Trends in atomistic simulation software usage [1.3]

Leopold Talirz[1,2,3*], **Luca M. Ghiringhelli**[4], **Berend Smit**[1,3]

[1]Laboratory of Molecular Simulation (LSMO), Institut des Sciences et Ingenierie Chimiques, Valais, École Polytechnique Fédérale de Lausanne, CH-1951 Sion, Switzerland; [2]Theory and Simulation of Materials (THEOS), Faculté des Sciences et Techniques de l'Ingénieur, École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland; [3]National Centre for Computational Design and Discovery of Novel Materials (MARVEL), École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland; [4]The NOMAD Laboratory at the Fritz Haber Institute of the Max Planck Society and Humboldt University, Berlin, Germany

This LiveCoMS document is maintained online on GitHub at https://github.com/ltalirz/livecoms-atomistic-software; to provide feedback, suggestions, or help improve it, please visit the GitHub repository and participate via the issue tracker.

This version dated October 14, 2021

Abstract Driven by the unprecedented computational power available to scientific research, the use of computers in solid-state physics, chemistry and materials science has been on a continuous rise. This review focuses on the software used for the simulation of matter at the atomic scale. We provide a comprehensive overview of major codes in the field, and analyze how citations to these codes in the academic literature have evolved since 2010. An interactive version of the underlying data set is available at https://atomistic.software.

***For correspondence:**
leopold.talirz@gmail.com (LT)

1 Introduction

Scientists today have unprecedented access to computational power. This statement would be unremarkable, were it not for the extent to which computational power has exploded. Figure 1 shows the performance ranking of the top 500 supercomputers in the world over the last decades, including the performance of the top-ranked machine, the machine at the bottom of the list, and the sum of all 500. Remarkably, the least-squares fit to the sum (green line) corresponds to a growth rate of ∼75% per year, which translates to a performance increase of more than one *million* times over the last 25 years. Similar improvements in commodity hardware mean that many of 2020's laptop computers would have made the top 500 list of the early 2000s [1].

First versions of quantum-chemistry codes, such as Gaussian [2], were already released in the 1970s, followed by force-field codes, such as GROMOS [3], and periodic density-functional theory (DFT) codes, such as CASTEP [4], in the 1980s and 1990s. In other words, many of these atomistic simulation engines have been around during this explosion of computational power, continuously evolving to take advantage of new algorithms, processor architectures, increasing parallelism and, more recently, dedicated accelerator hardware. Over time, they have developed from instruments for specialists to proven and tested tools in the arsenal of practitioners in physics, chemistry, and materials science.

Records of the pervasive use of these tools can be found in the scientific literature. In a 2014 survey, van Noorden et al. found that 12 of the top 100 most cited papers of all time were on density-functional theory [5]. As with other exam-

Title	Authors	Journal	# cited
Generalized gradient approximation made simple	Perdew, Burke, Ernzerhof	PRL (1996)	108 099
Development of the Colle-Salvetti correlation-energy formula ...	Lee, Yang, Parr	PRB (1988)	77 473
Efficient iterative schemes for ab initio total-energy calculations ...	Kresse, Furthmüller	PRB (1996)	58 176
Projector augmented-wave method	Blöchl	PRB (1994)	43 455
Self-consistent equations including exchange and correlation ...	Kohn, Sham	PR (1965)	42 795
From ultrasoft pseudopotentials to the projector augmented-wave ...	Kresse, Joubert	PRB (1999)	42 485
Special points for Brillouin-zone Integrations	Monkhorst, Pack	PRB (1976)	41 232
Density-functional exchange-energy approximation with correct ...	Becke	PRA (1988)	41 142
Inhomogeneous electron gas	Hohenberg, Kohn	PRB (1964)	35 445
Ab initio molecular dynamics for liquid metals	Kresse, Hafner	PRB (1993)	23 192

Table 1. Top ten most highly cited articles published by the American Physical Society, all of which deal with density functional theory and its practical application. Data collected from the Web of Science on June 16th, 2021.

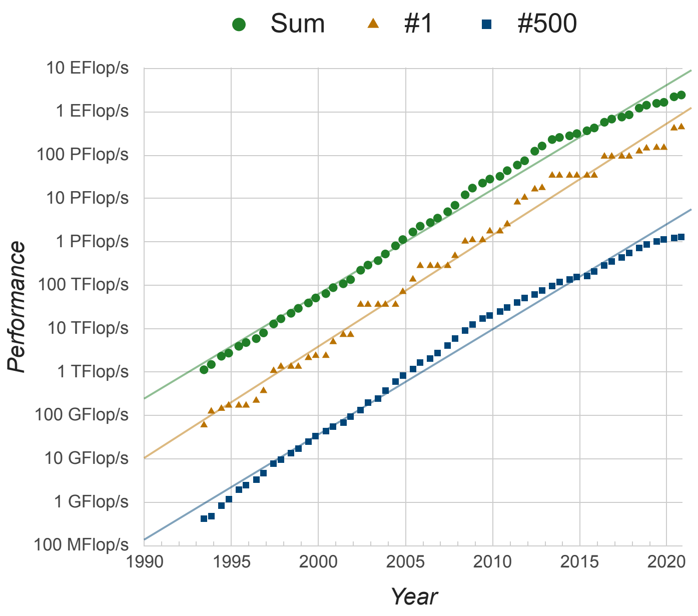


Figure 1. Performance of the top 500 supercomputers in the world from 1993 to 2020 in solving linear equations (Linpack benchmark), measured in 64bit floating point operations per second (FLOP/s). Shown are the sum of the entire list (green dots), the performance of the top machine (brown triangles), and performance of the bottom of the list (blue squares), together with least squares fits. Adapted from top500.org/statistics/perfdevel.

ples in van Noorden's list, the flood of citations are indicative of papers being cited by the many practitioners (here: of density-functional theory) rather than the few method developers. If one focuses the analysis on articles published in physics journals, the footprint of density-functional theory grows even further: For example, table 1 shows the top ten most cited papers published by journals of the American Physical Society. All of them are related to density-functional theory and its application.

There used to be a time when it was commonplace for computational condensed-matter physicists and quantum chemists to write their own electronic-structure code, and many of the atomistic-simulation engines that are in broad use today have started this way. Over the years, however, many of these engines have developed into complex software distributions. Table 2 shows counts for the lines of code in some of today's popular open-source simulation engines: they range from hundreds of thousands to millions of lines of code, typically written in Fortran or C++, with similar numbers being reported for commercial packages [19]. While statistics like these are by no means accurate measures of code complexity (and developers follow different approaches to packaging and outsourcing of functionality to external libraries), they nevertheless suggest that many of these code bases are too large to be sustained by any single person.

This poses important questions for how to sustain these software projects going forward: questions of funding, business models, and software licenses. Proponents of the open-source route argue that it democratizes research and education by removing barriers for both users and developers, and that science carried out with commercial software is harder to verify and reproduce [20, 21, 1]. The open-source model can also be adopted irrespective of the size of a code's target user group, while commercial activity tends to require a

Software	Version	Main language	Contributors 2020	Lines of code
NWChem [6]	7.0.2	Fortran	11	5 034 139*
LAMMPS [7]	27May2021	C++	51	1 039 872
CP2K [8]	8.2	Fortran	38	911 367
ABINIT [9]	9.4.2	Fortran	21	719 048
Quantum ESPRESSO [10]	6.7.0	Fortran	43	604 666
GROMACS [11]	2021.2	C++	25	589 108
Psi4 [12]	1.3.2	C++	34	496 858
SIESTA [13]	5.0.0-alpha	Fortran	4	411 829
Octopus [14]	10.5	Fortran	10	297 825
OpenMM [15]	7.5.1	C++	21	256 452
xtb [16]	6.4.1	Fortran	18	124 344
Linux kernel	5.13	C	>4000 [17]	20 904 410

Table 2. Counting lines of code for 11 popular open-source atomistic simulation engines (and the Linux kernel for comparison), using the latest releases as of June 2021. Line counts are determined by cloc v1.6.0 [18] and exclude blank lines, comments, and markup languages (detailed reports in the supporting information). Contributors for the year 2020 were determined by counting the number of different committers to the source code from January 1st 2020 to January 1st 2021 (numbers for the Linux kernel are from 2019). * Roughly 3 million lines of code of NWChem are computer-generated.

minimal market size. On the other hand, the strong focus on innovation and development often found in open-source scientific software can negatively impact usability and quality of documentation [22]. Proponents of commercial licenses argue that making academic software accessible to a broader community is a technical task best left to professional software engineers, and that the resulting gain in scientific productivity can easily outweigh the license fees that pay for it [19]. We note that open-source and commercial activity do not necessarily exclude each other: numerous software companies are built around an open-source core, and we start seeing first examples in atomistic modelling as well (e.g. Molcas/OpenMolcas [23]). Overall, it has been suggested that "scientific publications are a more sound metric [of the scientific impact of software] than either the price of a product or whether its source code is available in the public domain" [19]. Providing a peek into this citation record is one of the reasons for creating the atomistic.software collection.

The other reason is a practical one: When young scientists start their first research project in atomistic simulations, they often have no grasp of the extent of this software ecosystem, let alone of current trends in the field – at least this is the personal experience of the authors of this review. Software choices in research groups are therefore often informed by what other members of the group already use. This makes sense: colleagues have vetted the code for the type of problems the research group is working on, and built up expertise around which of the many knobs to turn in order to find the sweet spot between efficiency and accuracy.

But what if that code is no longer actively developed? What if there was another code that was better suited to solve the specific research problem at hand? That had a larger user/developer community? Was free instead of commercial? Was open source instead of closed source? The goal of the atomistic.software collection is to provide a comprehensive overview of all major atomistic simulation engines (cf. Figure 2), and to help newcomers to the field as well as experienced practitioners and software developers find better answers to some of these questions.

Readers interested mainly in the results are invited to go straight to the atomistic.software web site that displays the data discussed in this review. For those wanting to know more, the following sections provide details on the methodologies used, and discuss some of the trends that can be observed.

2 The atomistic.software collection

2.1 Overview

The atomistic.software collection draws upon existing lists of atomistic simulation codes [24, 25, 26, 27, 28], in particular the "Major codes in electronic-structure theory, quantum chemistry, and molecular-dynamics" [29] maintained by the NOMAD Centre of Excellence from 2017-2019. It enriches these with annual citation data from the Google Scholar search engine, which provides an overview of the current usage landscape as well as ongoing trends, both at the level of individual codes and at the ecosystem level.

The overview table Fig. 3 lists all codes in the data set, ordered by how often they are referenced by articles indexed in Google Scholar. Clicking on the citation count opens the corresponding query on Google Scholar, so users

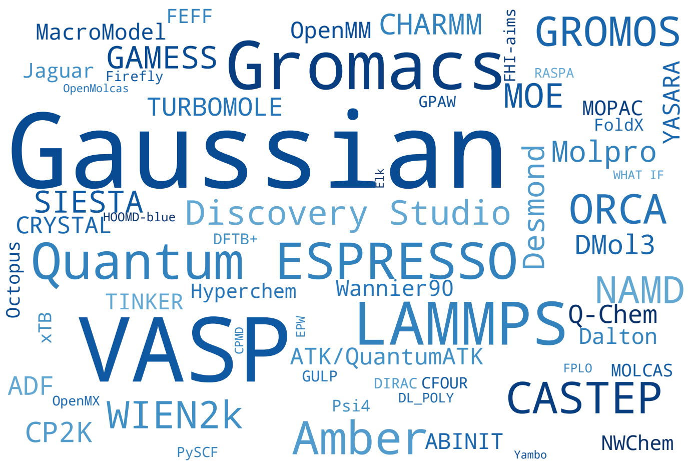
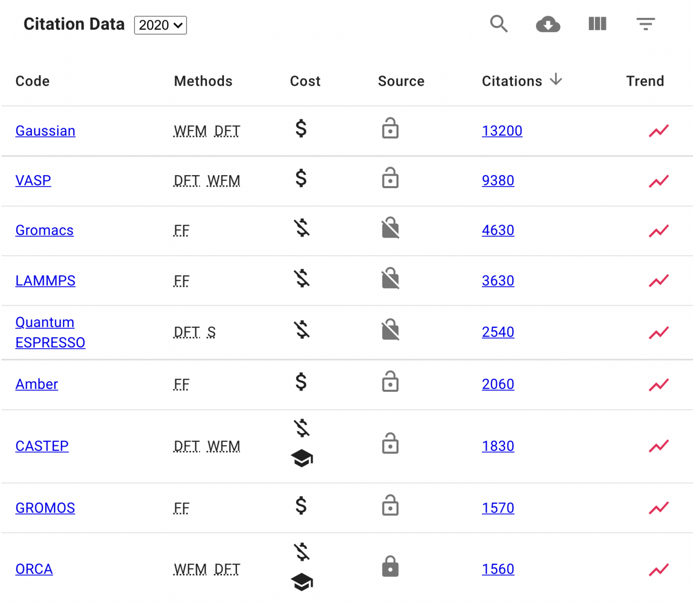
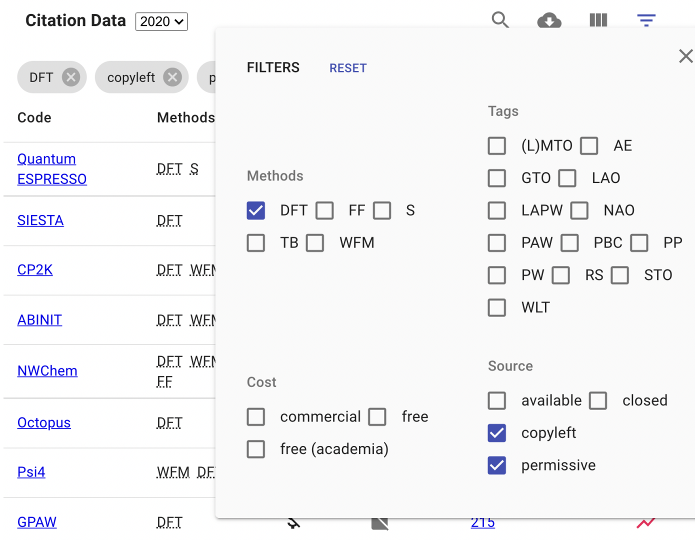


Figure 2. Highly cited atomistic simulation engines in the scientific literature. Font size scaled (approximately) by the number of citations during the year 2020 as reported by Google Scholar.


Citation Data 2020

Code	Methods	Cost	Source	Citations ↓	Trend
Gaussian	WFM DFT	$	🔓	13200	📈
VASP	DFT WFM	$	🔓	9380	📈
Gromacs	FF	$̸	🚫	4630	📈
LAMMPS	FF	$̸	🚫	3630	📈
Quantum ESPRESSO	DFT S	$̸	🚫	2540	📈
Amber	FF	$	🔓	2060	📈
CASTEP	DFT WFM	$̸ 🎓	🔓	1830	📈
GROMOS	FF	$	🔓	1570	📈
ORCA	WFM DFT	$̸ 🎓	🔒	1560	📈

Figure 3. Overview table of atomistic simulation engines, sorted by how often they are referenced on Google Scholar during the previous year (here: 2020). A drop-down menu provides access to annual citation data reaching back to the year 2010.


Citation Data 2020

DFT ⊗ copyleft ⊗ p

Code	Methods
Quantum ESPRESSO	DFT S
SIESTA	DFT
CP2K	DFT WFN
ABINIT	DFT WFN
NWChem	DFT WFN FF
Octopus	DFT
Psi4	WFM DF
GPAW	DFT

FILTERS RESET

Methods
[✓ DFT] FF S
TB WFM

Cost
commercial free
free (academia)

Tags
(L)MTO AE
GTO LAO
LAPW NAO
PAW PBC PP
PW RS STO
WLT

Source
available closed
[✓ copyleft]
[✓ permissive]

Figure 4. Filtering for density-functional theory codes with both permissive (P) and copyleft (CL) open-source licenses.

can sift through the references one by one and discover what science is being done with this software. Clicking on the name of the code instead opens the code's homepage for further information.

Users can filter codes by the methods or basis sets they use, and select only codes that are commercial, free or open-source (Fig. 4). Hovering with the mouse over any abbreviation in the list opens a tool-tip with an explanation. Further metadata include information on which range of the periodic table the code covers, available installation routes, support for parallelization/acceleration, support for standard APIs, and the availability of benchmarks. In order not to clutter the interface, not all metadata is displayed by default but columns can be added/removed via the "View Columns" button.

Besides the generic overview, each engine comes with a citation "trend" over the last couple of years, which serves as an indicator of how its user community has developed over time (Figure 5).

Finally, the statistics page looks at the top codes by citation growth, indicating a rapidly growing user community. Ranking by absolute growth naturally favors established codes, while considering relative growth provides insight into the dynamics of new contenders in the list. Ideally, codes rank highly in both metrics as is currently the case, e.g., for Desmond [30] and OpenMM [15].

A word of caution: The popularity of a code is a factor of many variables (starting, e.g., with the size of the target audience) – please do not choose the code for your next research project merely based on its ranking in this list. atomistic.software links to the scholar query for the papers citing

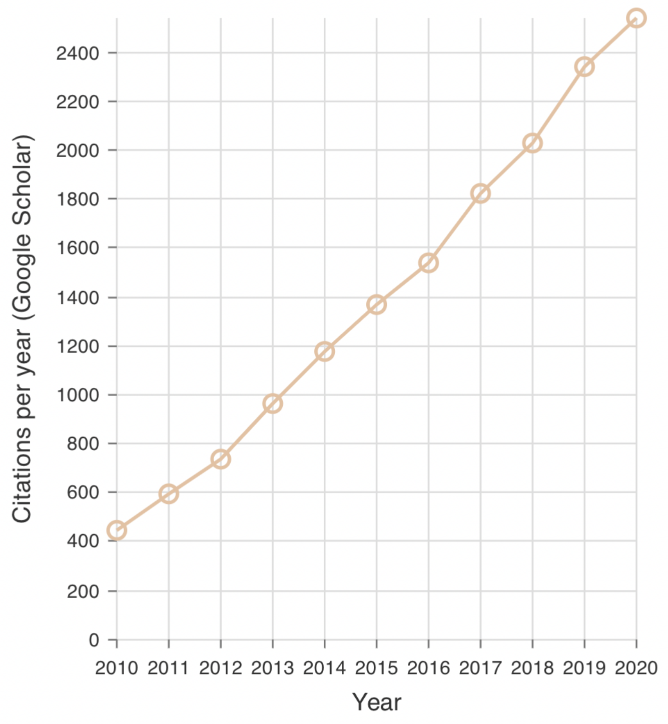
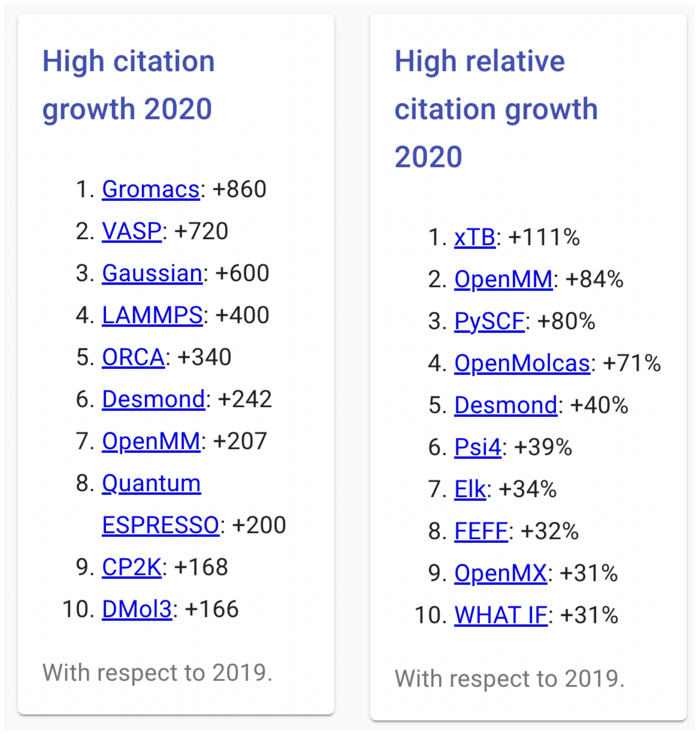


Figure 5. Citation trend for the Quantum ESPRESSO code.



High citation growth 2020

1. Gromacs: +860
2. VASP: +720
3. Gaussian: +600
4. LAMMPS: +400
5. ORCA: +340
6. Desmond: +242
7. OpenMM: +207
8. Quantum ESPRESSO: +200
9. CP2K: +168
10. DMol3: +166

With respect to 2019.

High relative citation growth 2020

1. xTB: +111%
2. OpenMM: +84%
3. PySCF: +80%
4. OpenMolcas: +71%
5. Desmond: +40%
6. Psi4: +39%
7. Elk: +34%
8. FEFF: +32%
9. OpenMX: +31%
10. WHAT IF: +31%

With respect to 2019.

Figure 6. 2020 rankings for absolute and relative citation growth with respect to the previous year.

the code, thus making it quick and easy to get an impression of the research the code is currently used for. Yet, certain aspects of a code are likely to correlate with popularity, such as how much used/tested the software is; how many Q&A resources one is likely to find online or how much tooling there is likely around this code, etc. From a software-developer perspective, knowing the popularity of a code can be useful for gauging the potential impact of supporting the code in your own tool (such as a workflow manager, visualization software, . . .). And, finally, the citation trend provides an interesting peek into the future – is the user community growing, stagnating or decreasing?

2.2 Scope

atomistic.software uses the following working definition of an atomistic simulation engine:

> A piece of software that, given two sets of atomic elements and positions (and, possibly, bond network), can compute their relative internal energies. In almost all cases, engines will also be able to compute the derivative of the energy with respect to the positions, i.e. the forces on the atoms, and thus be able to perform tasks like geometry optimizations or molecular dynamics.

This covers the Density-Functional Theory (`DFT`), Wave-Function Methods (`WFM`), Quantum Monte Carlo (`QMC`), Tight-Binding (`TB`), and Force-Field (`FF`) categories. Codes in the Spectroscopy (`S`) category are not necessarily simulation engines in the above sense, but compute the response of a given atomic structure to an external excitation (via photons, electrons, . . .).

atomistic.software aims to be a *comprehensive* list of all major atomistic simulation engines, with annual updates going forward. Since there is a long tail of simulation engines with a limited user base, a relevance criterion is introduced in order to keep maintenance of the list manageable. The criterion has been set to having at least one year with 100 citations or more. The value of 100 is not set in stone and could be re-evaluated in the future, once the list has had some time to consolidate. A "watch list" is kept of codes that do not yet meet the criterion.

2.3 Methodology & Limitations

Approximate citation counts are obtained from Google Scholar as follows:

1. Search for name of the code and the last name of a representative developer who is a coauthor of all key publications on the software (vast majority of codes). If no

such coauthor exists, this is easily extended to searching for the presence of one of multiple author names (or a company name for commercial codes).

2. When the name of the code is too common a search term, additional search terms may be added or citations of a major reference article are counted (minority of codes)

Google Scholar was chosen over alternative sources like the Web of Science or Scopus, since it provides full-text search and is available for free, thus enabling direct links to the queries. The supporting information contains a case study comparing citation counts from the search-based Google Scholar approach against counting the citations of reference papers (both in Google Scholar and the Web of Science).

Owing to the lack of standardization in today's software citation practices [31, 32], the citation counts reported here are necessarily approximate. Shortcomings include the following:

- While spot checks have been performed to weed out false matches (and reports on the ltalirz/atomistic-software GitHub repository are highly welcome), details of the query can have significant impact on the number of results. This means, in particular, that the ranking by absolute number of citations is not set in stone and may be subject to change if more accurate search terms are identified.
- Citation counts reported by Google Scholar are not entirely static, even for years that lie in the past. Reasons may include new publishers being indexed, more text being extracted, different citations being disambiguated, or even the heuristic evolving that predicts the total number of results. In our experience, citation data for the previous year can be subject to significant (upwards) fluctuation, while citation data for years further in the past are quite stable. For this reason, for each data point the date of collection is recorded in the source code repository.
- Counting citations does not directly measure how often simulation codes are *used* but how often they are *referenced* in the scientific literature. This may involve some systematic bias, for example if popular codes are more likely to be mentioned without being used, or if a software targets industrial users who may be less likely to publish their results.

The caveats listed above mainly affect the *absolute* number of citations reported, and thus the ranking of codes. Citation trends on the individual code level should be more robust, and potential shortcomings in that domain (e.g. missing citations to a new reference paper with different authors) can be addressed by adapting the corresponding query.

The categorization of codes in terms of methods, tags and licenses is an evolution of the classification devised by the NOMAD list [29]. For the sake of this data set, the following terminology has been adopted:

- *commercial*: payment required to obtain the software[1]
- *free for academic use*: free for academics around the world[2]
- *free*: free to use for anyone, possibly after registration
- *source available*: source code available either for free or against payment
- *open-source*: open-source license approved by the Open Source Initiative (OSI, https://opensource.org/)

We note that license terms can (and sometimes do) change over time. This is currently not reflected in this data set (only the latest license terms are recorded), but could be taken into account in future updates.

All data, as well as the source code of the web application running on atomistic.software are hosted in the ltalirz/atomistic-software GitHub repository. The data is released under version 4 of the Creative Commons Share-Alike Attribution International License (CC-BY-SA). The web application is written in JavaScript using the React framework (reactjs.org) and released under version 3 of the Affero General Public License (AGPL).

2.4 Trends

Extensive cross-checks of atomistic.software against other lists [24, 25, 26, 27, 28] suggest that the collection is already fairly complete, and can thus enable a look at the landscape of atomistic simulation software as a whole. Today's atomistic simulation engines are highly sophisticated pieces of software that each take many human-years of development, and developers have chosen different routes to support these efforts: from commercial to free, from closed source to open, and many shades of grey in between. One question we can ask is: How do commercial codes fare versus their free competitors?

Figure 7 compares the compound citations to commercial and free codes. It illustrates that commercial codes are alive and well: they are ahead in terms of citations gathered, and have been ahead throughout the last decade, with Gaussian [33] and VASP [34] together accounting for more than half of all citations of the 23 commercial codes. At the same time, citations of the 40 free codes (including those that are only free for academic use) have been growing roughly at the

[1]This definition of *commercial* does not necessarily mean *for profit* – research groups may price software below the actual development costs. Furthermore, commercial licenses may exempt specific groups from payment, e.g. based on country of residence, membership in consortia, etc.

[2]Some academic licenses may explicitly exclude researchers from specific countries.

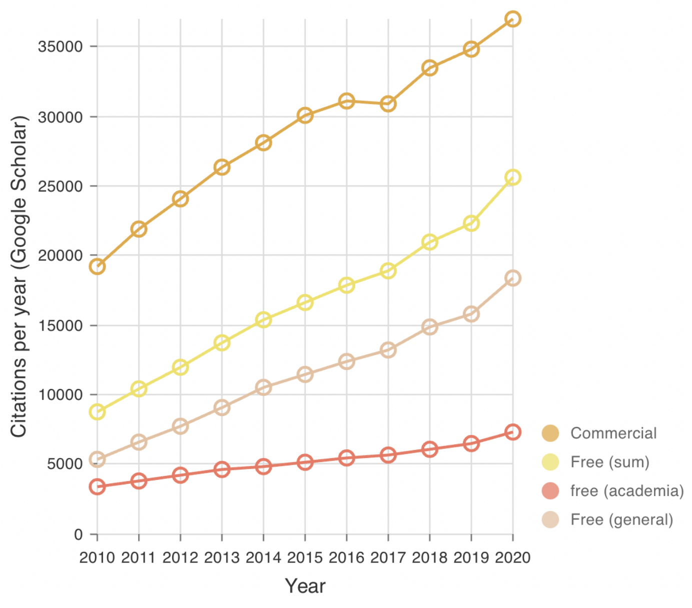


Figure 7. Citations of commercial vs free codes, including a breakdown into those free for general use and those free for academic use only.

same *absolute* rate, mostly driven by the codes that are free for general use.

We note some caveats that apply to this statistic:

- The current dataset only records the *latest* license conditions, while some codes (e.g. CASTEP [35] or Dalton [36]) have moved to more open license terms over time, thus switching categories.
- For codes that are free for academic use only, some researchers may prefer to use the commercial version (e.g. using CASTEP through Biovia's Materials Studio software [37]).

It seems safe to conclude, however, that – while commercial codes remain highly popular – free codes are slowly gaining market share.

Another important question concerns source-code availability, which is relevant for the ability of researchers to independently verify published calculations and pin down bugs. Figure 8 shows that consistently at least ∼90% of citations went to engines whose source code is available, strongly dominating over the 12 closed-source codes, whose citations have stagnated[3] during the 2010s. We recall here that counting citations in the scientific literature places a focus on the usage in academia, and that usage patterns in industry may differ from the trends identified here.

While citations to source-available engines have grown by ∼130% since 2010, citations to the 24 open-source en-

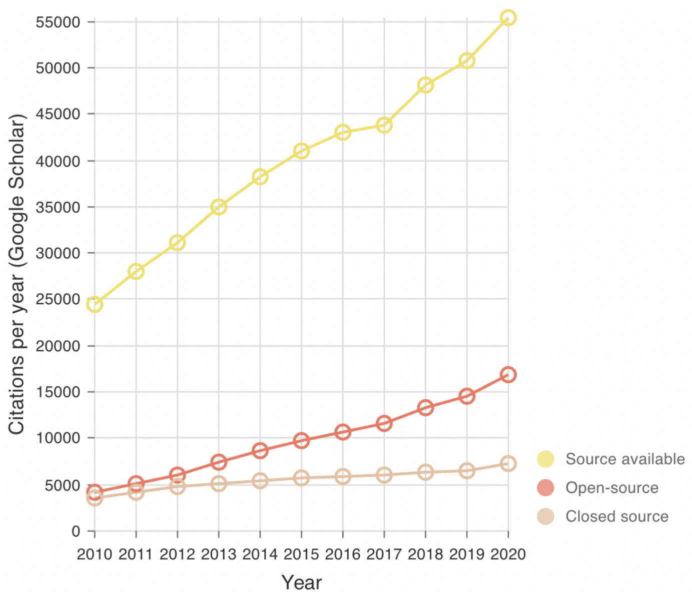


Figure 8. Citations by source code availability. "Source available" includes all engines whose source code can be obtained for free or for a fee. "Open-source" includes only OSI-approved licenses.

License	Version	Copyleft	Published
MIT			1980s
GPL	2	X	1991
LGPL	2.1	X	1999
BSD 3-clause	2		1999
Apache	2		2004
GPL	3	X	2007
LGPL	3	X	2007
ECL	2		2006

Table 3. OSI-approved open-source licenses used in the collection. See the SPDX license list at spdx.org/licenses for the full license terms corresponding to the abbreviations.

[3]One notable exception is the closed-source ORCA code [38] that is free for academic use.

gines within that group rose by >300% within the same time frame, gaining market share. In this context, it is useful to recall that the development of several engines on the lists predates the open-source movement and the creation of many of the open-source licenses that are in broad use today (see Table 3). atomistic.software distinguishes between

- *copyleft* open-source licenses, such as the GPL family, which require[4] derivative software to be distributed under the same open-source license (thus also called *share-alike* or *viral* licenses), and
- *permissive* licenses, such as the BSD, Apache, and MIT licenses, which permit relicensing of derivative works.

The enforced sharing of improvements in derivative works can be a competitive advantage of adopting a viral license. It can also be one path towards financial revenue when companies seek a separate license agreement that allows them to keep derivative works proprietary. Other developers may want to maximize impact of their software by lowering the barrier for adoption across the board, and thus prefer permissive licenses. Overall, the choice of license is highly nuanced and an extensive discussion is beyond the scope of this article (interested readers are referred to choosealicense.com) but it may be instructive to observe the choices made by the codes in the collection.

Out of the 24 open-source codes in the atomistic.software collection, the majority (20) adopt the GPL or LGPL license. The four codes that are distributed under *permissive* licenses (NWChem [6], OpenMM [15], RASPA [39] and PySCF [40]) either switched to this licensing scheme in the late 2000s or 2010s or started being developed during that time. This indicates that the use of permissive licenses is a recent phenomenon in the space of atomistic simulation engines, and may follow in the footsteps of the open-source community at large which is exhibiting a similar trend: according to an analysis of over 4 million open-source packages by WhiteSource [41], the use of permissive open-source licenses has nearly doubled from 41% in 2012 to 76% in 2020, with the Apache and MIT licenses alone accounting for more than half of all licenses that year.

So much about the differences between licensing models. Overall, citations of atomistic simulation engines in the collection have grown at an annual compound growth rate of ∼8%, roughly twice the 4% growth rate seen in the publication of peer-reviewed articles in science and engineering over the last decade [42]. While part of this difference may reflect changing citation practises[5], it likely indicates an increasing adoption of (atomistic) computational materials science throughout the scientific literature.

[4]Under specific circumstances, which differ significantly between the GPL and the LGPL.

[5]According to Mammola et al., the length of reference lists in ecology journals has been increasing by ∼2% per year over the last two decades. [43]

3 Conclusions & Outlook

At the time of writing, the atomistic.software collection contains over 60 simulation engines that each gather >100 citations per year, some several hundreds or thousands. Overall, this review paints a bright future for the field of atomistic simulation: a growing variety of both commercial and free software to choose from, citation growth rates that substantially outpace the rest of the scientific literature, and a forecasted trillion dollar market potential for a digitally-driven materials revolution [44]. There are, however, some challenges ahead as well.

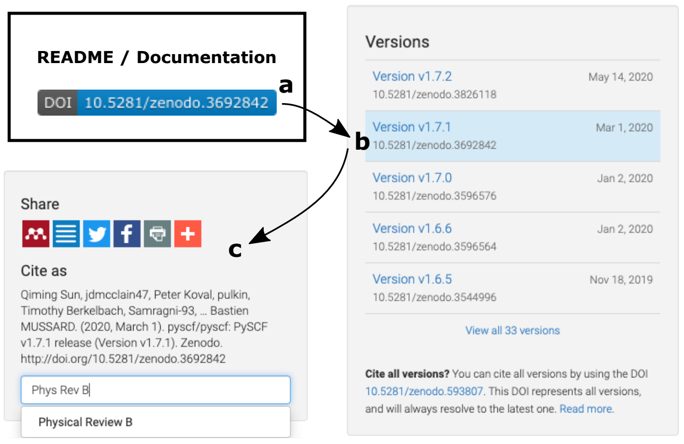
The continued slowdown in single-core performance scaling[6] creates a powerful driving force for the specialization of computer hardware. Small- to medium-size development teams often lack the expertise or the resources to adapt their code base to an ever growing number of hardware accelerators and are at risk of falling behind. One way of approaching this issue is to try and identify low-level, performance-critical primitives that are needed by multiple codes. These primitives can then be bundled into domain-specific libraries, such as libxc [45], libint [46], ELSI [47], SIRIUS [48], M-A-D-N-E-S-S [49], or TiledArray [50] that are ported to and optimized for the various accelerator architectures by HPC specialists.

Another issue that requires attention is the one of software citation. With the increasing role that software plays in advancing science, it is crucial that credit for the creation of software is attributed adequately and accurately. One reason why the citation counts in atomistic.software are approximate is that software citation in the field of atomistic simulations comes in many different forms:

- references of papers that summarize recent developments of the software,
- references of papers that describe the implementation of specific methods,
- references to the home page of the code, or even just
- mentioning the code by name in the main text, possibly followed by a key author or company in parentheses,

similar to what has been found in the field of biology [51]. Furthermore, there is no standardized way of expressing whether a specific version of the software was used or whether the software was referenced as a general concept, e.g. as part of an enumeration of different codes like in this review.

[6]While supercomputers have historically been able to escape this trend by driving up parallelization, Figure 1 suggests that the era of exponentially scaling supercomputer performance may also be coming to an end.



README / Documentation

DOI 10.5281/zenodo.3692842

Versions

Version v1.7.2 May 14, 2020
10.5281/zenodo.3826118

Version v1.7.1 Mar 1, 2020
10.5281/zenodo.3692842

Version v1.7.0 Jan 2, 2020
10.5281/zenodo.3596576

Version v1.6.6 Jan 2, 2020
10.5281/zenodo.3596564

Version v1.6.5 Nov 18, 2019
10.5281/zenodo.3544996

View all 33 versions

Cite all versions? You can cite all versions by using the DOI 10.5281/zenodo.593807. This DOI represents all versions, and will always resolve to the latest one. Read more.

Cite as

Qiming Sun, jdmcclain47, Peter Koval, pulkin, Timothy Berkelbach, Samragni-93, ... Bastien MUSSARD. (2020, March 1). pyscf/pyscf: PySCF v1.7.1 release (Version v1.7.1). Zenodo. http://doi.org/10.5281/zenodo.3692842

Phys Rev B

Physical Review B

Figure 9. User flow for citation via Zenodo-GitHub integration. (a) User clicks on DOI badge in the README file of the source code repository or in the documentation of the software. (b) User is redirected to the landing page of the Zenodo software record, where they can pick the version they used. (c) User enters the desired citation style and copies the citation [54] into their manuscript or downloads the citation in a format supported by their reference manager.

In order to encourage adoption of a consistent policy for software citation across disciplines and venues, in 2016 the software citation working group of the FORCE11 coalition (force11.org) issued detailed software citation principles [52], which include the recommendation of citing a unique, persistent identifier that indicates which version of the software has been used. Today, the technological infrastructure for *creating* these identifiers is in place: for example, for software hosted on github.com, the Zenodo-GitHub integration [53] automatically stores the source code of each software release on the Zenodo repository operated by CERN, and mints a document object identifier (DOI) for it.

Figure 9 illustrates how citing such a DOI works from the user's perspective: When code developers place the DOI badge offered by Zenodo in the "how to cite" section of their documentation, users can click on it and be redirected to the landing page of the accompanying Zenodo record. There, they select the DOI corresponding to the version they used – or, if they are referring to the software in general, they can cite the "concept DOI" of the software that represents all versions and always resolves to the latest one. Finally, they can select the desired citation style and copy the citation into their manuscript or download the citation in a format supported by their reference manager.

At least three codes in the atomistic.software collection (PySCF [40, 55], OpenMM [15, 56], and xtb [16, 57]) have already enabled the Zenodo-GitHub integration but none of them mention this in their citation recommendations yet, effectively reducing the functionality of the integration to that of a future-proof backup of individual software versions. This is a common theme found across software records on Zenodo today [31]. Part of the reason may be that Zenodo records are not (yet) indexed by the widely used scholarly search engines, such as the Web of Science, Scopus or Google Scholar. But as researchers are getting increasingly accustomed to using platforms like Zenodo, the Open Science Framework (osf.io) or Figshare (figshare.com) for depositing and citing *data sets*, it seems to be just a matter of time until analogous practices in software citation will reach the mainstream. Zenodo already *does* track citations of its records through publicly available sources such as Crossref and Europe PubMedCentral (and displays them on the record). Trailblazing developers can therefore recommend their users to cite a version-specific Zenodo DOI in *addition* to a review paper, and thereby get valuable statistics on which versions of their software are being used in return. It can also be a convenient way of making a new code citable before a paper has been written on it.

As for the atomistic.software collection, this review only marks the beginning. Going forward, the collection will receive annual updates, including updates of this perpetual review when warranted. Possible directions for further work include

- adding any simulation engines that were missed,
- recording the time evolution of licenses at the level of individual codes, and
- potentially evolving the scope of the collection, e.g. to include software for atomistic visualization or workflow management (although care would need to be taken in order not to lose focus).

Suggestions for future directions as well as updates and corrections of engine metadata (search keywords, tags, distribution channels, accelerator support, supported APIs, benchmarks, ...) are highly welcome, be it through public discussions on the ltalirz/atomistic-software issue tracker, via pull requests to the repository or via private communication to the authors.

4 Author Contributions

Leopold Talirz: Conceptualization, Methodology, Software and Data curation for atomistic.software. Writing- Original Draft, Reviewing and Editing.
Luca M. Ghiringhelli: Conceptualization, Methodology and Data curation for the original static version of the collection. Writing- Reviewing and Editing.
Berend Smit: Supervision, Writing- Reviewing and Editing, Funding acquisition

5 Other Contributions

The authors are grateful to Nicola Marzari and Matthias Scheffler for sharing their original idea of a list of atomistic simulation codes combined with citation data. Thanks also go to Nicola Marzari for providing the idea for table 1 and to Kevin Jablonka for a careful reading of the manuscript. Insightful discussions with Michele Ceriotti, Lars Holm Nielsen, Kevin Jablonka, Nicola Marzari, and Paul Saxe have contributed to the development of this work, as have numerous constructive suggestions by the anonymous reviewers of the manuscript.

For a more detailed description of contributions from the community and others, see the GitHub issue tracking and changelog at https://github.com/ltalirz/livecoms-atomistic-software.

6 Potentially Conflicting Interests

The authors declare no competing interests.

7 Funding Information

This work was supported by the MARVEL National Centre for Competence in Research funded by the Swiss National Science Foundation (grant agreement ID 51NF40-182892), and the “MaGic” project of the European Research Council (grant agreement ID 666983).

Author Information

ORCID:

Leopold Talirz: 0000-0002-1524-5903
Luca M. Ghiringhelli: 0000-0001-5099-3029
Berend Smit: 0000-0003-4653-8562

References

[1] S. Lehtola and A. Karttunen. “Free and Open Source Software for Computational Chemistry Education” 2021. https://doi.org/10.33774/chemrxiv-2021-hr1r0.

[2] W. J. Hehre et al. *Gaussian 70*. Quantum Chemistry Program Exchange, 1970.

[3] W. F. van Gunsteren and H. J. C. Berendsen. “Molecular Dynamics: Perspective for Complex Systems”. *Biochemical Society Transactions* 10.5 1982, pp. 301–305. https://doi.org/10.1042/bst0100301.

[4] M. C. Payne et al. “Iterative Minimization Techniques for Ab Initio Total-Energy Calculations: Molecular Dynamics and Conjugate Gradients”. *Reviews of Modern Physics* 64.4 1992, pp. 1045–1097. https://doi.org/10.1103/RevModPhys.64.1045.

[5] R. Van Noorden, B. Maher, and R. Nuzzo. “The Top 100 Papers”. *Nature* 514.7524 2014, pp. 550–3. https://doi.org/10.1038/514550a.

[6] E. Aprà et al. “NWChem: Past, Present, and Future”. *The Journal of Chemical Physics* 152.18 2020, p. 184102. https://doi.org/10.1063/5.0004997.

[7] S. Plimpton. “Fast Parallel Algorithms for Short-Range Molecular Dynamics”. *Journal of Computational Physics* 117.1 1995, pp. 1–19. https://doi.org/10.1006/jcph.1995.1039.

[8] T. D. Kühne et al. “CP2K: An Electronic Structure and Molecular Dynamics Software Package - Quickstep: Efficient and Accurate Electronic Structure Calculations”. *The Journal of Chemical Physics* 152.19 2020, p. 194103. https://doi.org/10.1063/5.0007045.

[9] X. Gonze et al. “The Abinitproject: Impact, Environment and Recent Developments”. *Computer Physics Communications* 248 2020, p. 107042. https://doi.org/10.1016/j.cpc.2019.107042.

[10] P. Giannozzi et al. “Quantum ESPRESSO toward the Exascale”. *The Journal of Chemical Physics* 152.15 2020, p. 154105. https://doi.org/10.1063/5.0005082.

[11] M. J. Abraham et al. “GROMACS: High Performance Molecular Simulations through Multi-Level Parallelism from Laptops to Supercomputers”. *SoftwareX* 1-2 2015, pp. 19–25. https://doi.org/10.1016/j.softx.2015.06.001.

[12] D. G. A. Smith et al. “PSI4 1.4: Open-Source Software for High-Throughput Quantum Chemistry”. *The Journal of Chemical Physics* 152.18 2020, p. 184108. https://doi.org/10.1063/5.0006002.

[13] A. García et al. “Siesta: Recent Developments and Applications”. *The Journal of Chemical Physics* 152.20 2020, p. 204108. https://doi.org/10.1063/5.0005077.

[14] N. Tancogne-Dejean et al. “Octopus, a Computational Framework for Exploring Light-Driven Phenomena and Quantum Dynamics in Extended and Finite Systems”. *The Journal of Chemical Physics* 152.12 2020, p. 124119. https://doi.org/10.1063/1.5142502.

[15] P. Eastman et al. “OpenMM 7: Rapid Development of High Performance Algorithms for Molecular Dynamics”. *PLOS Computational Biology* 13.7 2017, e1005659. https://doi.org/10.1371/journal.pcbi.1005659.

[16] C. Bannwarth et al. “Extended Tight-Binding Quantum Chemistry Methods”. *WIREs Computational Molecular Science* 11.2 2021, e1493. https://doi.org/10.1002/wcms.1493.

[17] *Linux Kernel History Report*. The Linux Foundation, 2020. https://www.linuxfoundation.org/wp-content/uploads/2020_kernel_history_report_082720.pdf (visited on 08/25/2021).

[18] A. Danial. *Count Lines of Code*. 2021. https://github.com/AlDanial/cloc (visited on 06/27/2021).

[19] A. I. Krylov et al. “What Is the Price of Open-Source Software?” *The Journal of Physical Chemistry Letters* 6.14 2015, pp. 2751–2754. https://doi.org/10.1021/acs.jpclett.5b01258.

[20] V. Stodden. “The Legal Framework for Reproducible Scientific Research: Licensing and Copyright”. *Computing in Science Engineering* 11.1 2009, pp. 35–40. https://doi.org/10.1109/MCSE.2009.19.

[21] J. D. Gezelter. “Open Source and Open Data Should Be Standard Practices”. *The Journal of Physical Chemistry Letters* 6.7 2015, pp. 1168–1169. https://doi.org/10.1021/acs.jpclett.5b00285.

[22] J. Swarts. “Open-Source Software in the Sciences: The Challenge of User Support”. *Journal of Business and Technical Communication* 33.1 2019, pp. 60–90. https://doi.org/10.1177/1050651918780202.

[23] I. Fdez. Galván et al. “OpenMolcas: From Source Code to Insight”. *Journal of Chemical Theory and Computation* 15.11 2019, pp. 5925–5964. https://doi.org/10.1021/acs.jctc.9b00532.

[24] *List of Quantum Chemistry and Solid-State Physics Software*. *Wikipedia*. 2021. https://en.wikipedia.org/w/index.php?title=List_of_quantum_chemistry_and_solid-state_physics_software (visited on 06/24/2021).

[25] *Comparison of Software for Molecular Mechanics Modeling*. *Wikipedia*. 2021. https://en.wikipedia.org/w/index.php?title=Comparison_of_software_for_molecular_mechanics_modeling (visited on 06/24/2021).

[26] *Materials Modelling and Computer Simulation Codes*. SklogWiki. http://www.sklogwiki.org/SklogWiki/index.php/Materials_modelling_and_computer_simulation_codes (visited on 06/27/2021).

[27] S. Pirhadi, J. Sunseri, and D. R. Koes. “Open Source Molecular Modeling”. *Journal of Molecular Graphics and Modelling* 69 2016, pp. 127–143. https://doi.org/10.1016/j.jmgm.2016.07.008.

[28] MolSSI. *MolSSI Software Database*. 2021. https://molssi.org/software-search/ (visited on 08/24/2021).

[29] L. M. Ghiringhelli et al. “Towards Efficient Data Exchange and Sharing for Big-Data Driven Materials Science: Metadata and Data Formats”. *npj Computational Materials* 3.1 2017, p. 46. https://doi.org/10.1038/s41524-017-0048-5.

[30] *Desmond Molecular Dynamics System*. Version 2021-2. New York: Schrödinger, 2021.

[31] S. van de Sandt et al. *Practice Meets Principle: Tracking Software and Data Citations to Zenodo DOIs*. Version 1. 2019. http://arxiv.org/abs/1911.00295 (visited on 06/28/2021).

[32] S. van de Sandt and A. Ioannidis. *A Tale of Software Citation Pitfalls*. 2020. DOI: 10.5281/zenodo.4263762.

[33] M. J. Frisch et al. *Gaussian 16 Revision C.01*. 2016.

[34] G. Kresse and J. Furthmüller. “Efficient Iterative Schemes for Ab Initio Total-Energy Calculations Using a Plane-Wave Basis Set”. *Physical Review B* 54.16 1996, pp. 11169–11186. https://doi.org/10.1103/PhysRevB.54.11169.

[35] S. J. Clark et al. “First Principles Methods Using CASTEP”. *Zeitschrift für Kristallographie - Crystalline Materials* 220.5-6 2005, pp. 567–570. https://doi.org/10.1524/zkri.220.5.567.65075.

[36] K. Aidas et al. “The Dalton Quantum Chemistry Program System”. *WIREs Computational Molecular Science* 4.3 2014, pp. 269–284. https://doi.org/10.1002/wcms.1172.

[37] M. Meunier and S. Robertson. “Materials Studio 20th Anniversary”. *Molecular Simulation* 47.7 2021, pp. 537–539. https://doi.org/10.1080/08927022.2021.1892093.

[38] F. Neese et al. “The ORCA Quantum Chemistry Program Package”. *The Journal of Chemical Physics* 152.22 2020, p. 224108. https://doi.org/10.1063/5.0004608.

[39] D. Dubbeldam et al. “RASPA: Molecular Simulation Software for Adsorption and Diffusion in Flexible Nanoporous Materials”. *Molecular Simulation* 42.2 2016, pp. 81–101. https://doi.org/10.1080/08927022.2015.1010082.

[40] Q. Sun et al. “Recent Developments in the PySCF Program Package”. *The Journal of Chemical Physics* 153.2 2020, p. 024109. https://doi.org/10.1063/5.0006074.

[41] WhiteSource. *Open Source Licenses in 2021: Trends and Predictions*. https://www.whitesourcesoftware.com/resources/blog/open-source-licenses-trends-and-predictions/ (visited on 06/29/2021).

[42] K. White. *Publications Output: U.S. Trends and International Comparisons*. 2019. https://ncses.nsf.gov/pubs/nsb20206/ (visited on 04/15/2021).

[43] S. Mammola et al. “Impact of the Reference List Features on the Number of Citations”. *Scientometrics* 126.1 2021, pp. 785–799. https://doi.org/10.1007/s11192-020-03759-0.

[44] G. Satell. “The Trillion-Dollar Potential in Crispr, Materials Science, and Quantum Computing”. *Barron’s* 2019. https://www.barrons.com/articles/the-trillion-dollar-potential-in-crispr-materials-science-and-quantum-computing-51550412044 (visited on 02/24/2020).

[45] S. Lehtola et al. “Recent Developments in Libxc — A Comprehensive Library of Functionals for Density Functional Theory”. *SoftwareX* 7 2018, pp. 1–5. https://doi.org/10.1016/j.softx.2017.11.002.

[46] E. F. Valeyev. *Libint*. 2021. https://github.com/evaleev/libint (visited on 08/24/2021).

[47] V. W.-z. Yu et al. “ELSI — An Open Infrastructure for Electronic Structure Solvers”. *Computer Physics Communications* 256 2020, p. 107459. https://doi.org/10.1016/j.cpc.2020.107459.

[48] A. Kozhevnikov et al. *SIRIUS*. Version 7.2.4. 2021. https://github.com/electronic-structure/SIRIUS (visited on 06/28/2021).

[49] R. J. Harrison et al. “MADNESS: A Multiresolution, Adaptive Numerical Environment for Scientific Simulation”. *SIAM Journal on Scientific Computing* 38.5 2016, S123–S142. https://doi.org/10.1137/15M1026171.

[50] J. A. Calvin and E. F. Valeev. *TiledArray*. Valeev Group, 2021. https://github.com/ValeevGroup/tiledarray (visited on 08/24/2021).

[51] J. Howison and J. Bullard. “Software in the Scientific Literature: Problems with Seeing, Finding, and Using Software Mentioned in the Biology Literature”. *Journal of the Association for Information Science and Technology* 67.9 2016, pp. 2137–2155. https://doi.org/10.1002/asi.23538.

[52] A. M. Smith, D. S. Katz, and K. E. Niemeyer. “Software Citation Principles”. *PeerJ Computer Science* 2 2016, e86. https://doi.org/10.7717/peerj-cs.86.

[53] GitHub. *Making Your Code Citable*. https://guides.github.com/activities/citable-code/ (visited on 06/28/2021).

[54] Q. Sun et al. *Pyscf/Pyscf: PySCF v1.7.1 Release*. Zenodo, 2020. https://doi.org/10.5281/zenodo.3692842.

[55] Q. Sun et al. *Pyscf/Pyscf: PySCF*. Zenodo, 2020. https://doi.org/10.5281/zenodo.593807.

[56] P. Eastman et al. *Openmm/Openmm: OpenMM*. Zenodo, 2020. https://doi.org/10.5281/zenodo.591297.

[57] S. Grimme, C. Bannwarth, and S. Ehlert. *Grimme-Lab/Xtb: Xtb*. Zenodo, 2021. https://doi.org/10.5281/zenodo.3712015.