Topics in Current Chemistry Collections

Michael Filatov
Cheol H. Choi
Massimo Olivucci *Editors*

New Horizons in Computational Chemistry Software

Springer

Topics in Current Chemistry Collections

Editor-in-Chief

Massimo Olivucci, Siena, Italy and Bowling Green, USA
Wai-Yeung Wong, Hong Kong, China

Series Editors

Hagan Bayley, Oxford, UK
Greg Hughes, Codexis Inc, USA
Christopher A. Hunter, Cambridge, UK
Seong-Ju Hwang, Seoul, Korea (Republic of)
Kazuaki Ishihara, Nagoya, Japan
Barbara Kirchner, Bonn, Germany
Michael J. Krische, Austin, USA
Delmar Larsen, Davis, USA
Jean-Marie Lehn, Strasbourg, France
Rafael Luque, Córdoba, Spain
Jay S. Siegel, Tianjin, China
Joachim Thiem, Hamburg, Germany
Margherita Venturi, Bologna, Italy
Chi-Huey Wong, Taipei, Taiwan
Henry N.C. Wong, Hong Kong, China
Vivian Wing-Wah Yam, Hong Kong, China
Chunhua Yan, Beijing, China
Shu-Li You, Shanghai, China

The series *Topics in Current Chemistry Collections* presents critical reviews from the journal *Topics in Current Chemistry* organized in topical volumes. The scope of coverage is all areas of chemical science including the interfaces with related disciplines such as biology, medicine and materials science. The goal of each thematic volume is to give the non-specialist reader, whether in academia or industry, a comprehensive insight into an area where new research is emerging which is of interest to a larger scientific audience. Each review within the volume critically surveys one aspect of that topic and places it within the context of the volume as a whole. The most significant developments of the last 5 to 10 years are presented using selected examples to illustrate the principles discussed. The coverage is not intended to be an exhaustive summary of the field or include large quantities of data, but should rather be conceptual, concentrating on the methodological thinking that will allow the non-specialist reader to understand the information presented. Contributions also offer an outlook on potential future developments in the field.

More information about this series at https://link.springer.com/bookseries/14181

Michael Filatov • Cheol H. Choi • Massimo Olivucci
Editors

New Horizons in Computational Chemistry Software

With contributions from

Mario Barbatti • Leonardo Barneschi • Filip Cernatic
Vsevolod D. Dergachev • Ilya D. Dergachev • Pavlo O. Dral
Emmanuel Fromager • Fuchun Ge • Yang Guo • Jong-Kwon Ha
Yi-Fan Hou • Jianxing Huang • Max Pinheiro Jr • Tae In Kim
Rika Kobayashi • Yibo Lei • Wenjian Liu • Aleksandr O. Lykhin
Robert C. Mauban • Seung Kyu Min • Massimo Olivucci
Yuriko Ono • Daniele Padula • Laura Pedraza-González
Vincent Robert • Mitra Rooein • Bruno Senjean • Yangyang Song
Tetsuya Taketsugu • Takuro Tsutsumi • Sergey A. Varganov
Luca De Vico • Bao-Xin Xue • Ning Zhang

Springer

Editors
Michael Filatov
Department of Chemistry
Kyungpook National University
Daegu, Korea (Republic of)

Cheol H. Choi
Department of Chemistry
Kyungpook National University
Daegu, Korea (Republic of)

Massimo Olivucci
Department of Biotech, Chemistry
and Pharmacy
University of Siena
Siena, Italy

Department of Chemistry
Bowling Green State University
Bowling Green
OH, USA

Partly previously published in Topics in Current Chemistry Volume 379 (2021); Topics in Current Chemistry Volume 380 (2022).

ISSN 2367-4067
Topics in Current Chemistry Collections
ISBN 978-3-031-07657-2

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
Chapters "MLatom 2: An Integrative Platform for Atomistic Machine Learning" and "Evolution of the Automatic Rhodopsin Modeling (ARM) Protocol" are licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/). For further details see licence information in the chapters.
This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Contents

Preface

In recent decades, scientific software has become an invaluable asset for the general science and technology community, as well as for the advanced engineering industry. With that it is becoming more and more obvious that the old-fashioned approach to the development and maintenance of the computational programs, which is traditionally carried out by a small group of core developers, becomes an obstacle on the way to design and implement new computational methodologies and techniques. For example, integrating new computational methodologies into the legacy codes and adapting them to the rapidly growing hardware capabilities and the new programming environments imposes a heavy burden on the core developers and slows down the progress of the scientific computing. Hence, new approaches to collaborative software design and development are desperately needed by the scientific community. An emerging concept of software modularity can potentially offer a way around the hurdle and allow multiple contributors to rapidly add missing functionalities and incorporate new computational methods. At the same time, the end users can benefit from a wide selection of available computational techniques and employ them to create customized workflows from the modules best suited for the problem at hand. It is the purpose of this collection to highlight some emerging methods and concepts in the domain of Computational and Theoretical Chemistry with an emphasis on collaborative environments, automation and machine learning also enabling fast and efficient implementation. A survey of novel computational schemes, theoretical concepts, and software technologies as well as their application to the computation and analysis of electronic structure, multiscale modeling, potential energy surface mapping, adiabatic and non-adiabatic molecular dynamics is given by leading experts in the field.

This topical collection opens with a comprehensive survey by Kobayashi of modern collaborative tools and technologies, which show a great potential in resolving the burden of computational software sustainability and development. Many existing computational codes have been started decades ago by relatively small groups of core developers, who faced the problem of continuity in maintaining these packages, keeping up with evolving hardware, operating systems and providing proper support to the end users. This underlines the necessity of strong collaborative frameworks as a means to guarantee sustainability of the software and its development. Kobayashi

gives a wide and deep overview of the existing and emerging collaborative tools encompassing the more traditional technologies, such as subversion and git, as well as the novel extended reality and virtual reality technologies, which have a potential to add the elements of "real" human interactions to remote collaboration and education.

One of the emerging technologies that has taken computational chemistry by storm is machine learning – a technology based on the use of the neural networks for organizing and interpolating massive arrays of heterogeneous data. Dral et al. describes a novel platform for atomistic machine learning simulations, the MLatom 2 suite, that, thanks to its modular design, enables an easy extension of its capabilities and simulation models. A novel approach to analyzing chemical reaction dynamics based on the reaction path network is presented by Taketsugu and coworkers. The Reaction Space Projector (ReSPer) technology enables reduction of the complexity of reaction paths on multiple potential energy surfaces by considerably reducing its dimensionality and projecting the dynamics trajectories onto the easily comprehensible low-dimensional reaction space. Along a similar venue, Varganov and coworkers propose a nonadiabatic statistical theory package NAST designed for analyzing and predicting kinetics of spin-dependent chemical processes. At a modest computational cost, NAST is capable of predicting the probabilities and rate constants of transitions between the electronic states of different spin multiplicities with the account for statistical and quantum effects.

One of the main hurdles in computational modeling of the reaction dynamics, especially the dynamics of the non-adiabatic reactions occurring in the excited electronic states, is setting up the appropriate and reproducible models with the inclusion of the effect of environment. Perdaza-González et al. describe the current status and future development perspectives of the Automatic Rhodopsin Modeling (ARM) protocol – a web based platform for designing multiscale computational models for the simulation of light induced processes occurring in a diverse family of light sensitive proteins. A special emphasis is put on standardization of the models and reproducibility of the computational results. A novel concept in the nonadiabatic dynamics simulations is presented by Min and coworkers, which is based on the exact factorization of the electron-nuclear wavefunction approach. The PyUNIxMD package implements a series of simulation methodologies derived from the exact factorization and enables transparent and efficient setup of the trajectory simulations. When simulating the nonadiabatic dynamics of photoexcited molecules, the computational chemistry methodologies are tasked with providing on-the-fly information on the energies, and gradients of the electronic states characterized by the strong electron correlation and multiconfigurational electronic structure. Song et al. present an overview of a series of novel approaches to the computation of such electronic states within the framework of multiconfigurational many-body perturbation theory. An emerging concept of ensemble density functional theory and its application to obtaining the excited states in a time-independent fashion is described by Fromager and coworkers. The emerging methodology holds a great promise to produce the accurate information on various excited states at a modest computational cost; what is precisely needed in the realm of modeling the excited state reaction dynamics.

We believe that this collection will prove useful and interesting for readers as an overview of the emerging computational tools and concepts in the realm of computational and theoretical chemistry. We would like to thank all the contributors to this collection. A special gratitude goes to the staff at Topics in Current Chemistry and the editorial board for offering us the opportunity to put together this edition.

Topics in Current Chemistry (2021) 379:41
https://doi.org/10.1007/s41061-021-00354-6

REVIEW

Technological Advances in Remote Collaborations

Rika Kobayashi[1]

Received: 16 August 2021 / Accepted: 23 September 2021 / Published online: 15 October 2021
© The Author(s), under exclusive licence to Springer Nature Switzerland AG 2021

Abstract
Sustainable scientific software needs a strong collaboration framework to ensure continuity by passing on the tools, skills and knowledge needed to the next generation. The COVID-19 pandemic triggered the unexpected effect of accelerating the development of remote platforms and tools to open up collaborations to a wider global community. In this article we outline the elements needed for such a framework, such as education, tools and community building, and discuss the current advances in technology with a nod to the future.

Keywords Remote collaboration · Computational chemistry · Software development · The Future of Meetings · XR

1 Introduction

Computational chemistry has been going strong since the 1950s when theoretical chemists first embraced the advent of digital computers to carry out ab initio molecular orbital calculations [1]. The earliest computational chemistry programs, such as POLYATOM [2], ATMOL [3] and Gaussian [4, 5], of which only Gaussian is still in use today, began to appear in the 1970s. Since then, the list of programs has proliferated, including a wider variety of methods, such as molecular mechanics, molecular dynamics and plane-wave codes. Some software packages are still being developed long after the people who started them have gone. Some have fallen by the wayside. This widely recognised problem of continuity in maintaining these packages—keeping up with new operating systems/architectures and support—has been coined software sustainability [6–8]. Software sustainability requires the inextricably linked resources of funding and people. Certainly, in computational chemistry,

Chapter 1 was originally published as Kobayashi, R. Topics in Current Chemistry (2021) 379: 41. https://doi.org/10.1007/s41061-021-00354-6.

✉ Rika Kobayashi
Rika.Kobayashi@anu.edu.au

[1] ANU Supercomputer Facility, Leonard Huxley Building 56, Mills Rd, Acton, ACT 2601, Australia

programming skills do not lead to a secure career path, especially in academia. As Katz [7] recognises, Software Developers need to "keep themselves supported/employed". In recognition of this, in 2016 the United States government funded the Molecular Sciences Software Institute (MolSSI) [9] to serve "as a nexus for science, education and cooperation serving the worldwide community of computational molecular scientists". This is the only such initiative in existence to our knowledge. A similar initiative led by Peter Gill was proposed to the Australian Government in 2014 but failed to gain funding. However, recognition and funding are only the beginning. Sustainable scientific software needs a strong collaboration framework.

The 2020 COVID-19 pandemic had an unintended effect of bringing the world closer together. The halt to international travel disrupted scientific exchange and researchers scrambled to find ways of making working virtually more effective and software platforms reinvented themselves. One such endeavour was the Future of Meetings (TFOM) symposium, which was organised by a like-minded community with the aim of investigating best practice in how to work, educate, collaborate and interact virtually. It became a truly cross-disciplinary symposium, encouraging us to explore key themes and share ideas from a variety of people and perspectives we don't normally interact with. The symposium proved highly successful and our findings were made into a report [10] and an invited comment from Nature [11]. One theme of the symposium was "Technology to help us collaborate virtually" and this brought to the fore a range of online tools and initiatives that could provide the infrastructure for scientific software sustainability.

2 Education

Probably the foremost issue for the continued development of scientific software is the shortage of programmers. This is recognised worldwide, as reflected in the many articles decrying the shortage of software engineers, e.g. Lee [12, 13] estimates that by the year 2024 that number is expected to reach 1 million. This problem is perceived as being one of education, spawning many government initiatives. In Australia, the government introduced the Coding Across the Curriculum initiative with the aim to promote the teaching of digital technologies, including coding, across the different year levels in Australian schools [14]. In 2016, President Obama proposed a CS For All initiative with a US$4 billion dollar budget for computer science education in the United States that did not get approved [15]. The European Commission produced a digital education report for Europe in 2019 [16] and a general worldwide overview can be found in a 2019 UNESCO report [17]. Most of these initiatives concern the Tech and IT sector and target languages, such as JavaScript, Java and Python [18]. Few address the shortage of scientific programmers, except in the rapidly growing field of data science. However, when it comes to software in the applied sciences, certainly in high-performance computing (HPC), the majority of programs are written in Fortran and C/C++ [19]. From a survey in 2015, Rouson (Rouson, personal communication) reported on the programming languages used at NERSC (National Energy Research Scientific Computing); Fortran accounted for close to 60%, followed by C++ and C, at about 35% and 31%, respectively. Fortran

was found to be the primary language for 23 of the 36 top codes, yet it is a language that is no longer widely taught. See also the entertaining talk by Roland Lindh in this series [20].

The technical skills shortage has been partly addressed by the rise in online learning [21]—a natural extension of distance education—suited particularly to programming and computer-related courses because of its basis in a digital environment. Worldwide government cuts in funding for education have made distance education more appealing as an easy source of revenue, leading to a highly competitive, and growing, education industry. At the turn of the century, the market for higher education through distance learning was estimated at US$ 300 billion. In 2019, pre-COVID times, Renub Research estimated that the online education market would reach US$ 350 billion by 2025 [22]. Renub's report highlighted online course providers Coursera [23] and Udacity [24] and indeed their most popular offerings are computer science related: programming, machine learning/artificial intelligence (ML/AI), data science. The majority of courses, as mentioned above, do not target scientific programmers. To date, the only online Fortran programming courses that could be found online were provided through Udemy [25] and Tutorialspoint [26], though there is further material available online in the form of university lecture notes and tutorial handouts. In fact, many online courses just provide lecture material, often in the form of videos and handouts. The more sophisticated online programming courses leverage the digital environment by embedding interactive exercises, as with the Tutorialspoint Fortran course, assessment and a discussion forum. At this point in time until this skills shortage is redressed, as has been done for the IT sector, passing on these advanced programming skills is in the hands of the scientific community.

3 Collaboration Tools

The global COVID-19 lockdowns found many technical platforms and tools coming into being or reinventing themselves to adapt to the remote working landscape as can be seen in Fig. 1 (a composite adapted and updated from several sources [27–29]). Video conferencing platforms became schools, fitness centers, places of worship; social apps, such as for chat and gaming, turned into tools for remote work.

The relevant tools from a programming perspective without a doubt would start with code hosting platforms or version control repositories. These have been in use for many years, starting with early version control systems [30], such as CVS [31] and Subversion [32], which were essentially a mechanism for tracking code revisions. However, as software development expanded and became more complex, often involving many authors working concurrently on different parts of code there was a need to have some form of coordination giving rise to distributed revision control, led by Git from Linus Torvalds in 2005 for development of the Linux kernel [33]. The majority of software packages are probably now hosted on such a code repository, the main ones in the scientific community being Github [34] and Gitlab [35]. This is often coupled with a Continuous Integration system. The idea behind the workflow is to keep a master copy of the code on the

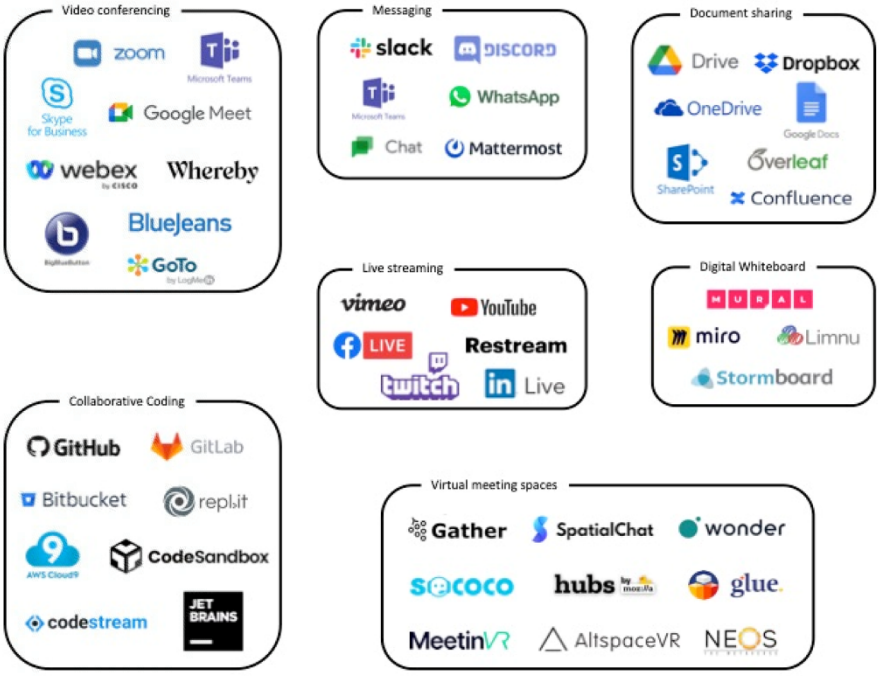

Fig. 1 Remote working tools for collaboration landscape as of 2021

repository which can be downloaded individually and worked on. Changes from the various developers' working copies are then merged back, with the platform providing a mechanism for tracking and resolving dependencies and conflicts. This is the most problematic part of collaborative software development, where independent developers can introduce changes incompatible with each other, e.g. reusing the same variable or changing the underlying structure. One way to mitigate this is to check out the code regularly to keep as close to the master copy as possible, the ideal being to spend less time merging the change than making the change itself—"integration hell". Continuous integration automates this merge to a frequent basis, at least daily, together with running a set of unit tests.

For the extended complicated software suites that make up the bulk of computational chemistry software, new developments can be a long time in the making, rendering "integration hell" unavoidable. Recent times, perhaps in mitigation of this or in conjunction with the popularity of "pair programming", have seen the springing up of real-time collaborative coding platforms where developers work on the same piece of code. This is already something that has been in place for a while with document sharing such as Google Docs [36] and Overleaf [37]. There will probably still be a need for a set of privileged developers to approve code changes but it is believed that the main advantage will be being able to see concurrent development as it happens, making potential conflicts more noticeable. Again, of the most popular collaborative coding tools [38], few target Fortran and

it is still early days so not clear whether this approach can scale to a complicated software suite with a number of modules and developers.

4 Community

The biggest concern during the COVID-19 global shutdown was arguably what could be termed the human factor, described variously as lack of connectedness, the need for "real" interactions, and is the main argument for "return to the office" (see next section). There was already a movement towards the formation of global research communities through the rise of collaborative hubs, also known as virtual research environments or science gateways [39]. The idea behind these is to provide a technological infrastructure of shared computational resources: software, data, tools, workflows, HPC access, through a web portal or apps, thus enabling research to a broader scientific community. There now exist many well-established communities, as can be seen in the Special Issue on International Science Gateways 2017 [39], and these have proved successful in some disciplines, notably the Galaxy Project [40], started in 2005, which, according to their latest report [41], had "served hundreds of thousands of users, been used in >5700 scientific publications, and provided 500+ developers with a framework provisioning accessible, transparent and reproducible data analysis".

There is already some online community through the various code repository platforms but a true collaborative hub should have elements of:

- venue—a central website accessible to all and possibly distributed regional hubs to serve a more localized community;
- repositories for collecting and sharing software, tools, data, workflows;
- active community that can communicate synchronously and asynchronously for the exchange of ideas and training;
- (optionally) access to high-end HPC facilities as part of the workflow.

In the computational chemistry world there are existing initiatives, such as NanoHub [42], and nascent hubs, such as AiiDA [43] and Edison [44]. Nanohub, having been established in 2002, is the oldest of these and describes itself as a science gateway supporting a global Network for Computational Nanotechnology community within a cloud environment. It provides a wealth of resources, including training courses, discussion forums, simulation and modeling tools, and a computing environment in which to run them [45]. Its main usage and success appears, however, to be in delivering courses and enabling simulations—supporting an application community rather than a programming and development one.

On the other hand, AiiDA started from the Quantum Espresso [46] developers community, primarily for building an infrastructure providing tools for designing, deploying and analysing materials science simulations, integrated into HPC environments. However, simultaneous efforts since into education and collaboration have expanded its range into Materials Cloud—a web platform for computational materials scientists to "share their work and promote open science" [47].

Furthermore, the Materials Cloud community have begun taking steps towards defining standards, such as file formats and metadata, to facilitate interoperability. The concept of interoperability, the ability for different groups to exchange data consistently, is recognized as an important part of software sustainability, but as yet has been addressed comparatively little in computational chemistry.

A final quick mention should be given to the talk that initiated and possibly inspired this symposium series "A Web Platform for Scientific Collaborations", lectured by Cheol Ho Choi [48]. Leveraging the principles of modular environments, they have created a web platform based on sharing and running computational chemistry modules and workflows via a graphical pipeline and opened it up to the wider quantum chemistry community in the hopes of establishing a scientific software ecosystem.

However, these collaboration hubs are still rooted in two-dimensional screens and do not authentically fill the gap of the lamented missing "real" human interactions—the body language, corridor conversations and serendipitous interactions. Continued TFOM activities have allowed us to explore further these social aspects, especially whether extended reality (XR) and immersive technology can help add a social human factor into our virtual offices and conferences. To this end, we have held a variety of events in virtual reality (VR) platforms such as Altspace [49], NEOS [50] and Glue [51]. These were engaging and fully immersive, but the immaturity of the platforms and the technological requirements do not make them practical today. We were able to explore this aspect further by being given the opportunity to discuss "The Future of Meetings: Working in XR?" with the XR Developer Community through a Birds of a Feather Session at SIGGRAPH 2021 [52]. As part of the session, the attendees were polled informally on a variety of questions concerning the state of XR (for the complete set see Ref. [53]). The poll was not a rigorously conducted exercise so not too much can be read into it, but it was indicative that the albeit small sample felt XR was able to substitute "real" human interactions and that the industry believed that we will be seeing meaningful change in the near future (Fig. 2).

Fig. 2 Informal poll results on various aspects of working in XR from our Birds of a Feather Session at SIGGRAPH 2021

5 Virtual Conferences

In the early days of the COVID-19 pandemic an article appeared in Nature headed "A year without conferences?" [54] discussing the impact on researchers and raising the prospect of a need to rethink the concept of meetings. This article was very quickly countered by examples of successful virtual conferences from around the world, spanning many disciplines, notably one from the Virtual Winter School on Computational Chemistry, which has been running annually since 2015 [55]. Traditionally, conferences have been a means for scientific communities to meet and share knowledge, but these stemmed from a time when communication was slow and travel not so easy. However, technological advances as described here have blurred the need to travel to achieve these outcomes. The pros and cons of virtual conferences were explored in depth in the aforementioned Future of Meetings symposium [10, 11] and continue to be explored through various initiatives by the TFOM community [56]. The definite "pro" of virtual conferences has, without doubt, been their accessibility, inclusivity and sustainability. On the whole, virtual attendance figures have been much higher than their in-person equivalents as the cost of travel is no longer a barrier to attendance. Junior researchers have reporting increased confidence and feeling of safety in the virtual environment and there is significantly less harm to the environment. TFOM calculated the symposium produced 1420 kg of CO_2 equivalent compared to an in-person equivalent of 280,000 kg of CO_2. The biggest disadvantage, certainly for Australia, has been juggling timezones, coupled with the difficulty in separating conference and domestic duties as few international conferences overlap with normal working hours. For developing countries technological accessibility is the biggest problem and a recent a poll of 900 Nature readers [57] cited "poor networking opportunities" as the biggest drawback.

6 Future Outlook

With the effects of the COVID-19 pandemic still being felt around the world today, it is probable that there will be long-lasting changes to the way we work and meet. Major companies, especially the tech giants, are reducing their office space with some going fully remote [58] following the lead in May 2020 of Twitter and Facebook who announced that they would give staff the option to work remotely permanently [59]. Together with the rise in the globally competitive Distance Education industry, online learning is becoming more accepted. The COVID-19 pandemic provided momentum for overcoming the potential barrier to adoption of remote teaching practices that had been considered niche, such as flipped teaching. Such practices have become mainstream, especially as more educators are recognising their effectiveness in this digital age. The perception that online degrees were not "real" degrees is diminishing now that many students have been able to experience direct comparisons. Similarly, Virtual Winter

School, which had been motivated by the desire to make accessible to a wider audience experts in the field they would not normally be able to hear live or interact with, was attended regularly by participants from less advantaged countries. The 2021 School had noticeably more attendance, typically reaching about 200 for most sessions, more than double the usual attendance, and especially from the more established community. The level of engagement demonstrated that the Virtual School is a viable format for fruitful scientific exchange and hopefully this participation level will continue.

There is evidence that scientists want virtual meetings to stay after the COVID pandemic from the Nature poll [57] and reflected in our own survey from TFOM shown in Fig. 3. The concept of "hybrid" is gaining in popularity and it could be that the future will be some mixture of in-person and virtual. However, to be done well, virtual collaborations, whether conferences, meetings or teaching, need more effort, from planning to delivery. TFOM activities have shown how virtual conferences can be effective and subsequently we have been continually called on to advise on best practice for a variety of virtual initiatives. The obvious benefits of accessibility, inclusivity and sustainability, as highlighted in our conference experiences, are still competing with the drawbacks of time-zones, technological accessibility and the human factor, though XR may soon be able to provide a solution for that. TFOM is seeing disheartening signs of people wanting to take what they think is the easy option, i.e. go back to the way things were before. The future will be determined by the people who can see what virtual can do versus the people who see what it can't.

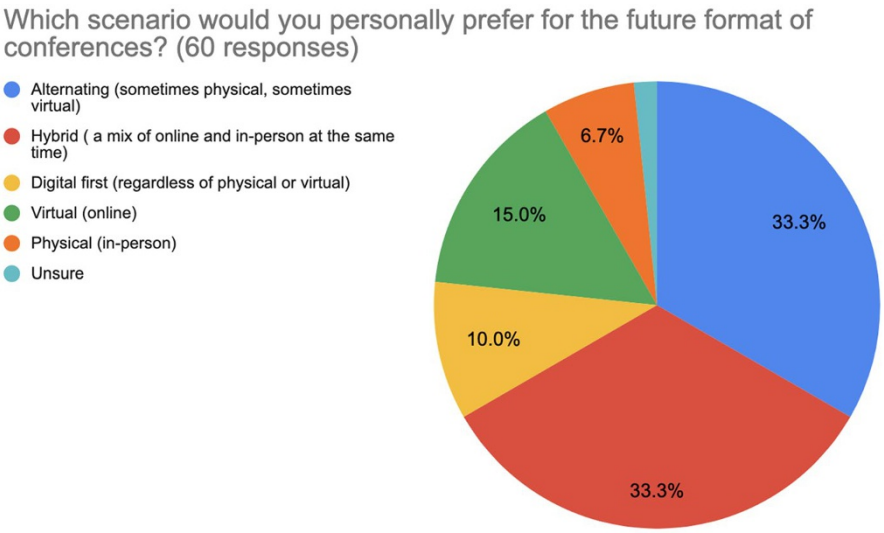

Fig. 3 Attendee preferences for the future format of conferences from The Future of Meetings symposium [10]

7 Conclusion

Sustainable scientific software needs a strong collaboration framework. With the lifetime of computational chemistry software packages exceeding that of the developers who began them, there needs to be a mechanism to pass down the generations the software, tools, skills and knowledge to maintain continuity. The COVID-19 pandemic opened up the world to a potentially global community of developers, whether through the imaginative creation of remote collaboration platforms and tools or just by forcing people to take the digital plunge. It is still early days to know what work in the post-pandemic world will look like, whether we just go back to how we used to do things or whether these remote innovations will be embraced and developed further. There is now a wealth of tools out there to help us meet and work, and possibly even develop software, better virtually. Through TFOM, we have demonstrated how these tools can be used to return to a better normal. Now is the time to keep the momentum going and make use of them to build the foundation of a solid and global software development community.

Acknowledgements Thanks to the TFOM community, especially Vanessa Moss, Glen Rees and Patrice Rey for accompanying me on this journey, and Aidan Hotan, Chenoa Tremblay, Claire Trenham, Roger Amos and "Earthmark" for cheering us on (and providing invaluable material). Thanks also to Damian Rouson of Berkeley Lab for his excellent "Why Fortran persists" presentation, and the Australian Government Department of Education, Skills and Employment for providing a copy of their Coding Across the Curriculum report.

Declarations

Conflict of interest The author has no conflicts of interest or relevant financial interests to declare.

References

1. Roothaan CCJ (1951) New developments in molecular orbital theory. Rev Mod Phys 23:69. https://doi.org/10.1103/RevModPhys.23.69
2. Moskowitz JW, Snyder LC (1977) Polyatom: a general computer program for ab initio calculations. In: Schaefer HF (ed) Methods of electronic structure theory. Modern theoretical chemistry, vol 3. Springer, Boston
3. ATMOL. http://www.chilton-computing.org.uk/acl/applications/qc/p003.htm. Accessed 23 July 2021
4. Gaussian.com. Expanding the limits of computational chemistry. https://gaussian.com/. Accessed 23 July 2021
5. Frisch MJ, Trucks GW, Schlegel HB, Scuseria GE, Robb MA, Cheeseman JR, Scalmani G, Barone V, Petersson GA, Nakatsuji H, Li X, Caricato M, Marenich AV, Bloino J, Janesko BG, Gomperts R, Mennucci B, Hratchian HP, Ortiz JV, Izmaylov AF, Sonnenberg JL, Williams-Young D, Ding F, Lipparini F, Egidi F, Goings J, Peng B, Petrone A, Henderson T, Ranasinghe D, Zakrzewski VG, Gao J, Rega N, Zheng G, Liang W, Hada M, Ehara M, Toyota K, Fukuda R, Hasegawa J, Ishida M, Nakajima T, Honda Y, Kitao O, Nakai H, Vreven T, Throssell K, Montgomery JA Jr, Peralta JE, Ogliaro F, Bearpark MJ, Heyd JJ, Brothers EN, Kudin KN, Staroverov VN, Keith TA, Kobayashi R, Normand J, Raghavachari K, Rendell AP, Burant JC, Iyengar SS, Tomasi J, Cossi M, Millam JM,

Klene M, Adamo C, Cammi R, Ochterski JW, Martin RL, Morokuma K, Farkas O, Foresman JB, Fox DJ (2016) Gaussian 16, Revision C.01. Gaussian Inc., Wallingford
6. Blanton B, Lenhardt C (2014) A scientist's perspective on sustainable scientific software. J Open Res Softw 2:17. https://doi.org/10.5334/jors.ba
7. Katz DS (2018) Fundamentals of software sustainability. https://danielskatzblog.wordpress.com/2018/09/26/fundamentals-of-software-sustainability/. Accessed 11 July 2021
8. Hannay J, MacLeod C, Singer J, Langtangen HP, Pfahl D, Wilson G (2009) How do scientists develop and use scientific software?" In: Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering, SECSE '09, Washington, DC, USA, IEEE Computer Society, pp 1–8
9. MolSSI–The Molecular Sciences Software Institute. https://molssi.org/. Accessed 23 July 2021
10. Moss VA, Hotan AW, Kobayashi R, Rees GA, Siegel C, Tremblay CD, Trenham CE, Engelke U, Gray A, Hurley-Walker N, Roos G (2020) The future of meetings: outcomes and recommendations. Zenodo. https://doi.org/10.5281/zenodo.4345562
11. Moss VA, Adcock M, Hotan AW, Kobayashi R, Rees GA, Siégel C, Tremblay CD, Trenham CE (2021) Forging a path to a better normal for conferences and collaboration. Nat Astron 5:213–216. https://doi.org/10.1038/s41550-021-01325-z
12. Lee T (2019) How to close the tech skills gap. Scientific American. https://blogs.scientificamerican.com/observations/how-to-close-the-tech-skills-gap/. Accessed 19 July 2021
13. Scarpelli B, Miller N, Stephens R (2017) State of the app economy, 5th edn. Act: The App Association
14. Australian Government Department of Education and Training (2018) Evaluation of the Coding Across the Curriculum program Australia: dandolopartners
15. Wills B (2016) The United States of coding. New America. https://www.newamerica.org/weekly/united-states-coding/. Accessed 19 July 2021
16. European Commission/EACEA/Eurydice (2019) Digital education at school in Europe. Eurydice Report. Publications Office of the European Union, Luxembourg
17. Storte D, Webb M, Bottino R, Passey D, Kalas I, Bescherer C, Smith J, Angeli C, Katz Y, Micheuz P, Røsvik S, Brinda T, Fluck A, Magenheim J, Anderson B, Fuschek G (2019) Coding, programming and the changing curriculum for computing in schools. Report of UNESCO/IFIP TC3 Meeting at OCCE—Wednesday 27th of June 2018, Linz, Austria
18. Hughes O (2021) These are the programming languages most in-demand with companies hiring. TechRepublic. https://www.techrepublic.com/article/these-are-the-programming-languages-most-in-demand-with-companies-hiring/. Accessed 19 July 2021
19. Partee S (2021) Why are climate models written in programming languages from 1950? PARTEE.IO. https://partee.io/2021/02/21/climate-model-response/. Accessed 15 July 2021
20. Lindh R (2021) Strategies for the OpenMolcas Legacy codes—fifty shades of Fortran [Video]. YouTube. https://youtu.be/LcDU_O4fmXg. Accessed 8 Mar 2021
21. Koksal I (2020) The rise of online learning. Forbes. https://www.forbes.com/sites/ilkerkoksal/2020/05/02/the-rise-of-online-learning/. Accessed 13 Nov 2020
22. Renub Research (2019) Online education market & global forecast, by end user, learning mode (self-paced, instructor led), technology, country, company. https://www.renub.com/online-education-market-p.php. Accessed 13 Nov 2020
23. Coursera. https://www.coursera.org/. Accessed 31 July 2021
24. Udacity. https://www.udacity.com/. Accessed 31 July 2021
25. Udemy Fortran Courses. https://www.udemy.com/topic/fortran/. Accessed 31 July 2021
26. Tutorialspoint —Learn Fortran. https://www.tutorialspoint.com/fortran/index.htm. Accessed 31 July 2021
27. Wengi F (2020) Remote work (WFH) tech landscape pauaventures. https://medium.com/paua-insights/remote-work-wfh-tech-landscape-85627ced3410. Accessed 23 Oct 2020
28. Out of office: 65+ startups helping you work from home (2020) CBINSIGHT. https://www.cbinsights.com/research/remote-work-from-home-market-map/. Accessed 23 Oct 2020
29. The ultimate remote work tools landscape. https://www.holloway.com/g/remote-work/sections/the-ultimate-remote-work-tools-landscape. Accessed 23 October 2020
30. Rochkind MJ (1975) The source code control system. IEEE Trans Softw Eng 1:364–370
31. Tichy WF (1985) Rcs — a system for version control. Softw Pract Exp 15:637–654
32. Collins-Sussman B, Fitzpatrick BW, Pilato CM (2004) Version control with subversion, 1st edn. O'Reilly Media, Sebastopol, CA

33. Chacon S, Straub B (2014) Pro git. Apress, Berkeley. https://doi.org/10.1007/978-1-4842-0076-6
34. GitHub. https://github.com/. Accessed 4 Aug 2021
35. GitLab. https://gitlab.com/. Accessed 4 Aug 2021
36. Google Docs. https://docs.google.com/. Accessed 4 Aug 2021
37. Overleaf. https://www.overleaf.com/. Accessed 4 Aug 2021
38. Jones B (2021) The top 5 online IDEs for in-browser development developer.com. https://www.developer.com/cloud/top-online-ide/. Accessed 31 July 2021
39. Dahan M, Pirzl R, Gesing S (2020) International science gateways 2017 special issue. Futur Gener Comput Syst 110:320–322. https://doi.org/10.1016/j.future.2020.04.045
40. Galaxy Community Hub. https://galaxyproject.org/. Accessed 4 Aug 2021
41. Afgan E, Baker D, Batut B, van den Beek M, Bouvier D, Čech M, Chilton J, Clements D, Coraor N, Grüning BA, Guerler A, Hillman-Jackson J, Hiltemann S, Jalili V, Rasche H, Soranzo N, Goecks J, Taylor J, Nekrutenko A, Blankenberg D (2018) The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. Nucleic Acids Res 46:537–544. https://doi.org/10.1093/nar/gky379
42. nanoHUB. https://nanohub.org/. Accessed 4 Aug 2021
43. Huber SP, Zoupanos S, Uhrin M, Talirz L, Kahle L, Häuselmann R, Gresch D, Müller T, Yakutovich AV, Andersen CW, Ramirez FF, Adorf CS, Gargiulo F, Kumbhar S, Passaro E, Johnston C, Merkys A, Cepellotti A, Mounet N, Marzari N, Kozinsky B, Pizzi G (2020) AiiDA 1.0, a scalable computational infrastructure for automated reproducible workflows and data provenance. Sci Data 7:300. https://doi.org/10.1038/s41597-020-00638-4
44. Edison MQCP-Modular Quantum Chemistry Package https://mqcp.edison.re.kr/. Accessed 15 Aug 2021
45. Madhavan K, Zentner L, Farnsworth V, Shivarajapura S, Zentner M, Denny N, Klimeck G (2013) nanoHUB.org: cloud-based services for nanoscale modeling, simulation, and education. Nanotechnol Rev 2:107–117. https://doi.org/10.1515/ntrev-2012-0043
46. Giannozzi P, Baroni S, Bonini N, Calandra M, Car R, Cavazzoni C, Ceresoli D, Chiarotti GL, Cococcioni M, Dabo I, Dal Corso A, de Gironcoli S, Fabris S, Fratesi G, Gebauer R, Gerstmann U, Gougoussis C, Kokalj A, Lazzeri M, Martin-Samos L, Marzari N, Mauri F, Mazzarello R, Paolini S, Pasquarello A, Paulatto L, Sbraccia C, Scandolo S, Sclauzero G, Seitsonen AP, Smogunov A, Umari P, Wentzcovitch RM (2009) QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials. J Phys Condens Matter 21:395502. https://doi.org/10.1088/0953-8984/21/39/395502
47. Talirz L, Kumbhar S, Passaro E, Yakutovich AV, Granata V, Gargiulo F, Borelli M, Uhrin M, Huber SP, Zoupanos S, Adorf CS, Andersen CW, Schütt O, Pignedoli CA, Passerone D, Van de Vondele J, Schulthess TC, Smit B, Pizzi G, Marzari N (2020) Materials cloud, a platform for open computational science. Sci Data 7:299. https://doi.org/10.1038/s41597-020-00637-5
48. Choi C (2021) A web platform for scientific collaborations [Video]. YouTube. https://youtu.be/2OnZ0DUigVA. Accessed 10 Mar 2021
49. AltspaceVR https://altvr.com/ Accessed 15 Aug 2021
50. NEOS https://neos.com/ Accessed 15 Aug 2021
51. Glue https://glue.work/ Accessed 15 Aug 2021
52. Rees GA, Kobayashi R and Moss VA (2021) The future of meetings: working in XR? [Conference session]. SIGGRAPH 2021, United States. https://siggraph2021.hubb.me/fe/schedule-builder/schedule/sessions/868127. Accessed 12 Aug 2021
53. SIGGRAPH (2021) Birds of a Feather. https://thefutureofmeetings.wordpress.com/siggraph-2021-birds-of-a-feather/. Accessed 15 August 2021
54. Viglione G (2020) A year without conferences? How the coronavirus pandemic could change research. Nature 579:327–328. https://doi.org/10.1038/d41586-020-00786-y
55. Roos G, Oláh J, Ingle R, Kobayashi R, Feldt M (2020) Online conferences—towards a new (virtual) reality. Comput Theor Chem 1189:112975. https://doi.org/10.1016/j.comptc.2020.112975
56. The Future of Meetings. https://thefutureofmeetings.wordpress.com/. Accessed 22 September 2021
57. Remmel A (2021) Scientists want virtual meetings to stay after the COVID pandemic. Nature 591:185–186. https://doi.org/10.1038/d41586-021-00513-1

58. Stahl A (16 April 2021) The future of offices and workspaces, post-pandemic. Forbes. https://www.forbes.com/sites/ashleystahl/2021/04/16/the-future-of-offices-and-workspaces-post-pandemic/. Accessed 29 July 2021
59. Brownlee D (2020) Twitter, Square announce work from home forever option: what are the risks. Forbes. https://www.forbes.com/sites/danabrownlee/2020/05/18/twitter-square-announce-work-from-home-forever-optionwhat-are-the-risks/. Accessed 31 July 2021

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Topics in Current Chemistry (2021) 379:27
https://doi.org/10.1007/s41061-021-00339-5

REVIEW

MLatom 2: An Integrative Platform for Atomistic Machine Learning

Pavlo O. Dral[1,2] · Fuchun Ge[2] · Bao-Xin Xue[1,2] · Yi-Fan Hou[1,2] · Max Pinheiro Jr[3] · Jianxing Huang[1,2] · Mario Barbatti[3]

Received: 22 February 2021 / Accepted: 7 May 2021 / Published online: 8 June 2021
© The Author(s) 2021

Abstract

Atomistic machine learning (AML) simulations are used in chemistry at an ever-increasing pace. A large number of AML models has been developed, but their implementations are scattered among different packages, each with its own conventions for input and output. Thus, here we give an overview of our MLatom 2 software package, which provides an integrative platform for a wide variety of AML simulations by implementing from scratch and interfacing existing software for a range of state-of-the-art models. These include kernel method-based model types such as KREG (native implementation), sGDML, and GAP-SOAP as well as neural-network-based model types such as ANI, DeepPot-SE, and PhysNet. The theoretical foundations behind these methods are overviewed too. The modular structure of MLatom allows for easy extension to more AML model types. MLatom 2 also has many other capabilities useful for AML simulations, such as the support of custom descriptors, farthest-point and structure-based sampling, hyperparameter optimization, model evaluation, and automatic learning curve generation. It can also be used for such multi-step tasks as Δ-learning, self-correction approaches, and absorption spectrum simulation within the machine-learning nuclear-ensemble approach. Several of these MLatom 2 capabilities are showcased in application examples.

Keywords Machine learning · Quantum chemistry · Kernel ridge regression · Neural networks · Gaussian process regression

Chapter 2 was originally published as Dral, P. O., Ge, F., Xue, B-X., Hou, Y-F., Pinheiro Jr, M., Huang, J. & Barbatti, M. Topics in Current Chemistry (2021) 379: 27. https://doi.org/10.1007/s41061-021-00339-5.

✉ Pavlo O. Dral
dral@xmu.edu.cn

[1] State Key Laboratory of Physical Chemistry of Solid Surfaces, Fujian Provincial Key Laboratory of Theoretical and Computational Chemistry, Xiamen 361005, China

[2] Department of Chemistry, and College of Chemistry and Chemical Engineering, Xiamen University, Xiamen 361005, China

[3] Aix Marseille University, CNRS, ICR, Marseille, France

1 Introduction

Machine learning (ML) has taken computational chemistry by storm [1–4]. It is often applied to find a relationship between given molecular geometry and quantum chemical (QC) properties. A particularly useful application of such atomistic ML (AML) models is mapping molecular potential energy surfaces (PESs) [5–8]. Creating AML models is, however, a complicated task and requires domain knowledge. Thus, much effort has been put into developing a mathematical foundation and writing specialized software for such simulations.

One of us (POD) started to develop the MLatom program package [9, 10] for atomistic simulations with ML already in 2013 when not many such packages were available. At first, it was written entirely in Fortran and parallelized with OpenMP as a self-contained black-box program for user-friendly calculations. Now, this part comprises the Fortran core of MLatom called MLatomF. Later, MLatomF added a Python wrapper called MLatomPy implementing multi-step tasks such as Δ-learning [11] and self-correction [12]. We have implemented these and other methods developed by ourselves, such as structure-based sampling [12], the KREG model [12], ML-nuclear ensemble approach (ML-NEA) for precise absorption spectrum simulations [13], as well as selected literature methods, such as those based on the Coulomb matrix descriptor [14, 15], in MLatom for development purposes, tighter integration, and higher efficiency (see "Native Implementations"). We have used these native implementations also for developing methods for improving QC Hamiltonian [16], accelerating ML nonadiabatic excited-state dynamics [17], and for PES construction with spectroscopic accuracy by introducing a hierarchical ML (hML) [18] approach.

In recent years, we have witnessed the rapid rise of many exciting new ML models [4, 5]. They are often designed for different applications ranging from very accurate ML PES trained on as few as a hundred molecular configurations of small- and medium-sized molecules [19, 20] to ML models trained on thousands or millions of points to be transferable to large molecules [21, 22]. Each has its own advantages and disadvantages. It is, therefore, highly desirable to be able to test different models before applying them to the problem at hand. This is, however, a formidable task because these models are scattered in many different software packages. Each has its own conventions for input and output.

We face the same problem in our research: when we want to test some promising ML model, there is often a high entry barrier for learning how to use the corresponding package. Sometimes the documentation is very poor, and only interaction with experienced users or developers enabled us to use some packages. Often, some critical functionality, such as hyperparameter optimization, is missing.

Thus, as a pragmatic solution, we have provided the community with an integrated platform based on MLatom that interfaces the selection of popular third-party software packages via MLatomPy written in Python 3.6+ [23, 24]. This platform is released as MLatom 2 with all Python interfaces available as open-source, free software for non-commercial use. Importantly, the same input and

output structure can now be used for many state-of-the-art, third-party ML models (see Interfaces). We have implemented the interfaces with sGDML [19, 25] (symmetrized gradient-domain ML), GAP and QUIP (providing GAP [26]-SOAP [27] method), TorchANI [28] (providing ANI [21] methods), DeePMD-kit [29] (providing the DPMD [30] and DeepPot-SE [31] methods), and PhysNet [22] programs. This selection of methods covers popular representatives of various types of methods, ranging from those based on kernel methods (KMs) to neural networks (NNs). Our implementation also supports hyperparameter optimization using Bayesian methods with Tree-structured Parzen Estimator (TPE) [32] via the hyperopt [33] package.

The modular structure of MLatom allows easy extension to other models in the future, as it requires only writing a separate independent module for converting MLatom input to the input of the third-party software and parsing the latter's output. A similar approach is also used for interfacing various QC software packages [34]. This differs, however, from an alternative approach where some packages offer only part of an ML model, e.g., only a descriptor of a molecule to be used as an input for ML, as in DScribe [35].

In the following, we provide an overview of MLatom 2 capabilities, and details of native implementations and interfaces. We also demonstrate the application of MLatom 2 to several typical AML simulation tasks (hyperparameter optimization and generation of learning curves), Δ-learning and structure-based sampling, and calculation of absorption spectra.

2 Overview

The philosophy behind MLatom is to provide the community with a black-box-like program that allows a variety of calculation tasks required for ML atomistic simulations to be performed (Fig. 1). The program provides only with user-friendly and intuitive input, and no scripting is required. Under the hood, MLatom is built of modules designed to be independent of each other as much as possible to the extent that many modules can be used as stand-alone programs entirely independent from the main program. As needed, the modules are combined to create a seamless workflow eliminating step-by-step manual calculations.

The calculation tasks in MLatom are ML tasks and data set tasks. ML tasks are calculations involving training and using ML models. Our implementation includes a wide range of such tasks from basic to multi-step procedures: using an existing ML model from a file, creating an ML model and saving it to a file for later use, estimating ML model accuracy (model evaluation to determine the generalization error), Δ-learning [11], self-correction [12], learning curve and nuclear-ensemble spectrum generation [13]. Data set tasks perform all the operations necessary for ML simulations, such as converting geometries given in XYZ format to the input vector $\mathbf{x}$ for ML, splitting the data set into the required subsets (e.g., training, validation, and test), sampling points into these subsets, and performing statistical analysis of ML estimated values (e.g., error concerning reference values). These tasks can be performed either independently from each other, e.g., creating an ML model from

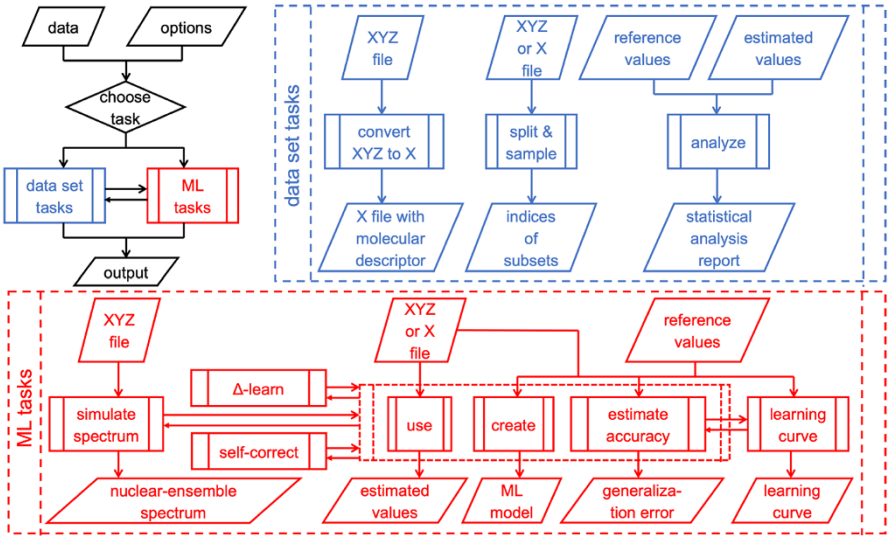

Fig. 1 Overview of tasks performed by MLatom

available input vectors **x** or combined, e.g., by first converting XYZ coordinates to **x** and then creating an ML model. The user just needs to modify several input lines to perform simulations using the first or second option. In the following, we describe each of these tasks and define the most important concepts in ML atomistic simulations.

2.1 ML Tasks

Currently, MLatom supports only supervised learning, which boils down to finding and using an approximating function $\hat{f}(\mathbf{x};\mathbf{h};\mathbf{p})$ that establishes a relationship between the reference values y and input vectors **x** in the training set based on statistically motivated assumptions [36] rather than on purely physical modelling. The approximating function typically has a large number of parameters **p** and so-called hyperparameters **h**.

2.1.1 Using ML Models

Using an existing ML model is conceptually simple as it requires information about the mathematical form of the approximating function and (hyper)parameters. It includes knowing how to transform a molecular geometry into an input vector **x**. One should pay attention, however, to many technical issues, such as ensuring consistent conventions for storing and retrieving this information from the file for long-term re-usability and reproducibility of scientific simulations. Another technical issue is related to performance and accuracy, as the information to be stored can be quite sizable, which can quickly lead to storage and input/output bottlenecks. MLatom saves ML model parameters and other information in a binary format file

with a fixed structure for the core ML models and uses native formats of interfaced third-party software without converting them.

In atomistic simulations, we are also often interested in derivatives of properties. For example, in molecular dynamics simulations, we need to know derivatives of energy with respect to atomic coordinates (energy gradients = − forces). Thus, MLatom can estimate both property values and partial derivatives with respect to elements of the input vector or atomic XYZ coordinates.

2.1.2 Creating ML Models

Creating an ML model is already a much more complicated task as one needs to find the model (hyper)parameters in the right way (Fig. 2). This means that one needs to search for optimal values in the parameter space, leading to as low a generalization error as possible [36]. This is not the same as fitting parameters (training) that would give as low an error in the training set as possible. Modern ML methods can easily and exactly reproduce (an extreme case of overfitting) the reference values in the training set [2]. Thus, the standard practice is to set aside a validation set to ensure that training on the remaining data points will not lead to a large error in the validation set, i.e., to avoid overfitting [36]. The remaining data points are called either training or sub-training set in the literature, which adds to the confusion. While both conventions are valid, we prefer to call them sub-training points both here and in MLatom. All data points that are used in creating the ML model we call the training set. This set includes the sub-training and the validation sets in our nomenclature. This convention allows for a fairer and more straightforward comparison of ML models trained on the same number of training points as the validation set, which is used indirectly in training, to be accounted

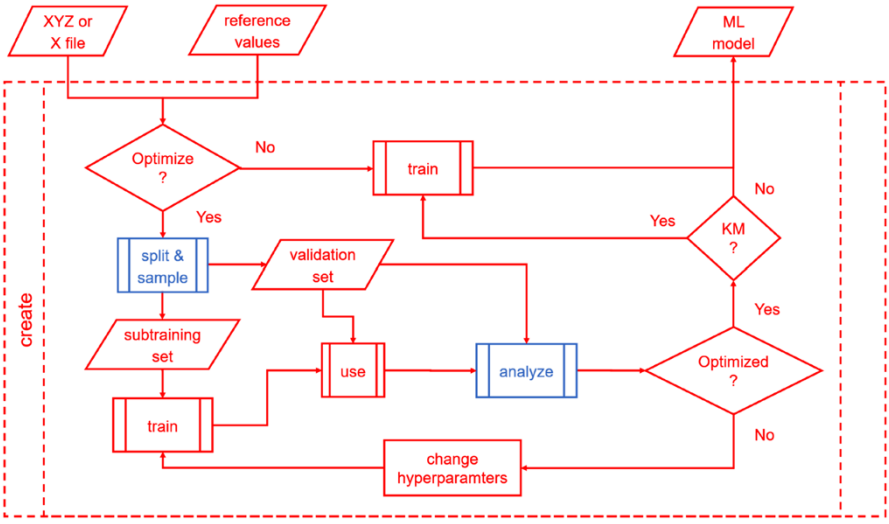

Fig. 2 Creating a machine learning (ML) model with MLatom can involve automatic model selection (hyperparameter tuning) using different types of the training set split into sub-training and validation sets and different sampling procedures

for. For example, if the model is trained on 1000 points, but used another 1000 points for validation, then the reference data is needed for all 2000 points, and such a model cannot be compared to another model trained on only 1000 points without using such a validation set.

When additional information such as derivatives of properties is available, it can be included into the training set too. It is common to train ML models for PESs simultaneously on energies and gradients (or only gradients), which is known to improve the quality of ML PESs significantly compared to fitting only on energies [7, 37, 38]. NNs simply fit parameters to the reference properties and their derivatives [38], while KMs can include the derivative information into their model explicitly [7, 37].

Many knobs exist and can be tuned in the process of finding suitable parameters for an ML model. One such knob concerns the ML model itself, and another deals with the splitting into sub-training and validation sets. As with the first type, while we do not need to touch the model parameters as this is the machine's task, we can influence the model by changing its hyperparameters manually [36, 39]. As a side note, the difference between parameters and hyperparameters is somewhat fussy as the latter can be found by a machine too. Some hyperparameters also enter the ML model, while others do not. The external hyperparameters that do not enter the ML model are clearly different from parameters, but influence the training process, e.g., the regularization hyperparameter in KRR [36].

In any case, (hyper)parameters can be fitted to attempt to reduce the generalization error of the ML model by minimizing the error in the validation set (Fig. 2). For so-called parametric models such as NNs, whose approximating function does not explicitly depend on the training points, finding (hyper)parameters reducing the validation error is usually the end of the story. However, nonparametric models such as kernel ridge regression (KRR) and Gaussian process regression (GPR) depend explicitly on the training points. In their case, not using the validation set for training the final ML model would lose valuable additional information available to the model and reduce its accuracy. Thus, after hyperparameters minimizing error in the validation set for models trained on the sub-training set are found (a procedure also known as model selection), MLatom uses them to train the final model on the whole training set.

During hyperparameter optimization in MLatom, by default, the root-mean-squared error (RMSE) is minimized, but the minimization of another type of error can be requested for native implementations. Alternatively, other defaults can be used by interfaces if they have their own hyperparameter optimization capabilities. When the ML model is trained on several different properties, the error (loss, L) should reflect the error for each of these properties. For example, for models trained on both property values and their derivatives, e.g., energies and energy gradients, the error in MLatom can be calculated as the sum of error for values (L_{val}) and weighted error for gradients in XYZ coordinates (L_{grxyz}) as typically done in the literature [37]:

$$L = L_{val} + w_{grxyz} L_{grxyz} \tag{1}$$

Although this approach gives the user additional flexibility, it has a drawback in that one has to choose an arbitrary parameter w_{grxyz}. To eliminate this parameter, we

introduce in MLatom the geometric mean of errors of different properties, which is used by default:

$$L = \sqrt{L_{val} L_{grxyz}} \tag{2}$$

The final model's accuracy also depends on how the training set is split into sub-training and validation sets; this topic is overviewed below in "Splitting and Sampling".

2.1.3 Estimating Accuracy of ML Models

MLatom also provides the means to estimate the accuracy (generalization error) of the ML model (Fig. 3). Since the validation set has already been used to create the ML model, it is good to use another independent test set to assess the model's performance [36]. The entire data set can be split into training and test sets, and points can be sampled into these subsets using procedures similar to those used for hyperparameter tuning (model selection; see "Splitting and Sampling"). Naturally, hyperparameter tuning and model evaluation can be combined in a single calculation task with MLatom.

2.1.4 Multi-step Tasks

The tasks above can be considered as basic. We now turn to describe multi-step tasks built upon these basic tasks. One such task is a Δ-learning task that combines predictions at the low-level QC method with ML corrections to make estimations approaching high-level QC accuracy [11]. Another is a self-correction task that combines several layers of ML models, with each layer correcting the previous layer's residual errors, which is useful for KRR models created with a focus on some region of the molecular PES [12]. Other multi-step tasks are learning curve generation and ML spectrum simulations [13], covered in the next two sub-sections in more detail.

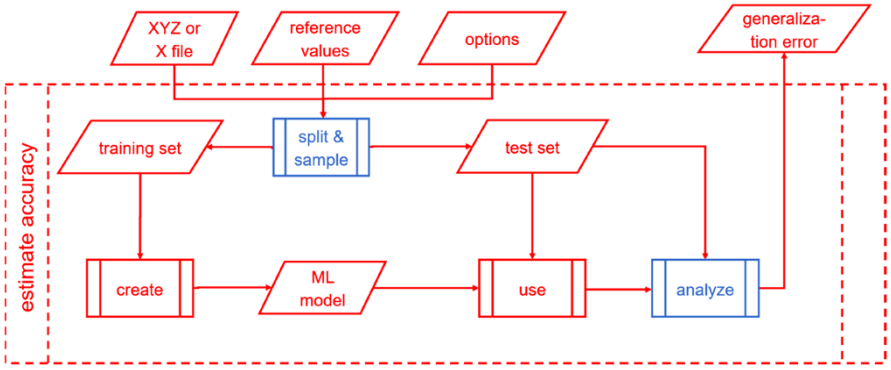

Fig. 3 Estimating the accuracy of the ML model

2.1.5 Learning Curves

Here, the concept of learning curves is used to investigate how ML generalization error depends on training set size. The relationship between ML error ε and training set size typically follows the power-law decay [40]:

$$\varepsilon = \varepsilon_{\mathrm{a}} + \frac{a}{N_{\mathrm{tr}}^{b}} \tag{3}$$

where ε_{a} is the asymptotic error in the limit of the infinitely large training set, a is a nonnegative amplitude, and b is the positive exponent telling us how fast the ML improves by training with more data. These three parameters define the learning curve, giving a more complete characterization of the performance of a given ML model type than a single generalization error estimated for one training set size.

Since the errors drop that fast, the learning curves are often plotted on a log–log scale. In this case, particularly for not too large training sets and small asymptotic errors, a linear curve is often observed [37]:

$$\log(\varepsilon) \approx \log(a) - b\log N_{\mathrm{tr}} \tag{4}$$

A learning curve cannot be drawn without the accuracy-estimating step (see "Estimating Accuracy of ML Models") being taken multiple times. Thus, we provide a dedicated task to automate this procedure (Fig. 4) in MLatom 2.

In the learning curve task, the accuracy of each training set size requested is examined in multiple repeats, where different training points are sampled. Testing with repeats helps to reduce the bias introduced by a specific combination of training and test sets, investigates the statistical variance, and reflects the robustness of an ML model. The results (RMSEs, wall-clock training and prediction times) from each test are recorded in the .csv database file.

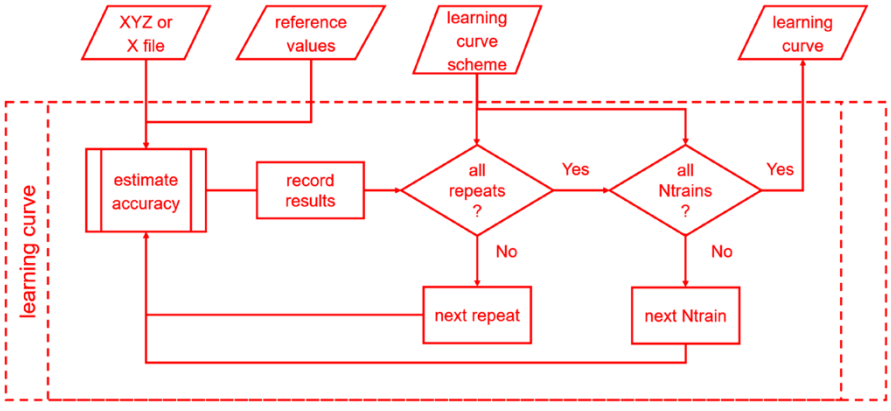

Fig. 4 Flowchart for the learning curve task

2.1.6 ML Nuclear Ensemble Spectra

Electronic spectrum simulation is yet another multi-step task that uses the ML-nuclear ensemble approach (ML-NEA) to automatically calculate quantities like absorption cross sections from as few QC calculations as possible [13]. This approach accelerates the traditional NEA, which usually requires thousands of points in a nuclear ensemble to generate a statistically converged spectrum [41]. Most nuclear ensemble points are relatively similar, making them suitable for using ML for efficient interpolation and replacing most QC calculations. Figure 5 shows this approach and its implementation in MLatom schematically.

The calculations require only an initial geometry, Gaussian 16 [42] input files for optimization and normal-mode calculations as well as for calculation of excited-state properties (excitation energies and oscillator strengths) with the QC method of choice. The user can also provide available pre-calculated data, such as output files with normal-mode calculations or nuclear ensemble geometries. Existing reference QC data can also be provided. MLatom has an interface to Gaussian 16, which automatically invokes and parses the QC calculations' output to get the equilibrium geometry and frequencies. Then, it passes the required data to the interface to Newton-X [43, 44] to generate a nuclear ensemble of 50k conformations sampled from a harmonic-oscillator Wigner distribution of the nuclear coordinates [45].

QC properties for training ML models are calculated iteratively. The number of training points is increased at each iteration to train $2N_{\mathrm{fs}}$ models for excitation energies (ΔE_{0n}) and oscillator strengths (f_{0n}) of transitions from ground-state to each excited state n. Then, we evaluate the convergence of ML predictions. We

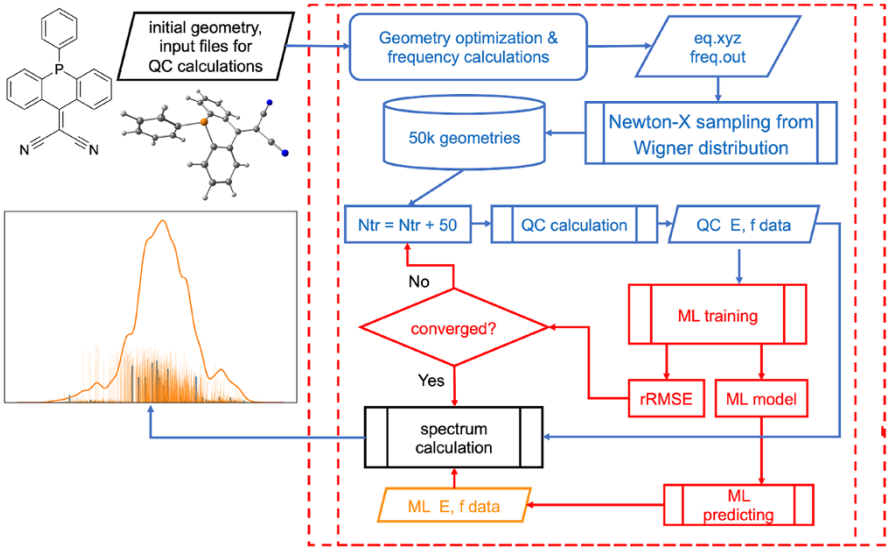

Fig. 5 *Left* Schematic representation of the machine learning–nuclear ensemble approach (ML-NEA). *Right* Implementation of ML-NEA for calculating absorption spectra. *Blue* quantum chemical (QC) data, *orange* ML