ARCHER2-eCSE01-16 Report
Achieving the sustainability and scalability of numeric-atomic-orbital-based linear response and electron-phonon functionality in FHI-Aims

Staff Member: Connor L. Box
PI: Reinhard J. Maurer
Project Partners: Andrew Logsdail, Volker Blum, Mariana Rossi, Christian Carbogno

November 1, 2021

Abstract

FHI-Aims is a numeric atomic-orbital based electronic structure code to perform density functional theory calculations and beyond. A proof-of-principle first order response module was previously implemented based on density functional perturbation theory, but in its fractured and redundant form it was unmaintanable and largely unused. We have performed a large-scale refactorisation of this code to provide consistent functionality, to increase scalability and memory efficiency. The new module framework is well documented and easily extendable. This is shown by the implementation of a new driver module for electron-phonon response and by interfacing the module with the distributed matrix library ELSI and the LibXC library. We further provide showcase applications and performance graphs on ARCHER2 for the redesigned functionality.

1 Project Objectives and State-of-the-Art

Density Functional Perturbation Theory. For the calculation of vibrational frequencies and phonon band-structures or molecular polarizabilities, the response of the electronic structure to a nuclear displacement (first order derivatives $\partial H_{ij}/\partial R_I$) or to an applied electric field (derivatives $\partial H_{ij}/\partial \mu$) is needed. These derivatives can be calculated in the framework of density-functional perturbation theory (DFPT) assuming a linear response to the external perturbation (e.g. lattice displacement, electric field, etc.). While different types of external

perturbations require different integrals to be evaluated, DFPT sits on top of this as a general framework that is valid for numerous response properties (see Figure 1). [1] Following a density functional theory (DFT) calculation which provides the ground-state electron density $\rho(r)$, the calculation of a response property involves the calculation of perturbation-specific properties, before a general DFPT calculation is performed. Similar to the self-consistent-field (SCF) algorithm that underpins DFT calculations, DFPT needs to self-consistently converge response properties via the Coupled Perturbed SCF algorithm (CPSCF), such as the density response $\rho^{(1)}$ (r), the first order response Hamiltonian $\boldsymbol{H}^{(1)}$ and wave function expansion coefficients $\boldsymbol{U}^{(1)}$. This sequence is universal and the only property-specific aspects are integrations over the relevant perturbation operator (for example during the construction of $\boldsymbol{H}^{(1)}$). Once $\rho^{(1)}$ and $\boldsymbol{H}^{(1)}$ are converged, the relevant response properties such as phonon bandstructures, phonon linewidth [2], or polarizability [3] can be calculated.

FHI-Aims is a numeric atomic-orbital based electronic structure code.[4] The implementation of DFPT in FHI-Aims is the first all-electron real-space atom-centred implementation of linear response in a periodic electronic structure code. In its original form it was implemented for atomic response (vibrations and phonons) [5] and electric field response. [3] The code is a naturally grown community development by several authors. The code featured heavy code duplication. As shown in Figure 2 multiple run modes for each type of response calculation existed that duplicate the central coupled perturbed self-consistent field (CPSCF) routines shown in Figure 1. Runmodes differentiate between periodic, aperiodic, and real-space supercell calculations as described by Shang et al. [5]. In the latter case, the atomic displacement response is calculated for the extended crystal volume in terms of local atom perturbations, which can be interpolated in reciprocal space to achieve denser Brillouin sampling than otherwise achievable.[6, 7] Runmodes are further differentiated between conventional full implementations of aperiodic and periodic cases, runmodes where only 1 response component is constructed at a time to reduce memory requirements ("reduced memory" in Figure 2), and runmodes that (partially) employ distributed matrix parallelism via the ScaLAPACK/BLACS protocol. All these modes were collected in different subfolders ("DFPT_<runmode>") within the main source code directory of FHI-Aims, with each folder containing copies of the key matrix evaluation routines whilst the ScaLAPACK/BLACS versions were located within the general purpose ScaLAPACK wrapper file within FHI-Aims. Additionally, some runmodes used evaluation routines from other runmode folders, resulting in significant cross-linking.

The aim of this project was to improve the DFPT implementation in FHI-Aims to deliver a long-term sustainable code framework with consistent functionality. The specific **objectives** of this project were:

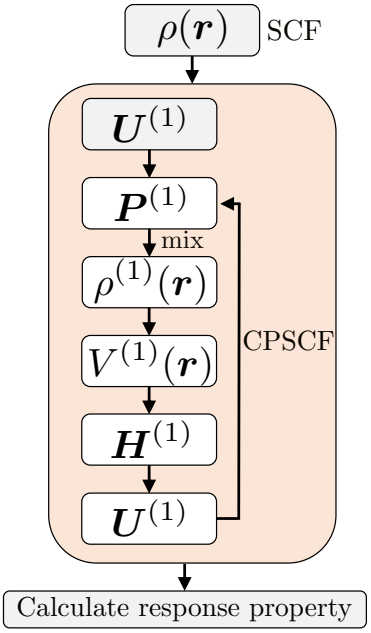

Figure 1: Flowchart of CPSCF cycle, which is central to DFPT linear response calculations. First the ground state density, $\rho(\boldsymbol{r})$ is calculated during a normal SCF cycle. Then an initial guess for the first order coefficients, $\boldsymbol{U}^{(1)}$ is provided. The CPSCF cycle begins with the calculation of the first order density matrix, $\boldsymbol{P}^{(1)}$ which is then mixed with the previous iteration. The first order electron density, $\rho^{(1)}(\boldsymbol{r})$ and potential $V^{(1)}(\boldsymbol{r})$ are calculated before the first order Hamiltonian, $\boldsymbol{H}^{(1)}$ and $\boldsymbol{U}^{(1)}$ can be calculated. If $\boldsymbol{P}^{(1)}$ is converged, the response quantities are passed to the respective driver to evaluate the response property. Otherwise the cycle is repeated until convergence is achieved.

1. Refactorizing the existing DFPT code to achieve long-term sustainability and code efficiency

2. Improve scalability and transferability of DFPT functionality by integration into peta- and (pre-)exa-scale electronic structure interface ELSI

3. Extend DFPT functionality to support Generalized-Gradient-Approximation (GGA) DFT functionals and van-der-Waals-compliant exchange-correlation functionals

4. Developing new driver modules for electron-phonon coupling and electronic friction to enable new scientific applications in energy materials and catalysis

The project was delivered in the period from November 2020 to October 2021. All code

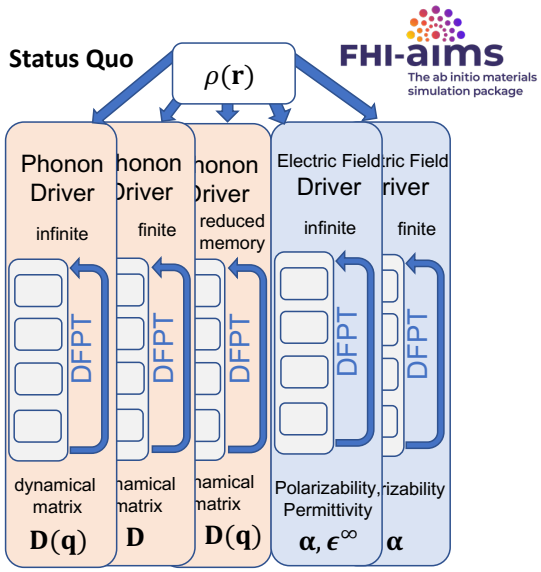

Figure 2: Original DFPT code in FHI-Aims, featuring heavy duplication of the DFPT infrastructure across different modules.

changes that occurred in that time and that are described in this document have been merged into the main branch of the official FHI-Aims GitLab repository [8] in October 2021 and will be included in the next release of FHI-Aims.

2 Objective 1: Refactorization of the existing DFPT code

Connor Box has refactored the DFPT code in FHI-Aims according to the diagram shown in Figure 3. The old infrastructure continues to exist in the original folder structure (labeled as "Legacy" in the figure), but will be discontinued over the course of the next 12 months after rigorous testing of the new code framework. The DFPT calculations are governed by a central "linear response wrapper" function that is called after the ground-state Kohn-Sham DFT calculation. This function can call different application models, which currently include "electric response", (polarisability, dielectric function) "atomic response" (phonon spectra and molecular vibrations), and electronic friction (electron-phonon coupling) calculations. This portfolio of options can easily be extended further, as the realised infrastructure serves as well documented template for future application modules of the central DFPT infrastructure. An example for a future extension into the central DFPT framework is given in the figure as the random phase approximation (RPA) functionality, which uses CPSCF to calculate important

input quantities.

Each application module on the left hand side calls into a central DFPT module (shown on the right hand side of Figure 3), which performs the universal CPSCF operations and constructs the required first order response quantities. This module also features the key interfaces to external libraries. This includes the ELSI library,[9] which is used to enable a restart of the CPSCF routine by exporting the first order density matrix. Another interface has been implemented to the LibXC library [10] to enable the calculation of the first order Hamiltonian for arbitrary LDA, GGA, and hybrid-level exchange-correlation functionals. Routines that are a significant bottle necks in memory distribution are shaded yellow, whilst routines with optimised memory distribution (even when $N_{\text{proc}} > N_k$) are shaded blue. The figure also depicts routines that currently support 2 spin channels (i.e spin collinear), `use_local_index` (for local dense real-space matrices, this reduces memory usage and can improve efficiency, particularly for large systems) and `collect_eigenvectors .false.` (for ScaLAPACK-type calculations, this reduces memory usage signficantly). We note that these features are not supported for the full real-space supercell approach as described by Shang et al. [5] We describe the specifics of the supercell approach below in the future work section.

Currently the user interface is still using the legacy keywords to control the driver modules, however, in the coming months we aim to migrate to new keywords that properly reflect the new driver and general CPSCF relationship, with keywords that control the CPSCF module that are no longer specific to the driver used, as well as a consolidation of the keywords general to atomic response, and general to electric field response. The current keywords to control the CPSCF are given in Table 1, which are mainly for convergence thresholds, first order density mixing and restart behaviour. Importantly, the new interface detailed here is not employed unless `DFPT_centralised` is set to `.true.`. A significant amount of error catching and messages for the user at runtime have been added, whereas the legacy code would often crash without warning. This makes the code far more useable. We aim to shortly provide a tutorial for use of the infrastructure on the FHI-Aims Gitlab Wiki page.

The newly refactored infrastructure is significantly more compact and easier to maintain. The total number of code lines has been reduced by just over 60% for the CPSCF code, with the newer inteface also including more features. Interfacing to the remaining FHI-Aims DFPT routines (such as RPA force code) will further reduce the total number of code lines. Specific runmode cases are now only dealt with at the most bottom level and high level developments at the level of the CPSCF cycle are decoupled. Each evaluation routine version for the key matrices are held in the same file, so that future developers understand that these are performing a common task. Additionally, several versions of some routines have been consolidated (e.g

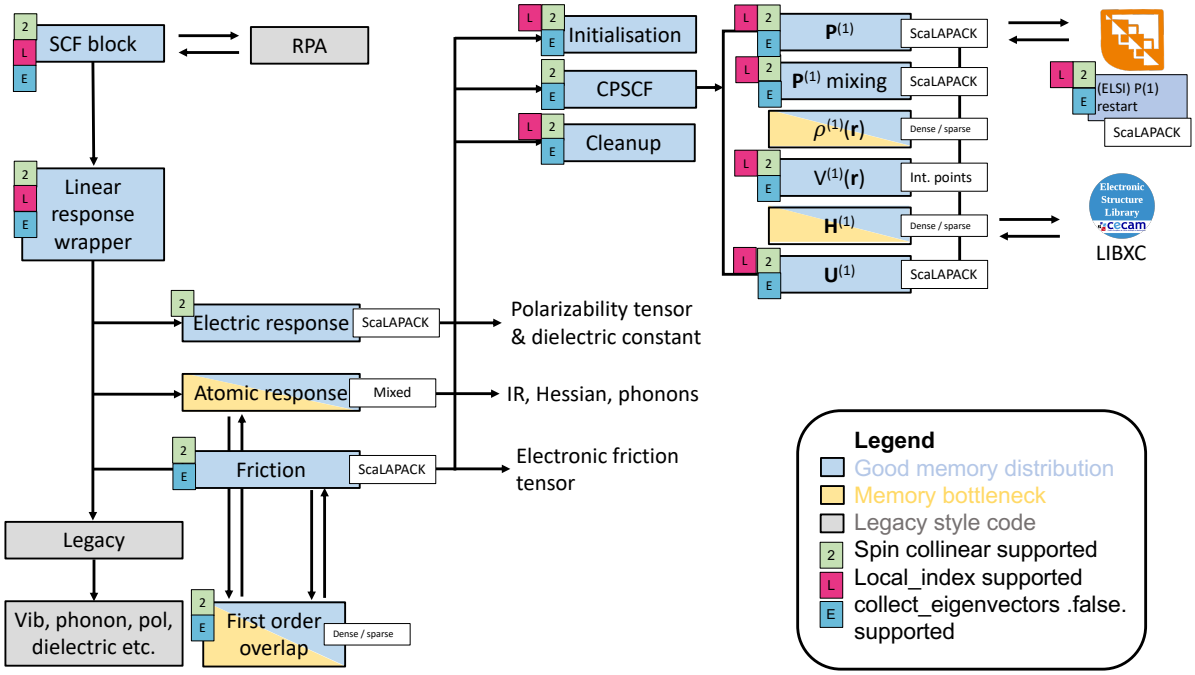

Figure 3: General code layout of DFPT code within FHI-Aims (excluding the mostly separate magnetic response implementation which only shares the Pulay mixing routine). Note that some depicted features may not be present for the supercell approach. "Dense / sparse" refers to when the full global dense (aperiodics) or sparse (periodics) response matrices are needed, "Int. points" refers to distribution over integration grid points and "ScaLAPACK" refers to scaLAPACK support for distribution of matrices. Mixed coloured regions currently have have a memory bottleneck for aperiodic systems due to the use of full dense matrices.

the ScaLAPACK version of first order coefficient matrix evaluation handles both atomic and electric field response). Hosting the ScaLAPACK versions of the evaluation routines in these files is an additional improvement for future development, as previously these were hosted within a general ScaLAPACK wrapper within FHI-Aims which is completely independent of DFPT calculations. Matrix construction routines for BLACS distributed response matrices were hardcoded for the specific response matrix, and so there were many repeats of essentially the same routine within the general ScaLAPACK wrapper. Now we have created a single file hosting general matrix constructions that is called many times within the CPSCF and driver infrastructure, this further improves readability and maintainability.

Table 1: User controlled parameters. The default of DFPT_centralised will soon be changed.

Variable	Type	Description	Default
DFPT_sc_accuracy_dm	real	Accuracy threshold for $\boldsymbol{P}^{(1)}$. Different units for Atomic vs Efield response	$1e-3$
DFPT_mixer	string	Type of mixer employed (linear and pulay supported). Linear just sets dfpt_pulay_steps to 1	pulay
dfpt_pulay_steps	integer	Number of steps to use in Pulay mixer.	8
DFPT_mixing	real	$\boldsymbol{P}^{(1)}$ mixing parameter (linear or pulay).	0.2
DFPT_restart_read_DM1	logical	Whether to read $\boldsymbol{P}^{(1)}$ from a previously outputted file on the first CP-SCF iterations	.false.
DFPT_restart_write_DM1	logical	Whether to write $\boldsymbol{P}^{(1)}$ every CP-SCF iteration for every cycle	.false.
DFPT_centralised	logical	Whether to use this interface (.true.) or the original H. Shang interfaces (.false.)	.false.

Array storage

The new DFPT implementation has a unified DFPT module, but still differentiates between certain runmodes. However, it does so at a lowest function level at which response matrices are evaluated and integrated. Every first order response array will feature n_dfpt_coords and atom_dim as leading dimensions, as well as n_spin if appropriate. Reciprocal space dense matrices are used (with or without ScaLAPACK distributed) for aperiodic systems, whilst periodics additionally include a mix of real-space sparse matrices. A full list of array dimensions for each runmode is provided in Table 7 in the appendix.

Every core routine has been developed to calculate a response property to a single perturbation (i.e response for one atomic coordinate or one electric field coordinate). Thus, the core routines are called within a perturbation loop in a single CPSCF iteration. The CPSCF infrastructure has been developed to accept any combination of atomic perturbations or electric field displacements. Thus it is possible to run a CPSCF cycle for all atomic coordinates or a

single atomic coordinate. This affords flexibility to the developer. This was also done to make the code compatible with the legacy-style code where there were 'reduced memory' and 'full memory' options, however, we found the computational overhead associated with calculating the perturbations in serial (i.e in multiple CPSCF cycles) was negligible compared to the 'full memory' run. To reiterate, the only difference is having the perturbation loop inside the CPSCF cycle or outside. Thus we have set the default to calculate the perturbations in serial. All major subroutines and their variations are given in Tables 8 to 12, where `red_text` represents subroutines that still belong to the legacy infrastructure, and are yet to be refactored and integrated in the new interface. This applies to the evaluation of the first order potential routines, which cannot be easily amalgamated and so will be moved directly in the future.

3 Objective 2: Improve scalability & integration with ELSI

DFT calculations within FHI-Aims feature several different parallelisation strategies. Integrations over potentials, densities or wavefunctions are performed on overlapping radial grids and the integration grid points are distributed over compute cores. Furthermore, Hamiltonian and overlap matrices constructed at different **k**-points are trivially distributed and solved independently on different CPUs to achieve efficient Brillouin zone sampling for periodic systems. These strategies lead to effective scalability in the regime where the number of cores $N_{\mathrm{CPU}} < N_{\mathbf{k}}$. As the number of compute cores exceeds the number of **k**-points, further scalability can only be achieved if matrix and vector quantities are distributed over cores and matrix operations are distributed. In FHI-Aims, this is achieved via BLACS distribution of matrices and parallel matrix algebra is outsourced to the ELPA/ELSI libraries[9, 11]. During refactoring, we have enabled distributed matrix algebra for the friction response calculations and have cleaned and debugged other response modes. Furthermore, the ELSI parallel matrix I/O module has been integrated for matrix restart functionality.

Capabilities

A rough guide of maximum system sizes that can be calculated with the new infrastructure for each runmode is given in Table 2, where calculations have been carried out on ARCHER2. This is indicative of the memory usage for each runmode. In principal, much larger system sizes can be investigated by underloading each node (to increase the maximum memory available to each task) but we employ the maximum number of 128 cores for each node in the calculations with the maximum 2000 MB memory per core for the standard queue.

Table 2: Rough system sizes achievable for each runmode with DFPT. 4 ARCHER2 nodes, 2000 MB per core, no underloading of nodes. LDA, light basis set. Aperiodic test system was polyethane, whilst periodic test system was bulk Si. For the latter, a $4 \times 4 \times 4$ **k**-grid is used. For electronic friction calculations we calculate the response for one atom only. * ScaLAPACK calculation failed for original dielectric infrastructure due to bugs in the legacy code. † refers to functionality that whilst is implemented, currently is not fully functional, but fixing these would just be a matter of checking that the array indices are correct. Supercell functionality currently requires underloading nodes or high memory nodes.

Response	Periodicity	ScaLAPACK?		Max system size, x	
		Legacy	New	Legacy	New
Atomic	0	No	No	$218 < x < 338$	$218 < x < 338$
Atomic	3	Yes	No†	$54 < x < 128$	$54 < x < 128$
Atomic	Supercell	Yes	Yes	0	0
Electric	0	Some	No	$338 < x < 674$	$338 < x < 674$
Electric	3	Yes*	No†	$128 < x < 250$	$128 < x < 250$
Friction	0	N/A	No	N/A	$218 < x < 338$
Friction	3	N/A	Yes	N/A	$128 < x < 250$

Scaling behaviour on ARCHER2

In this section we show the scaling behaviour for calculation of different response properties, with the calculations performed on ARCHER2. We first show scaling for a response property where there has been no significant improvement to the scaling, and then another where the scaling is significantly improved. A single CPSCF iteration takes a similar amount of time as a standard SCF iteration within FHI-Aims. This can be seen in Figure 4 which compares the average time per iteration for SCF and CPSCF using the atomic response functionality, without ScaLAPACK distribution, for a range of polyethylene polymer sizes. Performance of the new interface and the legacy code does not significantly differ for this runmode.

The CPSCF cycle shows favourable scaling on ARCHER2, in Figure 5 we show the scaling behaviour with increasing MPI tasks; the total time per iteration has a slope close to -1 when fit to a power series. The integration of $\boldsymbol{H}^{(1)}$ takes the most time per iteration, but may be reduced in future by implementation of localised dense matrices (rather than global sparse matrices), as is done for the ground state SCF in FHI-Aims (see future work section). The evaluation of the $V^{(1)}(\boldsymbol{r})$ has the worst scaling with a slope of -0.66, however the prefactor is very small so this is not expected to dominate in any feasible working range. A comparison

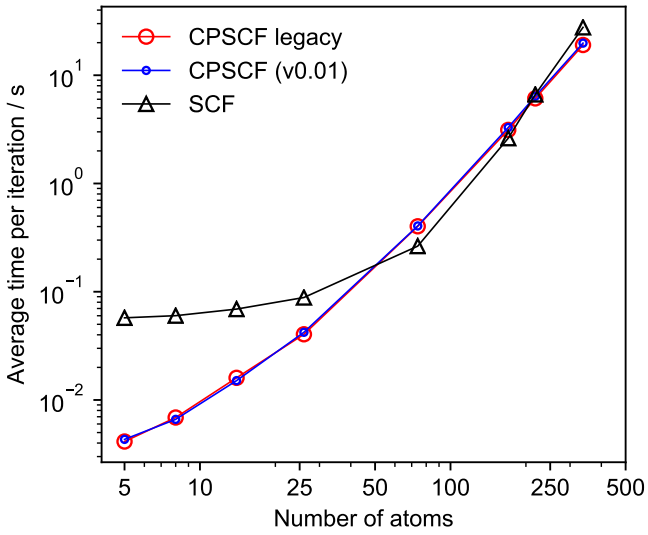

Figure 4: Average time per iteration for CPSCF (v0.01), CPSCF (legacy code) for the response to a single atomic perturbation compared to a ground state SCF calculation in FHI-Aims (version 210513, GNU compilation), for different numbers of atoms for a carbon chain starting from methane, building up to polyethylene chains. Structures are generated using Open Babel from SMILES code and have not been optimised. Calculations carried out without ScaLAPACK distribution. Light basis and LDA functional was employed. Larger system sizes cannot be achieved without underloading nodes. 4 ARCHER2 nodes used.

of the scaling for the total time per iteration for the new code and legacy code is shown in Figure 6. The new code shows significantly improved scaling, due to fixing the ScaLAPACK implementation, which was otherwise not functioning with the compiled code on ARCHER2.

New restart functionality

It is now possible to restart CPSCF from a first order density matrix file, and output said file. This functionality should be supported for each response and runmode except for i) supercells and ii) periodic systems+no ScaLAPACK+`real_eigenvectors`. The restart functionality uses parallel matrix I/O implemented in ELSI and the filetypes are '.csc'. The number and name of files depend on the periodicity, the number of spin channels, and the response type. The files store metadata on the system and calculation, allowing the restart files to be used for calculations with a different number of MPI tasks. A python parser for ELSI matrix files is conveniently included within the FHI-Aims code, this will allow users to post-process the first

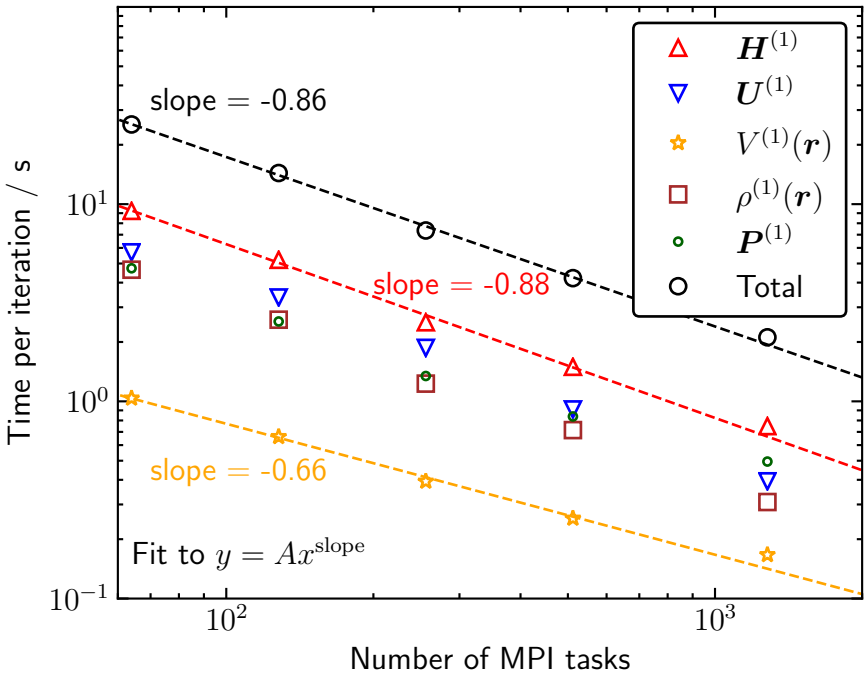

Figure 5: Time per CPSCF iteration for evaluation of key matrices as a function of the number of MPI tasks (cores) used. This is calculated for electric field response of bulk Si with 128 atoms on ARCHER2 with the new interface. The evaluation of $\boldsymbol{H}^{(1)}$ is the most costly for all number of MPI tasks investigated. A fit to power series is also shown for the total time per iteration and for the $\boldsymbol{H}^{(1)}$ and $V^{(1)}(\boldsymbol{r})$ evaluations, we find favourable scaling for all (including the total time) except for the $V^{(1)}(\boldsymbol{r})$ evaluation, however the prefactor for this is very small

order density matrix, and in future, all response matrices.

4 Objective 3: Support for GGA and beyond

FHI-Aims standard SCF routines are interfaced with the LibXC library, [10] which provides access to a wide range of exchange-correlation functionals. The previous DFPT module employed a small set of hardcoded functionals. We implemented a LibXC interface to the DFPT subroutines that depend on exchange correlation information. The new DFPT module supports the use of LDA, GGA, and hybrid functionals for calculation of the response matrices during CPSCF. However, the driver routines have mixed support for GGA and hybrids and this is documented in Table 3.

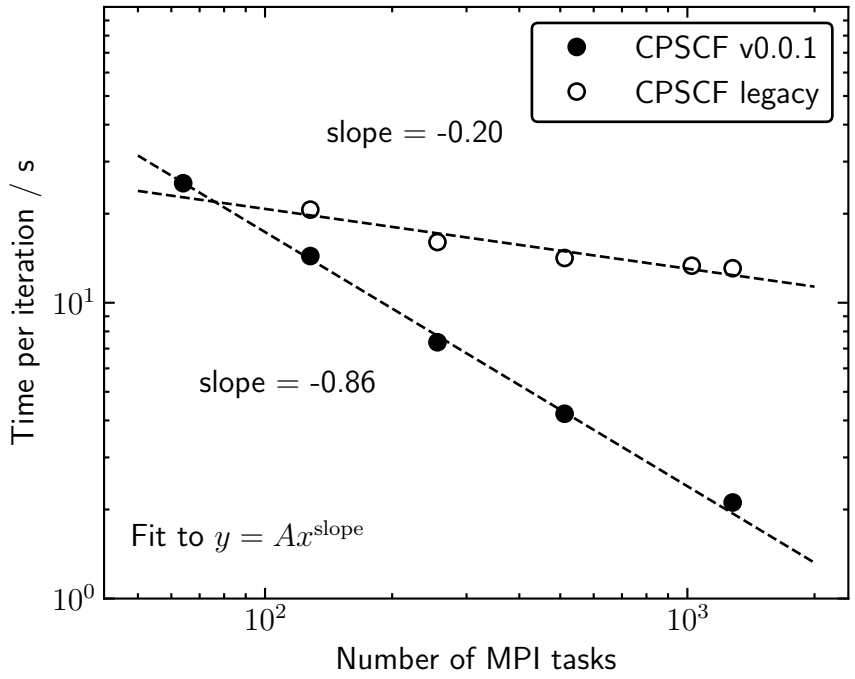

Figure 6: Comparison of total time per CPSCF iteration for legacy code and new interface (v0.0.1) for electric field response of bulk Si with 128 atoms on ARCHER2. In the legacy code, the ScaLAPACK implementation was not functioning on ARCHER2, this was fixed in the new interface as a result of this work.

Table 3: Current support for GGA and hybrid functionals

Property	GGA & hybrid support
CPSCF	Supported
Friction tensor	Supported
Hessian	LDA only
Dynamical matrix	LDA only
Polarizability	Supported
Dielectric constant	LDA & GGA only

5 Objective 4: New driver modules for electron-phonon coupling and electronic friction

Electron-phonon-coupling (EPC)-induced vibrational linewidths and electronic friction properties for molecules at metal surfaces were already implemented based on finite difference

response calculations. [12] As part of this project, we have refactored the electronic friction implementation as a driver to be coupled with the new DFPT framework. This has led to a significant performance increase and reduction in memory usage. The scaling behaviour for the finite difference implementation is shown in Figure 7, for two system sizes, the larger of which was not possible to be calculated in the original framework due to the lack of efficient memory distribution. Whilst the evaluation of the friction tensor may have an unfavourable scaling, it is only performed once per calculation and has a relatively low prefactor (comparable to a single SCF step). Whilst there is still an issue with metallic systems when using DFPT, (discussed below) the improvements to the friction driver also apply to the DFPT runmode.

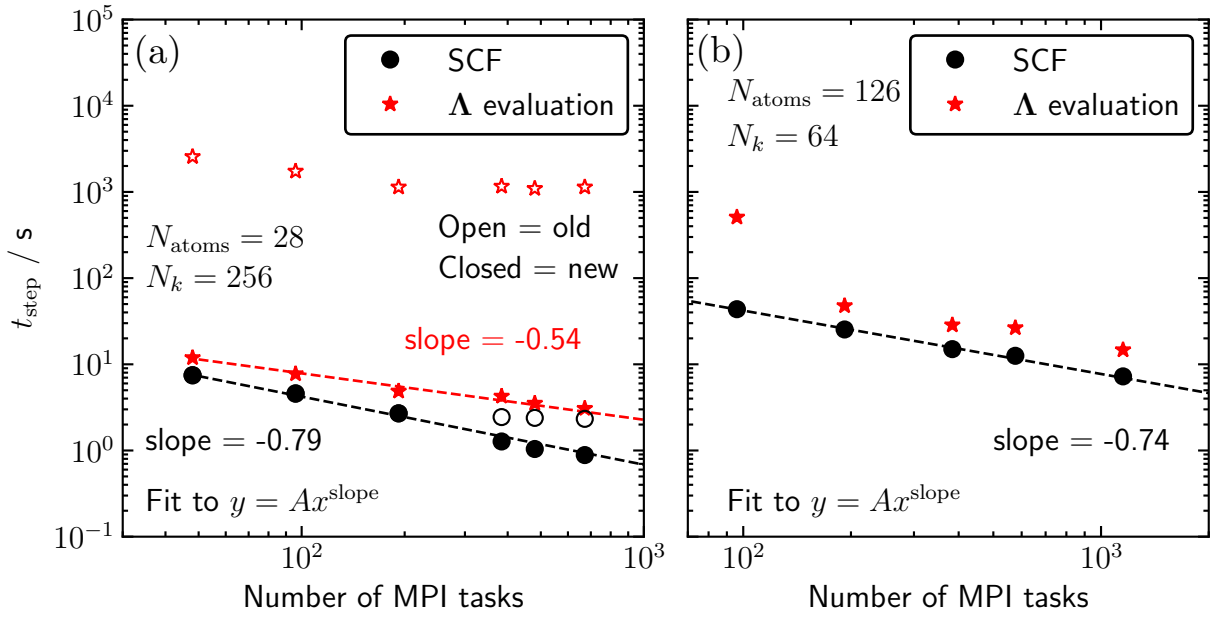

Figure 7: Scaling behaviour for the friction driver using finite difference for CO/Cu(100). We compare each SCF step and the friction tensor ($\mathbf{\Lambda}$ evaluation, one per calculation). In (a) we compare with the older code (190506, open symbols) with the current refactored code (210928, closed symbols). In (b) we use a larger system, which was not possible with the original code. Calculations performed on University of Warwick Avon HPC system (Dell PowerEdge C6420 compute nodes each with 2 x Intel Xeon Platinum 8268 2.9 GHz 24-core processors; 48 cores per node; 180 nodes).

6 Conclusion

In this project, we have performed a substantial rewrite of a large and fractured functionality in the FHI-Aims software package, namely the DFPT-based calculation of linear response properties. The new code has clear specifications and offers transparent functionality for various response properties. We were able to port most functionality from the previous DFPT code into the new infrastructure. Some previous runmodes were not functional in the old code and have been repaired. Some functionality is yet to be enabled in the new iterface, but will be corrected in the coming months. The current working functionality is detailed in Table 4. Overall, the new interface is far more maintainable and understandable, with a significant reduction in code duplication. The scaling behaviour in areas has significantly improved due to fixing the parallelism implementations. A huge improvement to the electronic friction / electron-phonon coupling driver has been accomplished, including improved scaling, memory usage, and implementation of DFPT to calculate first order response matrices using the new interface.

The new code framework has been disseminated via the FHI-Aims Slack channel and the functionality and merge request were discussed in detail on the FHI-Aims GitLab. While a number of outstanding issues for future work remain (see related section below), this project has provided an important starting point that will enable the developer community to address these issues in the future while retaining robust functionality for most runmodes and compatibility with most recent code innovations in FHI-Aims.

7 Future work

There are several existing and outstanding issues in the code that are described in Table 5. We propose possible solutions and detail whether the user is warned about these issues when running the calculations (code stop or warning message). We hope to be able to address these in the future or to support other developers to address them.

Random phase approximation (RPA) driver

The current `rpa_force` code used parts of the legacy DFPT framework to evaluate certain force components. It should be relatively easy to transfer this application module to the DFPT interface instead. The driver would need to call the `first_order_overlap` module in a similar way to the `atomic_response` or `friction` codes. This would require little coding effort and would ensure RPA benefits from any future developments in the new DFPT framework.

Table 4: Current working functionality with new DFPT interface. Periodicity refers to aperiodic cluster calculation (0), periodic boundary conditions (3), or real-space supercell construction of the first order response matrices (Supercell) as proposed by Shang et al.[5] Distribution refers to the parallelisation strategy, where "ScaLAPACK" refers to distributed matrix algebra.

Response	Periodicity	Distribution	State of functionality
Atomic	0	serial	fully functional
Atomic	0	ScaLAPACK	Full dense matrices used for integration. No $S^{(1)}$ ScaLAPACK distribution. Further scaling improvements possible.
Atomic	3	serial	fully functional
Atomic	3	ScaLAPACK	not functional
Atomic	Supercell	serial	not functional
Atomic	Supercell	ScaLAPACK	not functional
Electric	0	serial	fully functional
Electric	0	ScaLAPACK	Full dense matrices used for integration
Electric	3	Serial	fully functional
Electric	3	ScaLAPACK	not functional
Friction	0	Serial	fully functional
Friction	0	ScaLAPACK	Full dense matrices used for integration. No $S^{(1)}$ ScaLAPACK distribution.
Friction	3	Serial	fully functional
Friction	3	ScaLAPACK	fully functional
Friction	Supercell	Serial	Not implemented
Friction	Supercell	ScaLAPACK	Not implemented

Supercell approach

For electronic friction in particular, we are (usually) only interested in the response due to certain atoms not all atoms. Thus, it would be hugely beneficial in terms of memory and wall-time use to restrict the calculation to certain atoms and their periodic images in the supercell approach. All aperiodic and periodic non-supercell approaches already support restricting the response to certain atoms. Currently the electronic friction module is not compatible with the supercell approach. Currently, the supercell approach does not support spin collinear calculations, `use_local_index` or `collect_eigenvectors .false.`. The latter two options might

Table 5: Outstanding issues in the code, whether the user is warned at runtime and possible solutions

Issue	Description	User warned?	Solution
Fractional occupations	Eigenvalue difference goes to zero in denominator in $\boldsymbol{U}^{(1)}$. A smearing correction was proposed below, but was not implemented correctly.	Yes	Re-implementation of the numerically stable smearing fix.
`spin collinear`	parts of code don't support spin collinear treatment	Yes	further extension to spin collinear treatment is still under way (90% done)
Periodic atomic response	Lacks support for `real_eigenvectors`	Yes	Can be addressed with a minor extension.
Inconsistent variable naming	`n_basis` and `n_states` are mixed in places	No	Requires further code maintenance
`use_local_index`	Not supported for the construction of BLACS distributed matrices	Yes	A simple extension is required for construction of BLACS matrices. Unclear what is required for integration routines, if anything.

help tackle the significant memory usage of this runmode, due to the large supercell size. A significantly slow step is solving the eigenvalue problem for the supercell, this might be interfaced to ELSI so that the optimum eigensolver is chosen and so the runmode benefits from any future development on eigensolvers as part of the ELSI project.

Real space matrix use for aperiodic systems

Currently, when performing integrations for aperiodic systems, the global dense matrix is used which is a memory bottleneck. It would be better to use the global real space matrix (or the local dense real space matrix).

Implementing localised real-space matrices

This would enable use of the keyword `use_local_index` for reduced memory usage for very large systems. All ScaLAPACK matrix constructions are based on `construct_hamiltonian_scalapack`. To enable the use of `use_local_index`, sparse matrix constructions need to be modified (following the template of `set_sparse_local_ham_scalapack`).

Extension of hybrid functionals to driver routines

Though the CPSCF module is capable of calculating response matrices with hybrid functionals, the driver routines have limited hybrid support. Currently Hessian integration is limited to LDA only. Electric field response calculations are limited `RI_method V` (resolution of identity) whereas `RI_method lvl` is the safe default.

Output of all response matrices using ELSI

Currently the first order density matrix can be output using ELSI, as well as the first order Hamiltonian and overlap if using the friction code. We plan to include support for all response matrices, which will allow convenient post-processing. This is a relatively simple extension of the existing functionality.

Acknowledgements

This work was funded under the embedded CSE programme of the ARCHER2 UK National Supercomputing Service (http://www.archer2.ac.uk). We also wish to thank Dr Honghui Shang for helpful discussions.

References

(1) Baroni, S.; de Gironcoli, S.; Dal Corso, A. *Rev. Mod. Phys.* **2001**, *73*, 515–562.

(2) Giustino, F. *Rev. Mod. Phys.* **2017**, *89*, 015003.

(3) Shang, H.; Raimbault, N.; Rinke, P.; Scheffler, M.; Rossi, M.; Carbogno, C. *New J. Phys.* **2018**, *20*, 073040.

(4) Blum, V.; Gehrke, R.; Hanke, F.; Havu, P.; Havu, V.; Ren, X.; Reuter, K.; Scheffler, M. *Comp. Phys. Commun.*, *180*, 2175–2196.

(5) Shang, H.; Carbogno, C.; Rinke, P.; Scheffler, M. *Comp. Phys. Commun.* **2017**, *215*, 26–46.

(6) Giustino, F.; Cohen, M. L.; Louie, S. G. *Phys. Rev. B* **2007**, *76*, 165108.

(7) Agapito, L. A.; Bernardi, M. *Phys. Rev. B* **2018**, *97*, 235146.

(8) FHI-Aims, `https://aims-git.rz-berlin.mpg.de/aims/FHIaims`.

(9) Yu, V. W. z.; Corsetti, F.; García, A.; Huhn, W. P.; Jacquelin, M.; Jia, W.; Lange, B.; Lin, L.; Lu, J.; Mi, W.; Seifitokaldani, A.; Vazquez-Mayagoitia, A.; Yang, C.; Yang, H.; Blum, V. *Comp. Phys. Commun.*, *222*, 267–285.

(10) Marques, M. A.; Oliveira, M. J.; Burnus, T. *Comp. Phys. Commun.* **2012**, *183*, 2272–2281.

(11) Marek, A.; Blum, V.; Johanni, R.; Havu, V.; Lang, B.; Auckenthaler, T.; Heinecke, A.; Bungartz, H.-J.; Lederer, H. *J. Phys.: Condens. Matter* **2014**, *26*, 213201.

(12) Maurer, R. J.; Askerka, M.; Batista, V. S.; Tully, J. C. *Phys. Rev. B* **2016**, *94*, 115432.

Appendix

Key variables and arrays for developers

Commonly used variables in the DFPT module that control the CPSCF cycle and evaluation of the response matrices are shown in Table 6 in the Appendix. Of particular importance are `nuclear_response` and `dfpt_perturbation_list` which should be set in the driver that is calling this module. The array `dfpt_homo_lumo_states` enables the manual selection of limits of lowest occupied and highest unoccupied state between which a sum over states will be calculated. I stands for integer, F for real float, and L for logical

Table 6: Important variables in the new DFPT module

Variable	Dimension	Description	Range
nuclear_response	L	Whether we are calculating response to Atomic (True) or Efield perturbations (False)	True/False
n_dfpt_coords	I	Number of Cartesian directions per atom or for Efield in this CPSCF cycle	1 – 3
n_dfpt_atoms	I	Number of atomic perturbations in this CPSCF cycle (0 for Efield)	0 – n_atoms
atom_dim	I	Length of array dimension for atomic perturbations. If Efield then set to 1	1 – n_atoms
dfpt_perturbation_list	I(n_atoms,3)	Map of perturbations being calculated this CPSCF cycle. 1 if calculated. For Efield we only read row 1	0/1
dfpt_homo_lumo_states	I(n_spin,2,n_k_points)	Index of HOMO (1) and LUMO (2) state for each spin channel and **k**-point	1 – n_states
dfpt_supercell	L	Whether to use supercell approach for Atomic perturbations	True/False
change_of_first_order_DM	F	Change in $\boldsymbol{P}^{(1)}$ between successive CPSCF iterations	-

Table 7: Important array dimensions in different runmodes

System	Matrix	Dimensions
All	$\boldsymbol{\rho}^{(1)}$,$\boldsymbol{V}^{(1)}$	n_full_points
Aperiodic (LAPACK-style)	$\boldsymbol{P}^{(1)}$,$\boldsymbol{H}^{(1)}$,$\boldsymbol{U}^{(1)}$	n_basis,n_basis
Aperiodic (ScaLAPACK-style)	$\boldsymbol{P}^{(1)}$,$\boldsymbol{H}^{(1)}$,$\boldsymbol{U}^{(1)}$	mxld,mxld
Periodic	$\boldsymbol{P}^{(1)}$	n_hamiltonian_matrix_size
Periodic (LAPACK-style)	$\boldsymbol{H}^{(1)}$,$\boldsymbol{U}^{(1)}$	n_basis,n_basis,n_k_points_task
Periodic (ScaLAPACK-style)	$\boldsymbol{H}^{(1)}$,$\boldsymbol{U}^{(1)}$	mxld,mxcol
Supercell	$\boldsymbol{P}^{(1)}$,$\boldsymbol{H}^{(1)}$	n_hamiltonian_matrix_size
Supercell (LAPACK-style)	$\boldsymbol{U}^{(1)}$	n_basis_sc_DFPT,n_basis_sc_DFPT
Supercell (ScaLAPACK-style)	$\boldsymbol{U}^{(1)}$	mxld_sc_DFPT,mxcol_sc_DFPT

Table 8: Different routines to evaluate first order density matrix, $\boldsymbol{P}^{(1)}$ for different runmodes

Name	Input	Output	ScaLAPACK	Cluster	Periodic	Supercell	Collinear
evaluate_DM1	Dense	Dense	No	Yes	No	No	Yes
_scalapack	Dense	Dense	Yes	Yes	Yes	No	Yes
_sparse	Dense	Sparse	No	No	Yes	No	Yes
_sparse_supercell	Sparse	Sparse	No	No	(Yes)	Yes	No
_scalapack_supercell	Dense	Dense	Yes	No	(Yes)	Yes	No
_electric	Dense	Dense	No	Yes	No	No	Yes
_electric_sparse	Dense	Sparse	No	No	Yes	No	Yes

Table 9: Different routines for the integration of first order electron density, $\rho^{(1)}(\boldsymbol{r})$. None of the routines are ScaLAPACK-enabled.

Name	Input	Output	Cluster	Periodic	Supercell	Collinear
integrate_rho1	Dense	n_full_points	Yes	No	No	No
_sparse	Sparse	n_full_points	No	Yes	No	No
_sparse_supercell	Sparse	n_full_points	No	(Yes)	Yes	No
_electric	Dense	n_full_points	Yes	No	No	Yes
_electric_sparse	Sparse	n_full_points	No	Yes	No	No

Table 10: Different routines for evaluation of first order potential, $V^{(1)}(\boldsymbol{r})$.

Name	Input & Output	Cluster	Periodic	Supercell	Collinear
sum_up_whole_potential_p2_shanghui	n_full_points	Yes	No	No	Yes
_phonon_reduce_memory	n_full_points	No	Yes	No	Yes
_p1	n_full_points	No	(Yes)	Yes	Yes
_dielectric	n_full_points	No	Yes	No	Yes

Table 11: Different routines for integration of first order Hamiltonian, $\boldsymbol{H}^{(1)}$.

Name	Input	Output	ScaLAPACK	Cluster	Periodic	Supercell	Collinear
integrate_H1	Dense	Dense	No	Yes	No	No	No
_sparse	Sparse	Sparse	No	No	Yes	No	No
_sparse_supercell	Sparse	Sparse	No	No	(Yes)	Yes	No
_electric	Dense	Dense	No	Yes	No	No	Yes
_electric_sparse	Sparse	Sparse	No	No	Yes	No	No

Table 12: Different routines for evaluation of first order coefficient matrix, $\boldsymbol{U}^{(1)}$.

Name	Input	Output	ScaLAPACK	Cluster	Periodic	Supercell	Collinear
evaluate_U1	Dense	Dense	No	Yes	No	No	Yes
_scalapack	Dense	Dense	Yes	Yes	Yes	No	Yes
_sparse	Dense	Sparse	No	No	Yes	No	Yes
_electric	Dense	Dense	No	Yes	No	No	Yes
_electric_sparse	Dense	Sparse	No	No	Yes	No	Yes