MESS: Modern Electronic Structure Simulations

Hatem Helal [1] [*]   Andrew Fitzgibbon [2]

Abstract

Electronic structure simulation (ESS) has been used for decades to provide quantitative scientific insights on an atomistic scale, enabling advances in chemistry, biology, and materials science, among other disciplines. Following standard practice in scientific computing, the software packages driving these studies have been implemented in compiled languages such as FORTRAN and C. However, the recent introduction of machine learning (ML) into these domains has meant that ML models must be coded in these languages, or that complex software bridges have to be built between ML models in Python and these large compiled software systems. This is in contrast with recent progress in modern ML frameworks which aim to optimise both ease of use and high performance by harnessing hardware acceleration of tensor programs defined in Python. We introduce MESS: a modern electronic structure simulation package implemented in JAX; porting the ESS code to the ML world. We outline the costs and benefits of following the software development practices used in ML for this important scientific workload. MESS shows significant speedups on widely available hardware accelerators and simultaneously opens a clear pathway towards combining ESS with ML. MESS is available at github.com/graphcore-research/mess.

1. Introduction

Approximating the quantum-mechanical behaviour of electrons in atoms, molecules, and solids is a long standing central challenge in the physical sciences (Dirac, 1929). Overcoming this challenge would provide a deeper understanding of the atomistic processes that underpin diverse phenomena. For example, these simulations could be used to efficiently screen drug candidates through direct modelling of their interactions with biological targets at the molecular level (Batra et al., 2020). In materials science, this understanding could accelerate the design of advanced materials for renewable energy by precisely tailoring their electronic properties (Gómez-Bombarelli et al., 2016; Shipley et al., 2021). Density Functional Theory (DFT) has emerged as a leading approach for ESS, striking a favourable balance between computational cost and accuracy when compared to more computationally demanding methods like coupled cluster theory or configuration interaction (Keith et al., 2021; Szabo & Ostlund, 2012).

A recent high-impact research direction in computational chemistry has been the development of simulation environments that integrate ML models (Keith et al., 2021; von Lilienfeld & Burke, 2020; Westermayr et al., 2021). While significant progress has been made by establishing complex software bridges between traditional ESS codes and ML frameworks, this approach presents a steep learning curve for researchers. These scientists must become proficient in not only electronic structure theory and its application to atomistic systems, but also the intricacies of ML model training and deployment, as well as high-performance scientific computing techniques. This multifaceted expertise requirement acts as a significant barrier, hindering the broader adoption and potential impact of these powerful simulation tools.

To address these challenges and accelerate this naturally interdisciplinary research, we introduce MESS: a Modern Electronic Structure Simulation framework. MESS is designed to bridge the gap between electronic structure methods and machine learning models by enabling their integration within a single unified environment.

We summarise our contributions as follows:

- Implementation of **end-to-end differentiable** electronic structure simulations, enabling algorithmic gradient evaluation for routine tasks such as electronic energy minimisation and atomic force evaluation.

- Development of a **high-throughput DFT simulation** framework leveraging hardware acceleration and batch processing for efficient exploration of configurational space.

- Creation of an **extensible and modular simulation**

environment that facilitates the seamless integration of machine learning models into electronic structure calculations.

MESS is released as an open-source Python package under the permissive MIT license, with all examples and benchmarks described here provided as executable notebooks for reproducible results and interactive exploration. [1] Additionally, by leveraging the powerful libraries of the JAX ecosystem, MESS demonstrates the potential of adapting software abstractions developed for machine learning to enhance and accelerate computational chemistry workloads.

2. Related Work

DFT in principle provides an exact solution to the electronic structure problem by placing the electron density on par with solving the Schrödinger equation for the many-body wavefunction (Hohenberg & Kohn, 1964; Kohn & Sham, 1965). This dramatic simplification is possible by the introduction of the exchange-correlation functional that accounts for the quantum-mechanical nature of electron-electron interactions. However, the exact form of the universal exchange-correlation functional is still unknown (Burke & Wagner, 2013). Practical implementations of DFT require selecting an approximate exchange-correlation functional, introducing a critical trade-off between accuracy and computational cost (Rappoport et al., 2008).

With hundreds of such functionals available, navigating the vast landscape of options and selecting the most appropriate form for a specific application can become a laborious task even for experts in the field (Lehtola et al., 2018; Rappoport et al., 2008). Numerous studies have attempted to benchmark and classify these functionals, offering guidance on their strengths and weaknesses for different types of systems and properties (Pribram-Jones et al., 2015; Peverati, 2021; Goerigk & Mehta, 2019; Kim et al., 2013; Kalita et al., 2022). Recent efforts have leveraged machine learning and offer a promising route towards a computationally efficient approximation to the universal functional (Kirkpatrick et al., 2021; Li et al., 2021).

While DFT simulations offer valuable insights into the electronic structure and static properties of molecules, they do not inherently capture the dynamic evolution of these systems over time. To study the time-dependent behavior of atomistic structures, such as the breaking and forming of chemical bonds, molecular dynamics (MD) simulations are employed (Eastman et al., 2017; Thompson et al., 2022; Case et al., 2023). In MD simulations, the forces acting on atoms can be derived from a variety of models, ranging from simple analytic functions to more sophisticated quantum-

mechanical calculations (Car, 2006). These forces are then used to numerically integrate the equations of motion, allowing the trajectories of atoms to be followed over time. This approach provides a powerful tool for investigating a wide range of phenomena, from chemical reactions to the self-assembly of complex materials (Lindorff-Larsen et al., 2011).

DFT-based MD simulations offer high accuracy in capturing the intricate electronic interactions that govern chemical bonding and reactivity. However, their widespread adoption has been hampered by the inherent computational cost of repeatedly solving the electronic structure problem at each time step. This limitation has typically restricted DFT-based MD simulations to relatively small systems and short timescales (Musaelian et al., 2023). To overcome this bottleneck and enable simulations of large-scale systems over extended periods, researchers have utilised computationally cheaper alternatives, such as hand-crafted interatomic potentials and targeting specialised hardware architectures (Klepeis et al., 2009; Shaw et al., 2010; 2014).

One promising solution that has emerged in recent years are Machine Learning Interatomic Potentials (MLIPs) (Behler & Parrinello, 2007; Bartók et al., 2010; Thompson et al., 2015; Schütt et al., 2018; Deringer et al., 2019; Drautz, 2019; von Lilienfeld & Burke, 2020; Batzner et al., 2022). These potentials leverage the flexibility of machine learning algorithms to learn complex relationships between atomic configurations and their corresponding energies and forces, as calculated by DFT. By utilising supervised learning on vast datasets of DFT simulations, MLIPs can accurately reproduce the potential energy surface of a system, enabling efficient and accurate MD simulations (Eastman et al., 2023; Batatia et al., 2023). MLIP-driven MD simulations offer a substantial advantage over pure DFT-based MD, enabling the study of significantly larger systems and longer timescales due to their reduced computational demands (Musaelian et al., 2023; Jia et al., 2020).

Various permutations of the combined features of hardware acceleration, automatic differentiation, and dynamic interpreted programs have been previously explored for ESS. TeraChem pioneered ESS on GPU accelerators by implementing these calculations directly in the CUDA framework (Seritan et al., 2021; Nickolls et al., 2008). Being a closed-source project inherently limits the scope of algorithmic research that can be explored with TeraChem but it has been applied to investigate quantum simulations of proteins (Kulik et al., 2012). PySCF supports modular ESS workflows through the familiar and easy-to-use NumPy interface (Sun et al., 2018). Extensions to PySCF have been developed to support automatic differentiation and GPU acceleration (Zhang & Chan, 2022; Wu et al., 2024). Automatic differentiation is one of the pillars of

---

[1] Notebooks are available as documentation pages at graphcore-research.github.io/mess

modern machine learning and has recently been explored for ESS (Baydin et al., 2018; Tan et al., 2023). JAX MD implements a differentiable simulation environment with straightforward interoperability between MLIPs and MD time-integrators (Schoenholz & Cubuk, 2020). Differentiable implementations for the Hartree-Fock method have been explored (Tamayo-Mendoza et al., 2018; Yoshikawa & Sumita, 2022). These projects demonstrated the applicability of automatic differentiation for solving this particular instantiation of electronic structure simulation. The DQC project demonstrated that a differentiable DFT implementation enables learning the exchange-correlation functional from data (Kasim et al., 2022). DQC uses the libcint library to evaluate the molecular integrals which prevents utilising modern hardware accelerators (Sun, 2015).

To the best of our knowledge, MESS is the first project to fully integrate end-to-end differentiable high-level programs with hardware-agnostic acceleration through compilation.

3. Electronic Structure Theory in a Nutshell

The central problem of electronic structure is solving the Schrödinger equation for the ground-state configuration of electrons given a molecular structure of interest. We will focus the discussion here on simulations within the Born-Oppenheimer approximation (Martin, 2020). The electron representation in MESS is flexible and the examples shared here use localised molecular orbitals, represented as a linear combination of atomic orbitals (LCAO) (Szabo & Ostlund, 2012)

$$\psi_i(\mathbf{r}) = \sum_\mu C_{\mu i} \phi_\mu(\mathbf{r} - \mathbf{R}_\mu; Z_\mu), \qquad (1)$$

where the $\{\psi_i\}$ are the set of orthonormal molecular orbitals that are expressed as an expansion in terms of a basis set of atomic orbitals $\{\phi_\mu\}$. Each atomic orbital ϕ_μ is associated with a unique atom which defines the centre $\mathbf{R}_\mu$ and the atomic number Z_μ. We have some freedom in how we select the basis set of atomic orbitals and, following standard practice, MESS uses Gaussian orbitals (Szabo & Ostlund, 2012). The specific parametrisation of the basis set is a critical decision that significantly impacts the accuracy and computational cost of the simulation. A plethora of precomputed basis sets are available, ranging from minimal sets designed for rapid calculations to larger, more comprehensive sets that offer increased accuracy at the expense of computational resources. MESS interfaces with the basis set exchange which provides the pre-computed parameters defining the atomic orbitals (Pritchard et al., 2019).

The molecular orbital coefficients can be arranged as a matrix $\mathbf{C}$ which leads us to the Roothaan equation (Roothaan, 1951; Lehtola et al., 2020)

$$\mathbf{F}(\mathbf{C})\mathbf{C} = \mathbf{S}\mathbf{C}\epsilon, \qquad (2)$$

where $\mathbf{F}$ is known as the Fock matrix which is a function of the molecular orbital coefficients $\mathbf{C}$, $\mathbf{S}$ is the overlap matrix, and ϵ is a diagonal matrix containing the molecular orbital energies. The elements of both the Fock matrix and the overlap matrix are found by integrating over combinations of the atomic orbitals. For example, an element of the overlap matrix is evaluated as:

$$S_{\mu\nu} = \int \phi_\mu^*(\mathbf{r})\phi_\nu(\mathbf{r})d\mathbf{r}. \qquad (3)$$

The Gaussian representation of atomic orbitals has proven to be particularly convenient as there are closed-form analytic solutions for the various integrals needed to evaluate Equation 2. MESS provides a basic implementation of these closed-form expressions for all the relevant integrals needed for a typical simulation (Taketa et al., 1966).

Equation 2 is a non-linear generalised eigenvalue problem since the Fock matrix $\mathbf{F}$ depends on the molecular orbital coefficients $\mathbf{C}$. Standard implementations will iteratively solve the Roothaan equation until finding the electronic energies converge to a fixed-point. This process is known as the self-consistent field (SCF) method (Lehtola et al., 2020).

MESS provides an SCF solver but the preferred implementation performs direct minimisation of the total energy functional:

$$\min_\mathbf{C} E(\mathbf{C}), \qquad (4)$$

subject to the orthonormality constraint of the molecular orbitals:

$$\mathbf{C}^\mathsf{T}\mathbf{S}\mathbf{C} = \mathbf{1}. \qquad (5)$$

There are many ways to approach solving this constrained optimisation problem. We provide one such implementation in the `minimise` function of MESS that uses the BFGS solver from the Optimistix library (Rader et al., 2024). The orthogonality constraint is incorporated by using the QR decomposition method as outlined in the DQC library (Kasim et al., 2022). Further, we demonstrate using first-order gradient methods in Section 5 based on the Adam optimiser (Kingma & Ba, 2014).

4. Software Architecture

MESS is designed to facilitate a new generation of hybrid electronic structure and machine learning simulations by providing a common framework that is hardware agnostic and optimised for current and future hardware. This forward-looking perspective is encapsulated by the term *modern* in MESS. Our goals extend beyond mere integration: we aim to make electronic structure methods more accessible, transparent, and, crucially for research progress, more *hackable*.

To accelerate progress towards these ambitious goals, we have adopted several key requirements, inspired by factors

that have driven recent advancements in machine learning across various domains:

- **Hardware Acceleration:** Efficient utilisation of modern hardware accelerators is essential for successfully scaling computationally intensive simulations.

- **Automatic Differentiation:** This capability enables seamless gradient calculations, which are vital for optimising complex models and exploring potential energy surfaces.

- **High-Level Interpreted Language:** A high-level language (e.g. Python) allows for rapid prototyping and experimentation, while modern machine learning compilers can mitigate the traditional performance concerns associated with interpreted languages.

Researchers accustomed to electronic structure codes implemented in compiled languages such as FORTRAN or C might be skeptical about the performance of interpreted languages. However, the advent of machine learning compilers has significantly narrowed the gap between high-level code and optimised machine code. These compilers excel at optimising tensor operations, the fundamental building blocks of many scientific computing and machine learning workloads, often achieving performance comparable to hand-tuned implementations (Tillet et al., 2019).

MESS is implemented entirely within the JAX framework, leveraging its rich ecosystem of libraries tailored for both machine learning and scientific computing (Bradbury et al., 2018; Frostig et al., 2018). To encapsulate simulation parameters and provide an intuitive interface, we adopt the "callable dataclass" abstraction from the Equinox library (Kidger & Garcia, 2021). This approach enhances code readability and maintainability while aligning with JAX's functional programming paradigm.

JAX provides the following higher-order function transformations that we use throughout MESS:

- `vmap`: Automatically vectorises functions, enabling efficient batch processing of multiple inputs.

- `grad`: Provides convenient and efficient Automatic Differentiation capabilities, allowing us to compute gradients of arbitrary order.

- `jit`: Just-in-time (JIT) compilation is a key feature of JAX that dynamically compiles Python functions into optimised machine code. By leveraging XLA, a high-performance compiler for linear algebra, `jit` significantly accelerates numerical computations in MESS.

- `pmap`: This transformation enables single-program, multiple-data (SPMD) parallel execution of functions across multiple devices (e.g. CPU, GPU, TPU, etc).

Broadly speaking, electronic structure simulation and machine learning model training share a common thread: both involve optimisation processes where the objective function that is being evaluated requires the interleaving of linear algebra, reductions, and point-wise operations. This fundamental similarity suggests that software abstractions and optimisation techniques developed for machine learning can be effectively applied to accelerate electronic structure calculations.

However, the rapidly evolving landscape of specialised hardware architectures and accompanying software frameworks presents a significant challenge for scientific software development. This constant change can dramatically influence the computational feasibility of various simulation methods. Choosing JAX as the foundation for MESS represents a strategic wager in this "hardware lottery" (Hooker, 2021). We posit that the substantial community interest in JAX-based machine learning models will drive the continued development and optimisation of the framework. Therefore, ensuring its optimisation on current hardware accelerators and further adaptation to future systems. This, in turn, will directly benefit MESS by enabling it to leverage cutting-edge hardware capabilities.

5. Motivating Examples

The examples that follow have been adapted from executable notebooks that are provided as part of the documentation for MESS. We have curated these examples to demonstrate how first-order gradient optimisation methods developed for machine learning can be applied to tackle complex electronic structure simulations. In addition to this, a preliminary benchmark has shown a promising speedup of roughly 16x compared to performing the same energy minimisation with PySCF. While this initial result is encouraging, a more comprehensive benchmarking study is needed to definitively characterise the performance of MESS. We provide this benchmark as an executable notebook [2], inviting community feedback and ensuring the replicability of our findings.

---

[2]The reported 16X performance speedup was measured for a single water molecule with the PBE density functional and the 6-31G basis set. MESS was executed on an NVIDIA A100 GPU accessed through Google Colab and PySCF was executed on the CPUs from the same runtime. Full setup and measurements are available at graphcore-research.github.io/mess/tour

5.1. Direct Minimisation of Density Functional

A central problem of DFT simulations is finding the charge density $\rho(\mathbf{r})$ that minimises the total energy functional. This is a constrained optimisation since the charge density is derived from a basis of molecular orbitals that are required to be orthonormal to produce a valid solution. Within MESS a single DFT calculation is setup by first defining the molecule, selecting a basis set, and building a `Hamiltonian` instance.

```
mol = from_pyquante("CH4")
basis = basisset(mol, "6-31g")
H = Hamiltonian(basis, xc_method="pbe")
```

This builds a single molecule of methane and represents the molecular orbitals with the 6-31G Pople basis set retrieved through the basis set exchange (Pritchard et al., 2019). The quantum-mechanical interactions are modelled within the Hamiltonian by selecting the `xc_method` argument which in the above example uses the widely PBE exchange-correlation functional (Perdew et al., 1996). The PBE functional incorporates a gradient expansion of the electron density and the implementation in MESS uses automatic differentiation to evaluate this gradient. In addition to various density functionals, MESS also implements the Hartree-Fock method, another widely used technique (Szabo & Ostlund, 2012). Users can select Hartree-Fock by setting the `xc_method="hfx"` parameter to incorporate the Hartree-Fock exchange energy in the electron-electron interaction.

There are many possible approaches to solving the constrained optimisation for the molecular orbital coefficients that minimise the total energy. The conjugate gradient method has been widely used in electronic structure simulations (Payne et al., 1992). Here we take a different approach and minimise the electronic energy using the Adam implementation from the Optax optimisation library (Kingma & Ba, 2014; DeepMind et al., 2020). The objective function we are minimising evaluates the total energy given a trial set of unconstrained parameters. The total energy is comprised of an electronic contribution that depends on the trial parameters and a static nuclear contribution that represents the electrostatic energy of assembling the positively charged nuclear cores.

```
@jax.jit
@jax.value_and_grad
def total_energy(Z):
    C = H.orthonormalise(Z)
    P = basis.density_matrix(C)
    return H(P) + nuclear_energy(mol)
```

Notice that the objective function explicitly applies the orthonormality constraint within the `orthonormalise` method of the Hamiltonian instance. The current implementation uses the QR decomposition algorithm as described in the DQC project to ensure that the orthonormality condition in Equation 5 is satisfied (Kasim et al., 2022). Applying `value_and_grad` as a function decorator will transform the objective function to calculate the gradient with respect to the input at the same point as evaluating total energy. As described Section 4 the `jit` function decorator is applied to optimise and compile the simulation so that it runs efficiently on the available hardware.

We use an arbitrary initial guess for the unconstrained parameters. Otherwise the following energy minimisation will appear similar to a standard neural-network training loop

```
Z = jnp.eye(basis.num_orbitals)
optimiser = optax.adam(learning_rate=0.1)
state = optimiser.init(Z)

for _ in range(200):
    e, grads = total_energy(Z)
    updates, state = optimiser.update(grads, state)
    Z = optax.apply_updates(Z, updates)
```

Figure 1 visualises how the energy varies through the optimisation process and shows the convergence of the total energy, which is exactly analogous to the analysis of the training loss curve in a neural-network.
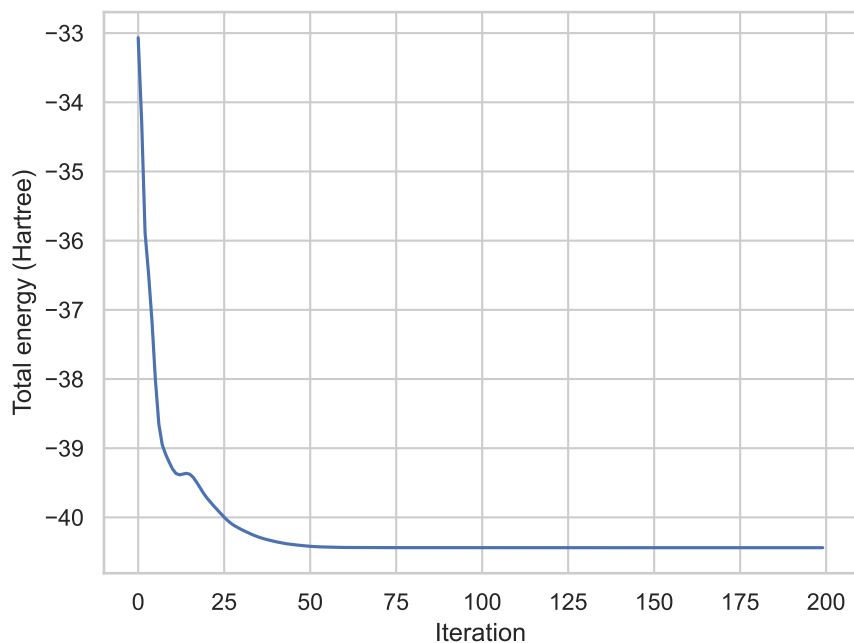
Figure 1. Total energy convergence for methane molecule with 6-31G basis set and PBE functional by direct optimisation using the Adam optimiser.

5.2. Batching over Conformations

Training neural networks over batches of data is an essential component when efficiently utilising massively parallel hardware accelerators. We can recast a standard electronic minimisation problem as a batched one by using the JAX vectorising map (`vmap`). By doing this we can parallelise the electronic structure optimisation over multiple conformations of the same molecule. Just like in training neural networks, this will allow for more efficient utilisation of the accelerator which in turn allows parallel sampling of potential energy surfaces using quantum-mechanical simulations.

We demonstrate this idea by calculating the molecular hydrogen dissociation curve using a batch of hydrogen molecules where the bond length (H-H distance) is varied. To setup the batch we build a `Hamiltonian` for each bond length and stack the built modules to create a batched-`Hamiltonian`. This example uses the minimal STO-3G basis set and the simple local density approximation (LDA) of DFT (Kohn & Sham, 1965; Parr & Yang, 1989). However, this batching formulation is not limited to these choices for how the `Hamiltonian` is represented.

```python
def h2_hamiltonian(r: float):
    mol = Structure(
        atomic_number=np.array([1, 1]),
        position=np.array([[0.0, 0.0, 0.0], [r, 0.0, 0.0]]),
    )
    basis = basisset(mol, basis_name="sto-3g")
    return Hamiltonian(basis, xc_method="lda")


num_confs = 64
rs = np.linspace(0.6, 10, num_confs)
H = [h2_hamiltonian(r) for r in rs]
H = jax.tree.map(lambda *xs: jnp.stack(xs), *H)
```

6

The following `energy` function evaluates the electronic energy of a single `Hamiltonian` for an unconstrained trial matrix **Z**. The `vmap` function transformation converts this function that evaluates the energy for a single `Hamiltonian` to process a batched-`Hamiltonian`. We also apply the `jit` function transformation to compile this function.

```python
@jax.jit
@jax.vmap
def energy(Z, H):
    C = H.orthonormalise(Z)
    P = H.basis.density_matrix(C)
    return H(P)
```

As described in Section 5.1, we convert the constrained optimisation over molecular orbital coefficients into an unconstrained one through the `orthonormalise` method of the Hamiltonian instance. We initialise the optimisation to simultaneously minimise the energy of the batch of molecular conformations.

```python
num_orbitals = H[0].basis.num_orbitals
Z = jnp.tile(jnp.eye(num_orbitals), (num_confs, 1, 1))
optimiser = optax.adam(learning_rate=0.1)
state = optimiser.init(Z)
```

We define a straightforward loss function that simply takes the sum over the energy for each molecular conformation in the batch. In basic terms, the optimisation will follow the gradient to minimise the loss. We use the transformation `value_and_grad` as a function decorator on this loss function to evaluate the loss and the corresponding gradient with automatic differentiation.

```python
@jax.value_and_grad
def loss_fn(z, h):
    return jnp.sum(energy(z, h))
```

Just as in Section 5.1, the optimisation looks familiar to a basic neural-network training loop

```python
for _ in range(128):
    loss, grads = loss_fn(Z, H)
    updates, state = optimiser.update(grads, state)
    Z = optax.apply_updates(Z, updates)
```

Our loss function is only minimising the electronic energy. In order to calculate the total energy we must add the electrostatic contribution that arises from each arrangement of nuclear charges. We can evaluate this additive constant using the `vmap` transformation to calculate the nuclear energy for the entire batch of Hamiltonians

```python
E_nuclear = jax.vmap(nuclear_energy)(H.basis.structure)
E_total = energy(Z, H) + E_nuclear
```

In Figure 2 we visualise the batched total loss as well as the converged bond dissociation curve which is evaluated entirely in parallel within our batch processing implementation for DFT. As far as we are aware, MESS is the first demonstration of a batched implementation for DFT with full parallel evaluation. We envision that batched implementations such as this can be used as a building block for high-throughput DFT simulations for generating large-scale datasets.
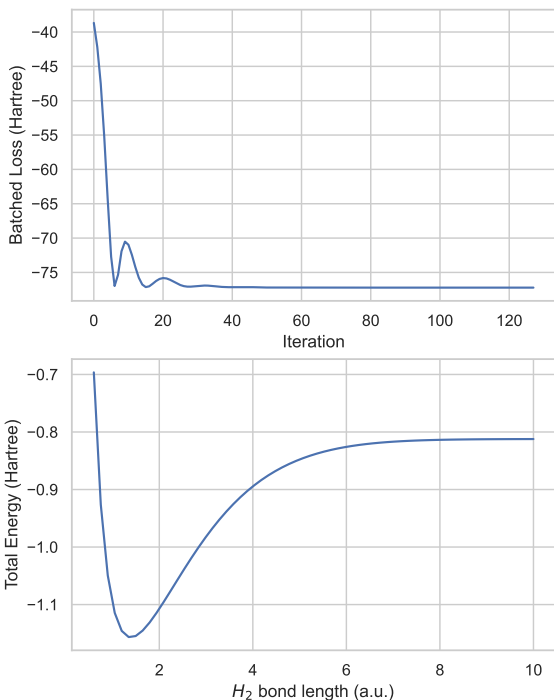
Figure 2. Batched evaluation of DFT energy for H_2 molecule within LDA approximation and the minimal STO-3G basis set.

6. Concluding Thoughts

We have presented MESS, a unified simulation environment designed to remove any friction for adopting machine learning within electronic structure simulations. One of MESS's key features is the ability to leverage the JAX ecosystem, enabling hardware acceleration, automatic differentiation and optimisation through high-level function transformations.

Another promising direction for MESS is accelerating algorithmic research in electronic structure. Significant progress has been made in reformulating DFT to scale linearly with the number of atoms in the simulation rather than the cubic scaling of standard DFT implementations (Bowler & Miyazaki, 2012; Kohn, 1996; Prodan & Kohn, 2005). These efforts have demonstrated that scaling electronic structure can be essential to elucidating complex phenomena across biochemistry and materials (Cole & Hine, 2016). We see MESS as an ideal environment for exploring novel electron density representations that can satisfy the requirements for achieving linear scaling DFT.

Software and Data

Source code for MESS is available on GitHub at the following address: github.com/graphcore-research/mess

Executable notebooks described here and API documentation for MESS are available at the following address: graphcore-research.github.io/mess

Acknowledgements

Impact Statement

This paper presents work whose goal is to advance the capabilities of electronic structure simulations enhanced by machine learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Bartók, A. P., Payne, M. C., Kondor, R., and Csányi, G. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Physical review letters*, 104(13):136403, 2010.

Batatia, I., Benner, P., Chiang, Y., Elena, A. M., Kovács, D. P., Riebesell, J., Advincula, X. R., Asta, M., Baldwin, W. J., Bernstein, N., et al. A foundation model for atomistic materials chemistry. *arXiv preprint arXiv:2401.00096*, 2023.

Batra, R., Chan, H., Kamath, G., Ramprasad, R., Cherukara, M. J., and Sankaranarayanan, S. K. Screening of therapeutic agents for covid-19 using machine learning and ensemble docking studies. *The journal of physical chemistry letters*, 11(17):7058–7065, 2020.

Batzner, S., Musaelian, A., Sun, L., Geiger, M., Mailoa, J. P., Kornbluth, M., Molinari, N., Smidt, T. E., and Kozinsky, B. E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature communications*, 13(1):2453, 2022.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18(153): 1–43, 2018.

Behler, J. and Parrinello, M. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.*, 98:146401, Apr 2007.

Bowler, D. R. and Miyazaki, T. Methods in electronic structure calculations. *Reports on Progress in Physics*, 75(3):036503, 2012.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Burke, K. and Wagner, L. O. Dft in a nutshell. *International Journal of Quantum Chemistry*, 113(2):96–101, 2013.

Car, R. Ab initio molecular dynamics: Dynamics and thermodynamic properties. *Contemporary Concepts of Condensed Matter Science*, 2:55–95, 2006.

Case, D. A., Aktulga, H. M., Belfon, K., Cerutti, D. S., Cisneros, G. A., Cruzeiro, V. W. D., Forouzesh, N., Giese, T. J., Gotz, A. W., Gohlke, H., et al. Ambertools. *Journal of chemical information and modeling*, 63(20):6183–6191, 2023.

Cole, D. J. and Hine, N. D. Applications of large-scale density functional theory in biology. *Journal of Physics: Condensed Matter*, 28(39):393001, 2016.

DeepMind, Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., Buchlovsky, P., Budden, D., Cai, T., Clark, A., Danihelka, I., Dedieu, A., Fantacci, C., Godwin, J., Jones, C., Hemsley, R., Hennigan, T., Hessel, M., Hou, S., Kapturowski, S., Keck, T., Kemaev, I., King, M., Kunesch, M., Martens, L., Merzic, H., Mikulik, V., Norman, T., Papamakarios, G., Quan, J., Ring, R., Ruiz, F., Sanchez, A., Sartran, L., Schneider, R., Sezener, E., Spencer, S., Srinivasan, S., Stanojević, M., Stokowiec, W., Wang, L., Zhou, G., and Viola, F. The DeepMind JAX Ecosystem, 2020. URL http://github.com/google-deepmind.

Deringer, V. L., Caro, M. A., and Csányi, G. Machine learning interatomic potentials as emerging tools for materials science. *Advanced Materials*, 31(46):1902765, 2019.

Dirac, P. A. M. Quantum mechanics of many-electron systems. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 123(792):714–733, 1929.

Drautz, R. Atomic cluster expansion for accurate and transferable interatomic potentials. *Physical Review B*, 99(1):014104, 2019.

Eastman, P., Swails, J., Chodera, J. D., McGibbon, R. T., Zhao, Y., Beauchamp, K. A., Wang, L.-P., Simmonett, A. C., Harrigan, M. P., Stern, C. D., et al. Openmm 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS computational biology*, 13(7):e1005659, 2017.

Eastman, P., Behara, P. K., Dotson, D. L., Galvelis, R., Herr, J. E., Horton, J. T., Mao, Y., Chodera, J. D., Pritchard, B. P., Wang, Y., et al. Spice, a dataset of drug-like molecules and peptides for training machine learning potentials. *Scientific Data*, 10(1):11, 2023.

Frostig, R., Johnson, M. J., and Leary, C. Compiling machine learning programs via high-level tracing. *Systems for Machine Learning*, 4(9), 2018.

Goerigk, L. and Mehta, N. A trip to the density functional theory zoo: warnings and recommendations for the user. *Australian Journal of Chemistry*, 72(8):563–573, 2019.

Gómez-Bombarelli, R., Aguilera-Iparraguirre, J., Hirzel, T. D., Duvenaud, D., Maclaurin, D., Blood-Forsythe, M. A., Chae, H. S., Einzinger, M., Ha, D.-G., Wu, T., et al. Design of efficient molecular organic light-emitting diodes by a high-throughput virtual screening and experimental approach. *Nature materials*, 15(10):1120–1127, 2016.

Hohenberg, P. and Kohn, W. Inhomogeneous electron gas. *Physical review*, 136(3B):B864, 1964.

Hooker, S. The hardware lottery. *Communications of the ACM*, 64(12):58–65, 2021.

Jia, W., Wang, H., Chen, M., Lu, D., Lin, L., Car, R., Weinan, E., and Zhang, L. Pushing the limit of molecular dynamics with ab initio accuracy to 100 million atoms with machine learning. In *SC20: International conference for high performance computing, networking, storage and analysis*, pp. 1–14. IEEE, 2020.

Kalita, B., Pederson, R., Chen, J., Li, L., and Burke, K. How well does kohn–sham regularizer work for weakly correlated systems? *The Journal of Physical Chemistry Letters*, 13(11):2540–2547, 2022.

Kasim, M. F., Lehtola, S., and Vinko, S. M. Dqc: A python program package for differentiable quantum chemistry. *The Journal of chemical physics*, 156(8), 2022.

Keith, J. A., Vassilev-Galindo, V., Cheng, B., Chmiela, S., Gastegger, M., Muller, K.-R., and Tkatchenko, A. Combining machine learning and computational chemistry for predictive insights into chemical systems. *Chemical reviews*, 121(16):9816–9872, 2021.

Kidger, P. and Garcia, C. Equinox: neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming workshop at Neural Information Processing Systems 2021*, 2021.

Kim, M.-C., Sim, E., and Burke, K. Understanding and reducing errors in density functional calculations. *Physical review letters*, 111(7):073003, 2013.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kirkpatrick, J., McMorrow, B., Turban, D. H., Gaunt, A. L., Spencer, J. S., Matthews, A. G., Obika, A., Thiry, L., Fortunato, M., Pfau, D., et al. Pushing the frontiers of density functionals by solving the fractional electron problem. *Science*, 374(6573):1385–1389, 2021.

Klepeis, J. L., Lindorff-Larsen, K., Dror, R. O., and Shaw, D. E. Long-timescale molecular dynamics simulations of protein structure and function. *Current opinion in structural biology*, 19(2):120–127, 2009.

Kohn, W. Density functional and density matrix method scaling linearly with the number of atoms. *Physical Review Letters*, 76(17):3168, 1996.

Kohn, W. and Sham, L. J. Self-consistent equations including exchange and correlation effects. *Physical review*, 140(4A):A1133, 1965.

Kulik, H. J., Luehr, N., Ufimtsev, I. S., and Martinez, T. J. Ab initio quantum chemistry for protein structures. *The Journal of Physical Chemistry B*, 116(41):12501–12509, 2012.

Lehtola, S., Steigemann, C., Oliveira, M. J., and Marques, M. A. Recent developments in libxc—a comprehensive library of functionals for density functional theory. *SoftwareX*, 7:1–5, 2018.

Lehtola, S., Blockhuys, F., and Van Alsenoy, C. An overview of self-consistent field calculations within finite basis sets. *Molecules*, 25(5):1218, 2020.

Li, L., Hoyer, S., Pederson, R., Sun, R., Cubuk, E. D., Riley, P., Burke, K., et al. Kohn-sham equations as regularizer: Building prior knowledge into machine-learned physics. *Physical review letters*, 126(3):036401, 2021.

Lindorff-Larsen, K., Piana, S., Dror, R. O., and Shaw, D. E. How fast-folding proteins fold. *Science*, 334(6055):517–520, 2011.

Martin, R. M. *Electronic structure: basic theory and practical methods*. Cambridge university press, 2020.

Musaelian, A., Batzner, S., Johansson, A., and Kozinsky, B. Scaling the leading accuracy of deep equivariant models to biomolecular simulations of realistic size. In *SC23: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–12. IEEE, 2023.

Nickolls, J., Buck, I., Garland, M., and Skadron, K. Scalable parallel programming with cuda: Is cuda the parallel programming model that application developers have been waiting for? *Queue*, 6(2):40–53, 2008.

Parr, R. G. and Yang, W. *Density-Functional Theory of Atoms and Molecules*, volume 16. OUP USA, 1989.

Payne, M. C., Teter, M. P., Allan, D. C., Arias, T. A., and Joannopoulos, J. D. Iterative minimization techniques for ab initio total-energy calculations: molecular dynamics and conjugate gradients. *Rev. Mod. Phys.*, 64:1045–1097, Oct 1992.

Perdew, J. P., Burke, K., and Ernzerhof, M. Generalized gradient approximation made simple. *Physical review letters*, 77(18):3865, 1996.

Peverati, R. Fitting elephants in the density functionals zoo: Statistical criteria for the evaluation of density functional theory methods as a suitable replacement for counting parameters. *International Journal of Quantum Chemistry*, 121(1):e26379, 2021.

Pribram-Jones, A., Gross, D. A., and Burke, K. Dft: A theory full of holes? *Annual review of physical chemistry*, 66:283–304, 2015.

Pritchard, B. P., Altarawy, D., Didier, B., Gibson, T. D., and Windus, T. L. New basis set exchange: An open, up-to-date resource for the molecular sciences community. *Journal of chemical information and modeling*, 59(11):4814–4820, 2019.

Prodan, E. and Kohn, W. Nearsightedness of electronic matter. *Proceedings of the National Academy of Sciences*, 102(33):11635–11638, 2005.

Rader, J., Lyons, T., and Kidger, P. Optimistix: modular optimisation in jax and equinox. *arXiv:2402.09983*, 2024.

Rappoport, D., Crawford, N. R., Furche, F., and Burke, K. Which functional should i choose. *Computational Inorganic and Bioinorganic Chemistry*, 594:1–25, 2008.

Roothaan, C. C. J. New developments in molecular orbital theory. *Reviews of modern physics*, 23(2):69, 1951.

Schoenholz, S. and Cubuk, E. D. Jax md: a framework for differentiable physics. *Advances in Neural Information Processing Systems*, 33:11428–11441, 2020.

Schütt, K. T., Sauceda, H. E., Kindermans, P.-J., Tkatchenko, A., and Müller, K.-R. Schnet–a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24), 2018.

Seritan, S., Bannwarth, C., Fales, B. S., Hohenstein, E. G., Isborn, C. M., Kokkila-Schumacher, S. I., Li, X., Liu, F., Luehr, N., Snyder Jr, J. W., et al. Terachem: A graphical processing unit-accelerated electronic structure package for large-scale ab initio molecular dynamics. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 11(2):e1494, 2021.

Shaw, D. E., Maragakis, P., Lindorff-Larsen, K., Piana, S., Dror, R. O., Eastwood, M. P., Bank, J. A., Jumper, J. M., Salmon, J. K., Shan, Y., et al. Atomic-level characterization of the structural dynamics of proteins. *Science*, 330 (6002):341–346, 2010.

Shaw, D. E., Grossman, J., Bank, J. A., Batson, B., Butts, J. A., Chao, J. C., Deneroff, M. M., Dror, R. O., Even, A., Fenton, C. H., et al. Anton 2: raising the bar for performance and programmability in a special-purpose molecular dynamics supercomputer. In *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 41–53. IEEE, 2014.

Shipley, A. M., Hutcheon, M. J., Needs, R. J., and Pickard, C. J. High-throughput discovery of high-temperature conventional superconductors. *Physical Review B*, 104 (5):054501, 2021.

Sun, Q. Libcint: An efficient general integral library for gaussian basis functions. *Journal of computational chemistry*, 36(22):1664–1671, 2015.

Sun, Q., Berkelbach, T. C., Blunt, N. S., Booth, G. H., Guo, S., Li, Z., Liu, J., McClain, J. D., Sayfutyarova, E. R., Sharma, S., et al. Pyscf: the python-based simulations of chemistry framework. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 8(1):e1340, 2018.

Szabo, A. and Ostlund, N. S. *Modern quantum chemistry: introduction to advanced electronic structure theory*. Courier Corporation, 2012.

Taketa, H., Huzinaga, S., and O-ohata, K. Gaussian-expansion methods for molecular integrals. *Journal of the physical society of Japan*, 21(11):2313–2324, 1966.

Tamayo-Mendoza, T., Kreisbeck, C., Lindh, R., and Aspuru-Guzik, A. Automatic differentiation in quantum chemistry with applications to fully variational hartree–fock. *ACS central science*, 4(5):559–566, 2018.

Tan, C. W., Pickard, C. J., and Witt, W. C. Automatic differentiation for orbital-free density functional theory. *The Journal of Chemical Physics*, 158(12), 2023.

Thompson, A. P., Swiler, L. P., Trott, C. R., Foiles, S. M., and Tucker, G. J. Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials. *Journal of Computational Physics*, 285:316–330, 2015.

Thompson, A. P., Aktulga, H. M., Berger, R., Bolintineanu, D. S., Brown, W. M., Crozier, P. S., In't Veld, P. J., Kohlmeyer, A., Moore, S. G., Nguyen, T. D., et al. Lammps-a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum

scales. *Computer Physics Communications*, 271:108171, 2022.

Tillet, P., Kung, H.-T., and Cox, D. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019.

von Lilienfeld, O. A. and Burke, K. Retrospective on a decade of machine learning for chemical discovery. *Nature communications*, 11(1):4895, 2020.

Westermayr, J., Gastegger, M., Schütt, K. T., and Maurer, R. J. Perspective on integrating machine learning into computational chemistry and materials science. *The Journal of Chemical Physics*, 154(23), 2021.

Wu, X., Sun, Q., Pu, Z., Zheng, T., Ma, W., Yan, W., Yu, X., Wu, Z., Huo, M., Li, X., et al. Python-based quantum chemistry calculations with gpu acceleration. *arXiv preprint arXiv:2404.09452*, 2024.

Yoshikawa, N. and Sumita, M. Automatic differentiation for the direct minimization approach to the hartree–fock method. *The Journal of Physical Chemistry A*, 126(45): 8487–8493, 2022.

Zhang, X. and Chan, G. K.-L. Differentiable quantum chemistry with pyscf for molecules and materials at the mean-field level and beyond. *The Journal of Chemical Physics*, 157(20):204801, 2022.