ARCHER2-eCSE04-3: Relativistic all-electron orbital-constrained Density Functional Theory to simulate x-ray photoemission and absorption spectroscopy

Samuel J. Hall, Dylan B. Morgan, and Reinhard J. Maurer

Department of Chemistry, University of Warwick, Gibbet Hill Road, CV4 7AL Coventry, UK

11.08.2023

Abstract

Core-hole calculations simulating x-ray photoelectron (XPS) or x-ray absorption spectra (XAS) are important to interpret complex experimental spectra. Density functional theory (DFT) is one method that can be used to efficiently and accurately simulate core-level spectra. The DFT code FHI-aims already has this functionality and has the added benefit of being able to produce absolute XPS binding energies, thanks to its all-electron numeric atomic-orbital basis approach. However, the code for these calculations was outdated and inefficient, not utilizing newer algorithms and software libraries. This project aims to bring the code up-to-date, improving the scaling and efficiency and restructuring and simplifying the code to make it more maintainable for future development. The new code achieves a significant improvement in the scaling of core-level simulations, on the same scale as standard single-point calculations, and improved the structure and readability of the code by removing almost 3000 lines of code and including new documentation and tutorials.

Keywords:

ARCHER2, eCSE, Density Functional Theory, core-level spectroscopy, x-ray photoemission, x-ray absorption, Delta Self Consistent Field

1 Project Description

Simulation of x-ray photoemission and x-ray absorption spectroscopy can be a valued tool in predicting and elucidating experimental spectra. There are many different methods of simulating spectra from density functional theory (DFT), GW and coupled cluster (CC), with a variety of software codes. Whilst methods such as GW and CC can provide highly accurate absolute binding energy simulations for small molecular systems, the computational cost to perform such simulations of larger and more complex molecules and materials is much greater. DFT methods such as Delta-Self-Consistent-Field (ΔSCF) and the transition potential (TP) method can be used to accurately predict core-level spectra for these larger systems at a much more manageable computational cost. Commonly, these calculations are performed in electronic structure codes that employ pseudopotential basis sets, which means that absolute XPS binding energies are not accessible.

FHI-aims [1] is an all-electron, full-potential electronic structure package that utilises a numeric atomic-orbital basis approach allowing for accurate calculation of total energies and properties. The explicit treatment of core and valence electrons means absolute XPS binding energies and precise treatment of relativistic effects such as spin-orbit coupling are possible. FHI-aims has an implementation of the self-consistent-field algorithm that allows for non-Aufbau principle occupations via efficient orbital-constrained core hole simulation methods such as ΔSCF and TP. These methods can produce spectra on an absolute energy scale, enabling the prediction of XP and XA spectral signatures in various materials. This method has been previously used to simulate spectra of large and complex systems such as large transition-metal complexes, bulk metals and organic/inorganic interfaces [2–4].

The downfall with performing these calculations in the FHI-aims code is that the desired performance of these calculations can be inconsistent [4], with the user-facing multiple challenges to achieve the correct result such as SCF convergence issues, limited memory, and errors in core hole localisation. The reason for these issues stems from the fact that the code (initially developed in 2008) has grown organically and not kept up with recent developments. It employs a simplistic approach to constrain core holes and does not utilise recent innovations. One of these recent innovations is the electronic structure library ELSI, which contains highly efficient algorithms and links to the ELPA library for massively scalable parallel matrix algebra. This library is used by other codes, such as DFTB+ and SIESTA and can be optimally tuned to allow for ideal HPC scaling beyond 10,000 atoms.

This project aims to overhaul the current code implementation by isolating, refactoring and augmenting the existing code into a single Fortran module. The current code is spread across various modules and with unnecessary duplicated code, with little to no documentation. In its current state, it is hard for a new user to study and update the code with new functionality. By creating a new single module to contain all core hole constraining code, with appropriate documentation, will allow for a significant reduction in the length of code needed and easier future development. Linking the code with the main SCF routines in ELSI will allow calculations to utilise parallel distributed eigensolvers to improve scalability.

To achieve this the project has been split up into four main objectives:

1. Redesign of orbital-constrained DFT code to establish consistent functionality and performance for aperiodic and periodic core-level simulations. [completed]

2. Improving scalability and transferability of functionality by integration with (pre-)exascale electronic structure interface ELSI. [completed]

3. Developing improved core hole localisation constraints to enable the simulation of core hole excitation from arbitrary electronic core states. [started as part of the project]

4. Merge with spin-orbit functionality for the simulation of spin-resolved core-level absorption and x-ray dichroism in relativistic materials. [postponed]

2 Summary of Work

The first step in the process was to understand the current code implementation for performing a core-hole-constrained calculation. As previously mentioned, this code was strewn across multiple modules and mixed in with other functionality which made gaining a full overview of how the code works in order to start developing and improving it a difficult and time-consuming task. The task of surveying the code was much harder than initially anticipated. A flow chart summarising the logic route taking place at the start of the project is shown in Figure 1. The flowchart shows the main route for the two main keywords that FHI-aims can use, *force_occupation_projector* (FOP) and *force_occupation_basis* (FOB). For a standard FHI-aims calculation, the code would move over into the `elsi_wrapper.f90` module (grey box labeled with *), in order to calculate the chemical shift and occupations, whereas for an FOP calculation the code would move into the `force_occupation.f90` module (orange, left in Figure 1) and follow internal written subroutines, which followed the legacy approach that was standard before the introduction of

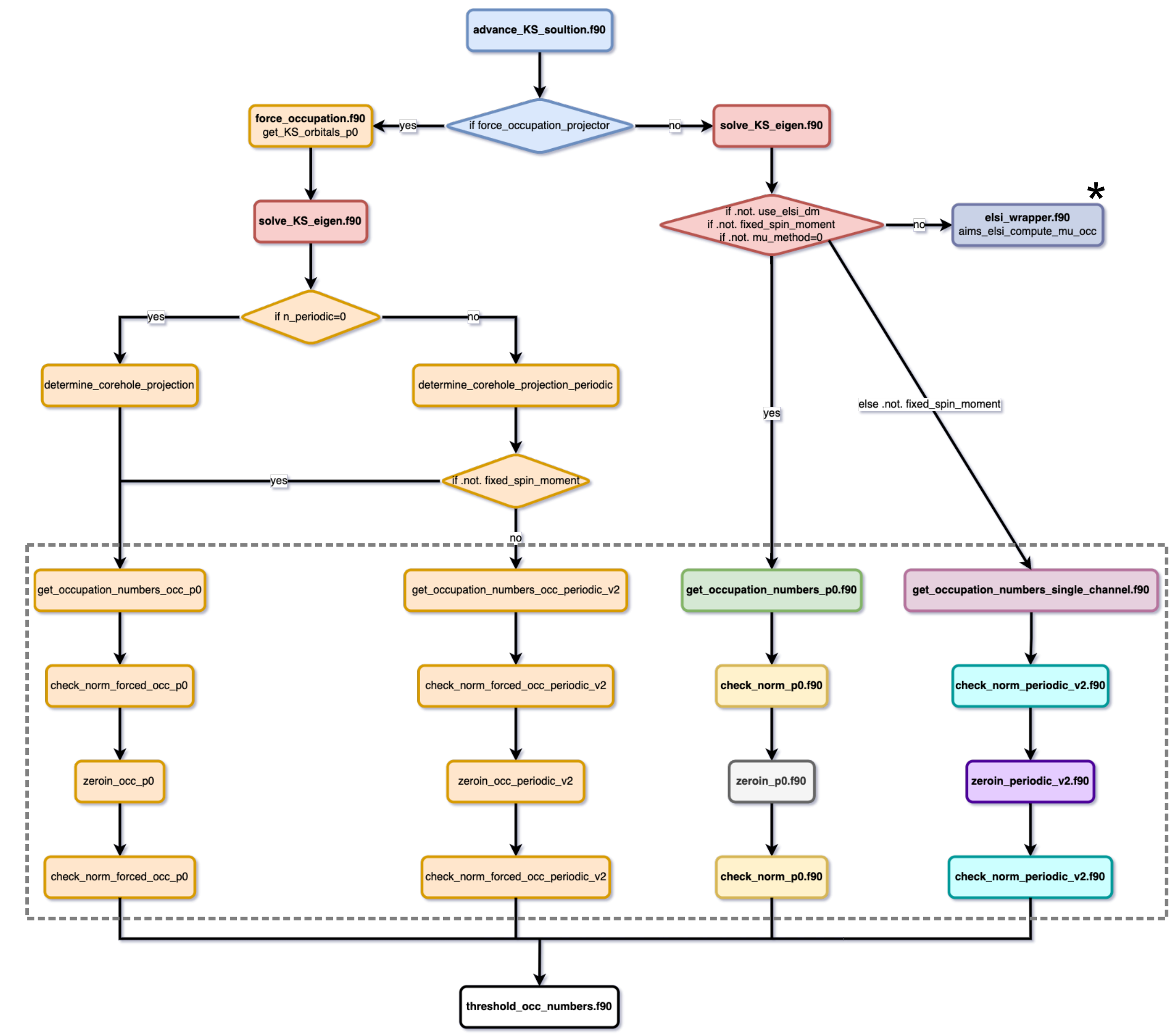
advance_KS_soultion.f90
if force_occupation_projector
yes
force_occupation.f90
get_KS_orbitals_p0
no
solve_KS_eigen.f90
solve_KS_eigen.f90
if .not. use_elsi_dm
if .not. fixed_spin_moment
if .not. mu_method=0
no
elsi_wrapper.f90
aims_elsi_compute_mu_occ
*
if n_periodic=0
yes
no
determine_corehole_projection
determine_corehole_projection_periodic
yes
else .not. fixed_spin_moment
if .not. fixed_spin_moment
yes
no
get_occupation_numbers_occ_p0
get_occupation_numbers_occ_periodic_v2
get_occupation_numbers_p0.f90
get_occupation_numbers_single_channel.f90
check_norm_forced_occ_p0
check_norm_forced_occ_periodic_v2
check_norm_p0.f90
check_norm_periodic_v2.f90
zeroin_occ_p0
zeroin_occ_periodic_v2
zeroin_p0.f90
zeroin_periodic_v2.f90
check_norm_forced_occ_p0
check_norm_forced_occ_periodic_v2
check_norm_p0.f90
check_norm_periodic_v2.f90
threshold_occ_numbers.f90

Figure 1: Flowchart depicting the code logic when running a core hole constrained calculation in the FHI-aims code before this project. Each colour represents a different Fortran module in the code.

ELSI. For an FOB calculation, the situation was even more difficult with subroutines spread out over multiple modules and files (various colours, right in Figure 1). This all made for a very convoluted and messy route for the code to take and for developers to come in and work on.

Out of the four objectives outlined in the plan of this project, only the first two have been achieved. This was mainly due to the steep learning curve associated with getting into a position to properly tackle the planned objectives. From the code being strewn across multiple places and embedded at various points throughout the code, to the complete lack of documentation giving any indication of the purpose or instruction of how the code works.

2.1 Objective 1: Moving functionality into ELSI

The first objective was to be achieved by updating the occupation and chemical shift calculation process to be in line with the default method used in FHI-aims. By moving the non-Aufbau principle occupation code required for the core-level constrained calculations into the ELSI library, the subroutine can be used by any code that uses ELSI and we reduce the FHI-aims-specific routines and allow for a simpler process for both aperiodic and periodic simulations. In the old code structure, there were a lot of subroutines that were dedicated to performing the same task as the ELSI library but all had separate subroutines depending on the system and type of calculation that was performed. This can be seen in Figure 1, where all subroutines included in the grey dashed box are responsible for the same things under different specific run modes (as can be seen by the similar names of the routines). The same can be achieved within ELSI through `elsi_wrapper.f90`. Therefore all this code was made redundant and removed.

In order to transfer over to ELSI to handle the occupation and chemical shift, we had to write a new subroutine within ELSI that would contain the properties of constraints that had been chosen, such as the number of constraints, which KS state and spin state the new occupation would be in and the new electron occupation. This was achieved by creating a set of new property arrays that ELSI would use to store all this information. Each array has a length of the number of constraints and three different arrays for the KS state (`constr_state`), the spin channel (`constr_spin`), and electron occupation (`constr_occ`). A logical flag was created, `flag_occ_non_aufbau` which is used to tell ELSI whether a non-aufbau occupation is requested and for the code to apply these defined occupations when this is the case. These variables are all that the library needs, which are passed over from FHI-aims (and can be implemented for any other code that uses the ELSI library) and if done so will perform the occupation and chemical shift with the defined constraints.

To ensure that this functionality remains in working condition, a new set of regression tests checking the occupations across a multitude of cases such as a single constraint, multiple constraints, and across a k-grid have been added to the integration tests.

2.2 Objective 2: Improving scalability of core hole calculations

Once ELSI had been set up to be able to receive the new occupation constraints and efficiently calculate the chemical potential and non-Aufbau occupations, the next step and second objective was to integrate and restructure the FHI-aims code to utilise this new ELSI occupation functionality. This would improve the scalability and performance of the code, by allowing matrix algebra operations to be properly distributed across all cores and to reduce memory bottlenecks. This also allowed us to completely redesign and restructure the core hole constraining code, removing all the code that now would be performed by ELSI and replacing all the code that was previously jumbled around in various places and sort it all into one new clearly structured and documented module.

A new module called `delta_scf.f90` was created which would hold all the subroutines that are needed to perform a core hole-constrained calculation. We have laid out this module with a clear and concise format to help any future developers to understand its contents. At the top of the module is a breakdown of all the global and local subroutines (in order of appearance) and the global variables that are defined in this module. At the beginning of each subroutine is a brief description of the purpose of the code. A single entry point to the module has been created through the use of a single logical variable `start_deltascf` which should be used whenever a core hole calculation is performed and is used to enter the main subroutine, `deltascf_main`, where from here a case select is used to determine which direction the code is to go depending on what keywords or calculation you are performing. Currently, there are two ported-over keywords from before, renamed to `deltascf_projector` and `deltascf_basis`. After this, the code then determines whether the calculation is a periodic or aperiodic calculation, performs the necessary routines, and then passes over to `elsi_wrapper.f90` to calculate the occupation and chemical shift like any other FHI-aims calculation. A new flowchart of this logic is shown in Figure 2. When compared to the old code flowchart in Figure 1, a much simpler and shorter design of the code has been achieved.

Restructuring and rewriting the code allowed for opportunities to improve the performance and readability of parts of the code. The code was severely lacking in any documentation describing what the code was doing and parts relied on multiple series of loops to perform matrix multiplications which were very hard to follow and understand. This was seen for the projector keyword when performing the maximum overlap method (MOM). [5] This checks whether the core hole has moved in between SCF steps by checking where the maximum overlap is between steps and changes the occupation constraints accordingly if the hole has moved. The method behind this works by calculating an orbital overlap matrix, $\boldsymbol{O}$ from the eigenvectors of the previous step $\boldsymbol{C}^{\text{old}}$ and current step $\boldsymbol{C}^{\text{new}}$ and the overlap matrix $\boldsymbol{S}$

$$\boldsymbol{O} = (\boldsymbol{C}^{\text{old}})^{\top} \cdot \boldsymbol{S} \cdot \boldsymbol{C}^{\text{new}} \tag{1}$$

In the old code, this was calculated using a series of loops to pass over the matrices and calculate the specific overlap of each particular state and store the overlap variable, overriding this if the next one was larger. Whilst this in essence worked, it was a rather inefficient method and was hard to understand. As this was in fact just simple matrix multiplication, this could easily be replaced with simple functions contained within LAPACK, which calculate the whole overlap matrix much more efficiently in one operation.

The previous basis procedures were convoluted and messy, with a lot of redundant code spread over many files with inconsistent nomenclature. The refactoring of the code was heavily motivated by simply streamlining this, and it was redesigned with the philosophy of implementing it in an intelligible and extensible manner. Much of the redundant code included unnecessary loops and branches, and these have now been removed, so FHI-aims now only compares each of the coefficient weights of the basis functions from the current SCF iteration and sets the constraint for the next SCF iteration to the basis function with the largest weight. After this, the code then utilises ELSI to calculate the chemical shifts and occupations which were previously completed in FHI-aims.

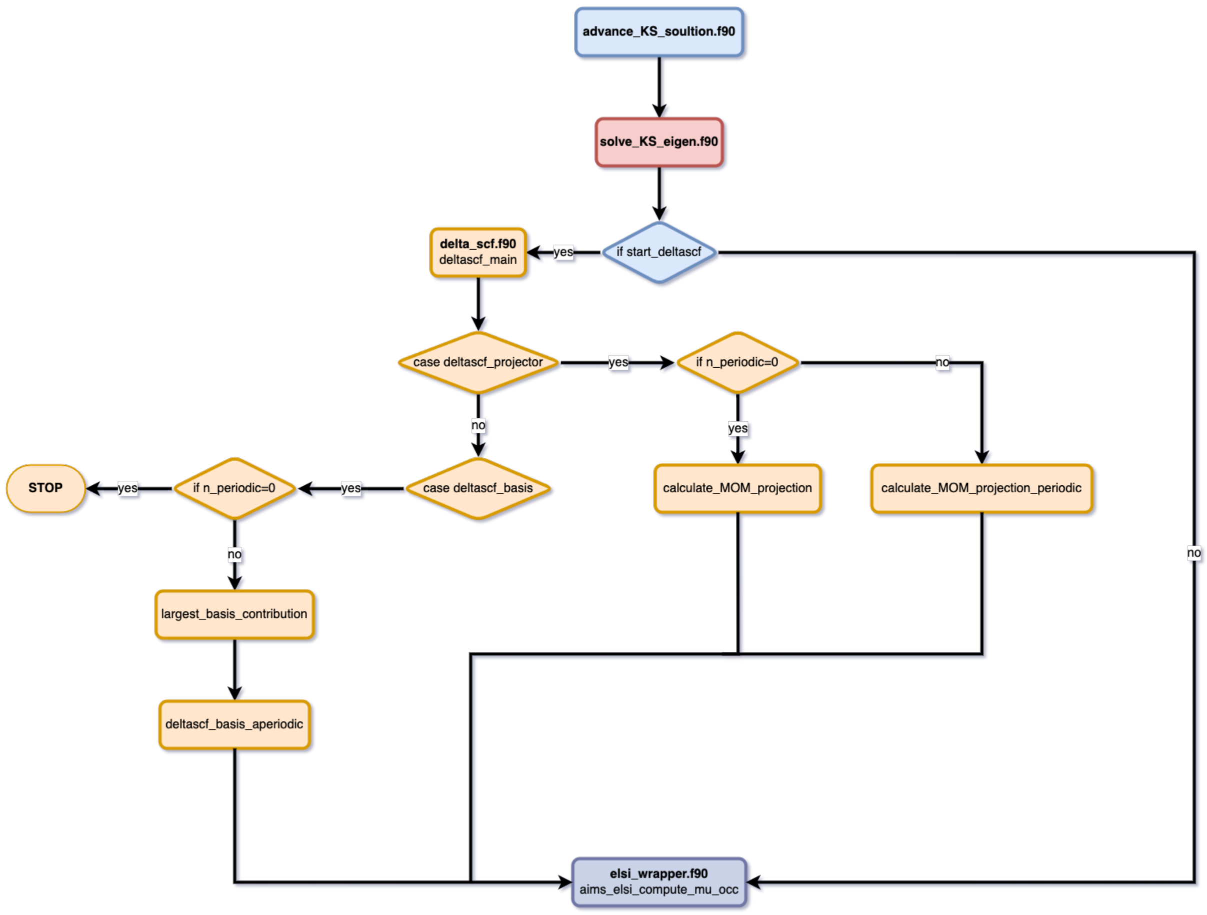

Figure 2: Flowchart depicting the new code logic when running a core hole constrained calculation in the FHI-aims code. Each colour represents a different Fortran module in the code.

2.3 Performance and scalability of new code

After implementing objectives 1 and 2, the performance of the new code and keywords was compared with the old code, to first check if the same result was achieved and secondly to compare the scalability of the new code. The code was also tested against a series of various settings and parameters to compare. In Table 1 we show in plain yes or no, a variety of cases in which the code is likely to be used and whether it can run or not.

Table 1: Table comparing the functionality of the old and new code looking at whether the code can run for either an aperiodic or periodic structure and using serial or parallel matrix algebra ("KS Method"). FOP and FOB refer to the old *force_occupation_projector* or *force_occupation_basis* keywords, whilst DSP and DSB refer to the same new *deltascf_projector* or *deltascf_basis* keywords.

Structure	KS Method	FOP	FOB	DSP	DSB
Aperiodic	serial	✓	✓	✓	✓
	parallel	✗	✗	✓	✓
Periodic	serial	✓	✗	✓	✗
	parallel	✗	✗	✗	✗

All functionality from the old code is retained in the new code. One thing to note about an FOB calculation in the original code is that, under periodic conditions, the calculation would run but not correctly apply the constraint and so would never converge. In the new code, we introduced a warning for users that this particular run mode does not work. THe calculation is stopped in this case during the initial calculation checks. What has been improved and will benefit the wider community is that now for aperiodic calculations, the new DSP and DSB keywords can perform in parallel, which was not capable before. This leads to a significant scaling increase which can be seen in figure 3.

A number of scaling tests were performed to compare the differences between the new and the old code. We tested both the old forced occupation basis (FOB) and forced occupation projector (FOP) keywords. The systems we used to test this were peroxide-terminated diamond clusters with 205 atoms for the non-periodic case and a similar system except with 287 atoms in the periodic example. As FOB is only implemented for non-periodic systems, only the diamond cluster was tested with this keyword, whereas both the cluster and periodic system were tested with FOP. For the periodic systems, tests were done at both the Γ-point and with a k-grid of $8 \times 8 \times 1$. To ensure the maximum probability of SCF convergence for the core hole, the 'double initialisation' method was used as first outlined by Kahk et al.[3]. This helps to prevent the delocalisation of the core hole across near-degenerate Kohn-Sham states, especially in symmetry-equivalent atoms of the same element. The double initialisation method involves performing a single-point (SP) calculation to SCF completion with an additional charge of 0.1. Then, using restart files generated from the 1st initialisation calculation, another SP calculation is performed, constraining the occupation of a specific Kohn-Sham state, with a charge of 1.1 for a single SCF step. This step is the key as it breaks the degeneracy of the targeted core Kohn-Sham state with other states of similar energy across the same chemical species. Finally, this output is used to continue the SP calculation until SCF convergence is reached but with a charge of 1, which we denote as the 'hole' calculation. For the purposes of this work, we only investigated the scaling of the hole calculation.

The tests were performed by allowing the calculation to run 20 self-consistent field (SCF) iterations, of which the average of the final 10 was taken. Not taking the first 10 iterations into account allowed enough iterations for the time taken per SCF step to stabilise for the first several large jumps in the absolute energy. The runs were stopped after 20 iterations as the final result was not a quantity of interest for the scaling tests, and this allowed enough time for a meaningful average to be calculated.

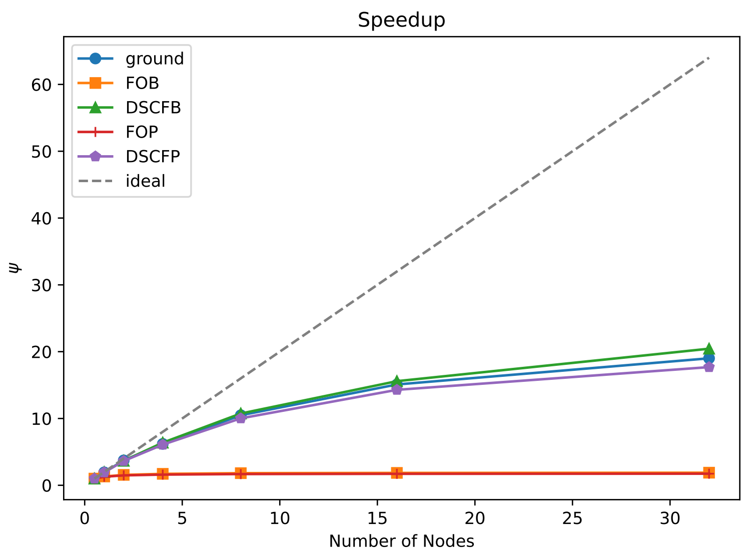

Figure 3: Comparison in speedup between the old force occupation basis (FOB) and force occupation projector (FOP) routines against the new deltascf_basis (DSCFB), deltascf_projector (DSCFP), and their corresponding ground-state calculation. The dashed line represents an ideal 1:1 speedup with respect to the number of processes. Note that the x-axis is given in nodes, where each node contains 128 CPU cores.

Speedup (ψ) measures how a code deviates from scaling linearly with calculation time with respect to the number of processes and is a consequence of Amdahl's Law[6]. Ideal speedup infers that there is perfectly linear scaling, and the increase in communication times between processes with an increasing number of processes does not affect the overall calculation time. ψ is calculated by $\psi = \frac{\psi(1)}{\psi(P)}$, where P is the number of processes. The results for the speedup data are displayed in figure 3. With both of the new routines, the speedup is shown to be comparable to that of a ground-state calculation, whereas previously, there would be almost no improvement in scaling performance when utilising more than 5 nodes on Archer2.

A similar picture is also portrayed with the parallel efficiency (ε) and Karp-Flatt metrics (e), shown in figures 4 and 5, respectively[7]. The efficiency is calculated from the speedup, where $\varepsilon = \frac{\psi}{P}$. Similarly, the Karp-Flatt metric is also calculated from the speedup but can also elucidate performance aspects of the code that can be challenging to observe from the speedup and efficiency. The formula for e is given below:

$$e = \frac{\frac{1}{\psi} - \frac{1}{P}}{1 - \frac{1}{P}}. \tag{2}$$

For e, the gradient of the curve with respect to P is significant, as it indicates if the slowdown with increasing processors is due to an increase in parallel overhead, which is the case if the gradient is positive. Otherwise, if the gradient is flat, then it signifies that e% of the computation time was run in serial. We show in figure 5 how we have improved the parallelisation from less than 50% for both FOB and FOP to at least 95% in DSCFB and DSCFP. This has been achieved whilst ensuring that the parallel overhead has not substantially increased with increasing processes from the previously implemented forced occupation methods, as the gradient of the curves is still flat. Furthermore, the level of parallelisation is now comparable to that of the ground state calculation.

Not only have we demonstrated significant improvements in the scaling, but the absolute calculation time has also been significantly reduced. The results for this are given in figure 6. Here we observe substantial improvements, where for instance, the original code, when running on 32 nodes (4096 cores), was slower per SCF step than the re-factored code being run on half a node (64 cores).

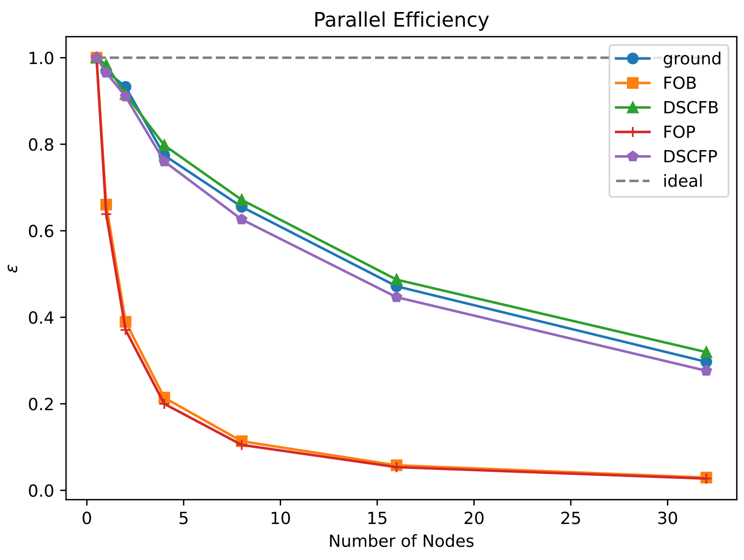

Figure 4: Comparison in parallel efficiency between the old force occupation basis (FOB) and force occupation projector (FOP) routines against the new deltascf_basis (DSCFB), deltascf_projector (DSCFP), and their corresponding ground-state calculation.

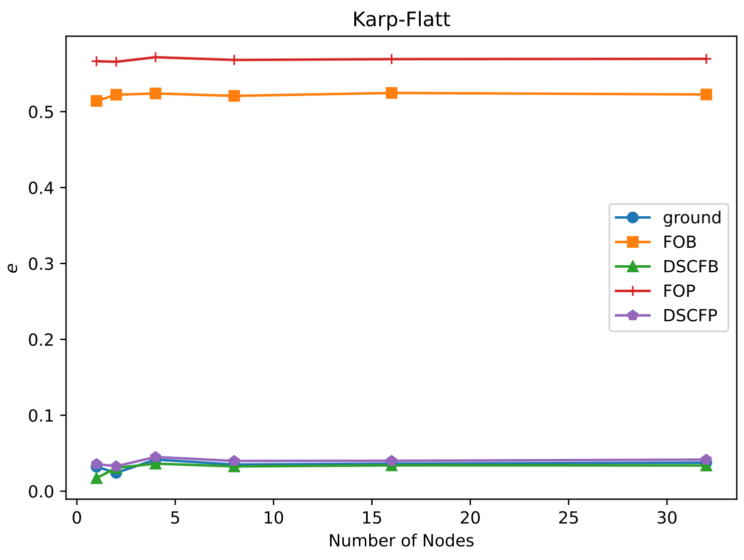

Figure 5: Comparison of Karp-Flatt metric between the old force occupation basis (FOB) and force occupation projector (FOP) routines against the new deltascf_basis (DSCFB), deltascf_projector (DSCFP), and their corresponding ground-state calculation.

3 Outlook

The work carried out in this project has been pushed into the main FHI-aims codebase and is now available to all FHI-aims users on ARCHER2. The progress that has been made from this project will significantly help the simulation of core-level spectroscopy by bringing the functionality up to date and in line with the general FHI-aims code. The greatly improved scalability and memory efficiency for the aperiodic structures will allow for much larger systems to be investigated. We have also developed a new tutorial

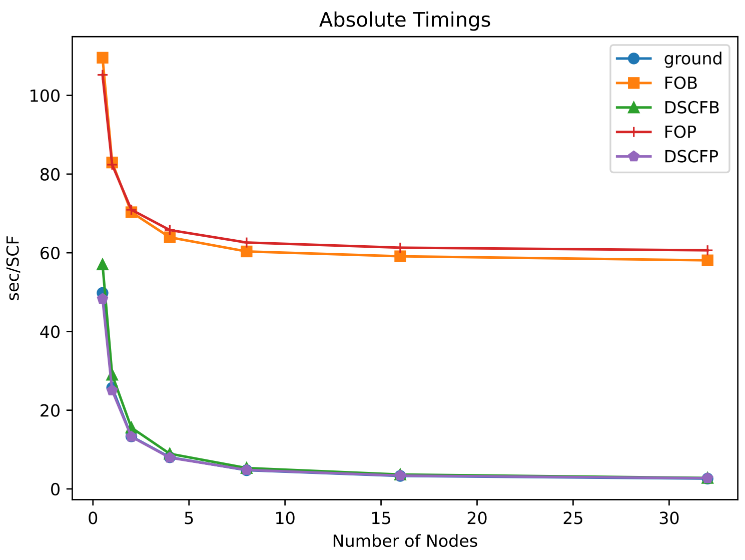

Figure 6: Absolute timings from the scaling tests, given as the average number of seconds taken per stabilised SCF step.

on how to simulate core-level spectroscopy using the new code, which will allow for these types of calculations to be performed by a much broader audience and allow for new science to be investigated.

Looking forward, the next aim will be to tackle the two objectives that were initially part of this project, namely improving the core hole localization, achieving a robust black-box localization method, and including spin-orbit functionality.

Acknowledgements

This work was funded under the embedded CSE programme of the ARCHER2 UK National Supercomputing Service (http://www.archer2.ac.uk).

References

[1] V. Blum, R. Gehrke, F. Hanke, P. Havu, V. Havu, X. Ren, K. Reuter and M. Scheffler, *Computer Physics Communications*, 2009, **180**, 2175–2196.

[2] J. M. Kahk and J. Lischner, *Physical Review Materials*, 2019, **3**, 100801.

[3] J. M. Kahk, G. S. Michelitsch, R. J. Maurer, K. Reuter and J. Lischner, *The Journal of Physical Chemistry Letters*, 2021, **12**, 9353–9359.

[4] B. P. Klein, S. J. Hall and R. J. Maurer, *Journal of Physics: Condensed Matter*, 2021, **33**, 154005.

[5] N. A. Besley, A. T. B. Gilbert and P. M. W. Gill, *The Journal of Chemical Physics*, 2009, **130**, 124308.

[6] D. R. Gene and M. Amdahl, Association for Computing Machinery, Inc, 1967, pp. 483–485.

[7] A. H. Karp and H. P. Flatt, *Communications of the ACM*, 1990, **33**, 539–543.