, Guindy (Chennai, India) and M.S. and Ph.D. degrees from The Ohio State University. His research focuses on parallel programming models, quantum computing, fault tolerance, and compile-time/runtime optimizations for high-performance computing. In addition, he is the lead for the development of the TAMM software and its application to coupled cluster equations.

Suraj Kumar is a postdoctoral researcher at the National Institute for Research in Computer Science and Automation (INRIA) Paris, France. Prior to this, he was also a postdoctoral researcher at Pacific Northwest National Laboratory, USA. Suraj received his M.E. and Ph.D. degrees in computer science from Indian Institute of Science, Bangalore, India and INRIA Bordeaux, France, respectively. His research interests include tensor algorithms, parallel computing, runtime systems, heterogeneous architectures, linear algebra, and scheduling.

Erdal Mutlu is a Computer Scientist at Pacific Northwest National Laboratory (PNNL). Erdal received his B.Sc. and M.Sc. degrees in computer science and engineering in Sabanci University, Istanbul. Later, he completed his Ph.D. studies in computer science at Koc University, Istanbul, where he focused on software reliability for asynchronous programming models. After joining PNNL as a post-doctoral research associate, he worked on development of the tensor algebra framework, "Tensor Algebra for Many-body Methods (TAMM)", that is being used as the main tensor computation engine in several modern high performance computing systems. Currently, he is working on extending TAMM capabilities for reduced scaling methods developed in NWChemEx.

Bruce Palmer is a research scientist at PNNL. Bruce received his undergraduate degrees in math and chemistry from Bowdoin College and a Ph. D. in chemical physics from Harvard University. Bruce has worked extensively on the statistical properties of fluids as well as working on several high-performance computing libraries. He has been a long time developer of the Global Arrays communication library, used in many quantum chemistry applications, as well as working on frameworks for simulating flow in groundwater and more recently a framework for developing HPC applications to model the power grid.

Ajay Panyala is currently a computer scientist in the High Performance Computing group at PNNL. Ajay received his B.Tech. degree in computer science from Jawaharlal Nehru Technological University, Hyderabad, India. He then completed his Ph.D. in computer science at Louisiana State University where he focused on developing compiler optimizations for high performance computing applications. He is currently one of the lead developers of the Tensor Algebra for Many-body Methods (TAMM) parallel computational infrastructure effort in the DOE ASCR ECP NWChemEx and DOE BES SPEC projects where he closely works with computational chemists on the development of computational chemistry applications. He is primarily responsible for all aspects of TAMM design, development and performance tuning used to drive scalable implementations of many-body methods.

Bo Peng is a computational scientist in the physical and computational science division at Pacific Northwest National Laboratory (PNNL). Bo received his B.S. degree in chemistry from Nankai University in Tianjin, China. After several years research at the Institute of New Energy Material Chemistry, he then moved to the United State, and completed his Ph.D. in physical and theoretical chemistry at the University of Washington in Seattle, where he focused on the theory and algorithm development for ab initio calculations. He then joined PNNL as a Linus Pauling post-doctorate fellow and was promoted to full time staff. Bo has contributed to several quantum chemistry packages. His currently research is focused on the developments and applications of novel many-body theories and their high-performance software libraries.

Ryan M. Richard is a Scientist II at Ames Laboratory working in Professor Theresa L. Windus's group. Ryan received his B.S. degree in chemistry from Cleveland State University and his Ph.D. in chemistry at The Ohio State University. Ryan's research focuses on reducing the time to solution for ab initio methods. In particular he is interested in reduced scaling techniques (e.g., fragment-based methods, local orbital methods) and the use of high-performance computing.

T. P. Straatsma is a Distinguished Research Scientist in the Oak Ridge Leadership Computing Facility OLCF) at the DOE Oak Ridge National Laboratory (ORNL), and an

Adjunct Professor in the Department of Chemistry at the University of Alabama in Tuscaloosa. Straatsma earned his Doctoral and Doctorate degrees in Mathematics and Natural Sciences from the University of Groningen. His specialization is in the design and implementation of advanced modeling and simulation methodologies applied to chemical and biochemical systems. Straatsma is a core developer for the NWChem computational chemistry software suite, and the lead developer for the ARGOS molecular dynamics code and the GronOR massively parallel and GPU-accelerated non-orthogonal configuration interaction package.

Peter V. Sushko is a staff scientist and Materials Sciences group leader in the Physical Sciences Division, Pacific Northwest National Laboratory. Peter received his BSc and MSc degrees in physics from St. Petersburg State University, Russia. He then completed his PhD in physics at University College London, UK, where he was developing embedded cluster approaches for simulating electronic properties of point defects in ionic materials. Peter's research is focused on revealing atomic-scale mechanisms of diffusion, defect formation, and chemical reactions, and predicting the effects of disorder and defects on materials properties and functions.

Edward F. Valeev received M.Sc. in Chemistry in 1996 from the Higher Chemistry College of the Russian Academy of Sciences (Moscow, Russia) and Ph.D. in Chemistry in 2000 from the University of Georgia (Athens, GA). He held a research scientist post at the School of Chemistry and Biochemistry at Georgia Tech, with a joint research appointment at the Oak Ridge National Laboratory during 2004-2006. In 2006 he joined the Department of Chemistry at Virginia Tech, where he is now a Professor of Chemistry. Prof. Valeev's interests focus on accurate (many-body) methodology for electronic and molecular structure and high-productivity and high-performance scientific computing.

Marat Valiev is a Computational Scientist at Environmental Molecular Sciences Laboratory at PNNL. Marat received his M.S. and Ph.D. degrees in condensed matter physics at University of Connecticut, where he was focused on development of novel field based formulations for density functional theory. His current research interests revolve around the development of new theoretical methods for molecular simulations of complex systems, including their implementation for high performance computational codes. Marat is core member of NWChem computational chemistry project and a primary developer of QM/MM module in NWChem.

Hubertus J. J. van Dam is an HPC Application Architect at Brookhaven National Laboratory. Hubertus received his B.Sc. in chemical engineering from the Fontys University of Applied Sciences, and his M.Sc. in chemistry from Radboud University. He received his PhD in theoretical chemistry from Utrecht University for research on multi-reference correlation methods. Hubertus has contributed to multiple electronic structure packages.

Jonathan M. Waldrop is a Postdoctoral Research Associate with Ames Laboratory. Jonathan received his B.Sc. in chemistry from Mercer University. He then received a Ph.D. in chemistry at Auburn University under the supervision of Prof. Konrad Patkowski, focusing on electronic structure theory development and application. His research has focused on extensions of symmetry-adapted perturbation theory (SAPT), and currently centers on the implementation and development of projector-based embedding within NWChemEx.

David B. Williams-Young is a Research Scientist in the Computational Research Division at Lawrence Berkeley National Laboratory. In 2013, David received his B.S. in Chemistry and Mathematics from the Indiana University of Pennsylvania, and in 2018, his Ph.D. in Chemistry from the University of Washington. David's dissertation work focused on the development of high-performance algorithms for the treatment of light-matter interaction and relativistic effects in molecular systems. David has contributed to quantum chemistry software packages, including Gaussian, Chronus Quantum, and NWChemEx. Currently, the primary focus of David's research is in the development of numerical solvers for large scale electronic structure problems on emerging and massively parallel computing architectures.

Chao Yang is a senior scientist in the Computational Research Division at Lawrence Berkeley National Laboratory (LBNL). He received his B.S in Computer Science and Mathematics from Central Missouri State University, M.A. in Mathematics from University of Kansas and Ph.D in Applied Mathematics from Rice University. He was the 1999 Householder fellow at Oak Ridge National Laboratory. He joined LBNL in 2000. His research focuses on numerical linear algebra, high performance computing with applications in computational chemistry. He has contributed to eigensolvers and nonlinear equation solvers for the NWChemEx project.

Marcin Zalewski is a software engineer at Nvidia. Marcin received his B.A. and M.S. degree in computer science from Rensselaer Polytechnic Institute, and his Ph.D. degree in computer science from Chalmers University of Technology where he worked on topics in generic programming in C++. Marcin worked on research topics in large-scale distributed HPC applications at Indiana University and Pacific Northwest National Laboratory prior to joining Nvidia.

Theresa L. Windus is a Distinguished Professor of Chemistry at Iowa State University (ISU), an Associate with Ames Laboratory, an ISU Liberal Arts and Sciences Dean's Professor, and a Fellow of the American Chemical Society. Theresa received her B.A. degrees in chemistry, mathematics and computer science from Minot State University. She then completed her Ph.D. in physical chemistry at Iowa State University where she focused on developing high performance algorithms. Theresa has contributed to multiple chemistry packages and currently develops new methods and algorithms for high performance computational chemistry as the director of the NWChemEx project as well as applying those techniques to both basic and applied research.

Table of Contents image