D5.4

Final report on turn-key solutions for the various flagship codes

Marnik Bercx, Miki Bonacci, Emanuele Bosoni, Augustin Degomme, Andrea Ferretti, Alberto Garcia, Luigi Genovese, Henning Janssen, Nicola Marzari, Giovanni Pizzi, Francisco F. Ramirez, Vasily Tseplyaev, Daniel Wortmann, and Aliaksandr Yakutovich

Due date of deliverable: 30/11/2021
Actual submission date: 30/11/2021

Lead beneficiary: EPFL (participant number 6)
Dissemination level: PU - Public

Document information

Project acronym:	MaX
Project full title:	Materials Design at the Exascale
Research Action Project type:	European Centre of Excellence in materials modelling, simulations and design
EC Grant agreement no.:	824143
Project starting / end date:	01/12/2018 (month 1) / 31/05/2022 (month 42)
Website:	www.max-centre.eu
Deliverable No.:	D5.4

Authors: M. Bercx, M. Bonacci, E. Bosoni, A. Degomme, A. Ferretti, A. Garcia, L. Genovese, H. Janssen, N. Marzari, G. Pizzi, F.F. Ramirez, V. Tseplyaev, D. Wortmann, and A. Yakutovich

To be cited as: M. Bercx et al., (2021): Final report on turn-key solutions for the various flagship codes. Deliverable D5.4 of the H2020 project MaX (final version as of 30/11/2021). EC grant agreement no: 824143, EPFL, Lausanne, Switzerland.

D5.4 Final report on turn-key solutions for the various flagship codes

Content

1 Executive Summary

In the past 18 months, extensive progress has been made both in the AiiDA-core code, as well in the range and robustness of the workflows supported by the plugin of all MAX flagship codes.

This report summarizes the major improvements and the new delivered workflows for each of them, together with a few selected examples for each plugin. Thanks to these extensive development efforts, tens of new workflows are now available open-source, providing the possibility to compute in an efficient and reproducible way a large range of properties of materials.

The focus has been in general directed towards a few main broad areas: (1) supporting new workflows of materials properties, as well as extending the number of features of the underlying quantum code that are supported via AiiDA workflows; (2) increasing the easiness and robustness of the calculations and the workflows themselves (with robust error handlers in the workflows for common code failure modes, with automatic parameter choice and basis set selections for the codes, and with improved automatic convergence workflows); (3) improving the performance of the workflow engine and the workflows themselves, to enable efficient high-throughput runs of HPC simulations, as well as high-performance data analysis of the results.

Furthermore, a core focus of MAX in the past 18 months has been the development, implementation and testing of common-workflow interfaces for common tasks (like the relaxation of a crystal structure) for 11 different codes, including all MAX codes that perform total-energy calculations. The concept of these common workflows, just outlined in the previous MAX deliverable D5.3, has been further refined, implemented for all codes, and tested on various structures; the results have been published and are presented briefly below. The work is ongoing to support more basic properties, perform stringent verification tests of the results, and use these common workflow interfaces as the building blocks for truly common workflows to compute advanced materials properties (equation of state, dissociation curves, phonon dispersions, band structures, density of state, …).

2 AiiDA common workflows

2.1 Common interface

As discussed in section 9 of the MAX D5.3 report, one of the goals we set out over the past year was to construct a common interface to run the MAX codes via AiiDA. The objective here is to facilitate the submission of workflows to calculate material properties with different codes. This makes the code that runs the DFT calculations a simple additional "input" of the turn-key workflows developed in the AiiDA ecosystem, and avoids that researchers have to learn again how to run a DFT calculation if they want to switch code.

This will, on the one hand, allow for easy and powerful verification studies to be performed across multiple DFT codes (more details will be added in D5.7 M38); on the other hand, it will enable to make advanced DFT simulations easily accessible also to non-experts (e.g. experimental colleagues), toward the goal of not only having open and FAIR data sharing, but also FAIR access to HPC simulation capabilities, especially in combination with user-friendly GUIs that can be used by e.g.

experimentalists, like the one we already developed in AiiDAlab (for now only for Quantum ESPRESSO), and we presented in D5.5.

The first work chain (the technical term of a workflow in AiiDA) with such a common interface that we created allows users to perform a geometry optimization or "relaxation" of a structure, which forms the building block for a number of higher-level work chains discussed below. The interface consists of two components:

1. `<Code>CommonRelaxWorkChain`: the AiiDA WorkChain that implements the logic of the geometry optimization. As is clear from Fig. 1, although the inputs of this work chain are code-specific, the outputs are a set of properties as defined by the common interface: the relaxed structure S_r, forces F, stress tensor T, total energy E_t and total magnetization M_t.

2. `<Code>CommonRelaxInputGenerator`: as the inputs for the work chain component are necessarily code-specific, the second component is an input generator designed for each code, but with inputs defined by the common interface. The fundamental inputs we decided on are the structure, quantum engines, protocol and relaxation type. The engines specify the code(s) that are run as well as the resources used (walltime, CPU, memory, …), whereas the protocol is a simple string ('fast', 'moderate' or 'precise') that defines a set of input parameters for running the calculations (the specific choice is necessarily code-specific, since we include already 11 different codes with very different basis sets and algorithms). Finally, the relaxation type indicates the degrees of freedom in the geometry optimization (none; positions only; cell only; change everything fixing the volume; …). Note that allowing no degrees of freedom ('none') basically corresponds to a static calculation.

So far, eleven codes or quantum engines support the common interface, including all the MAX codes that perform total-energy calculations (Quantum ESPRESSO, Siesta, Fleur, CP2K and BigDFT).

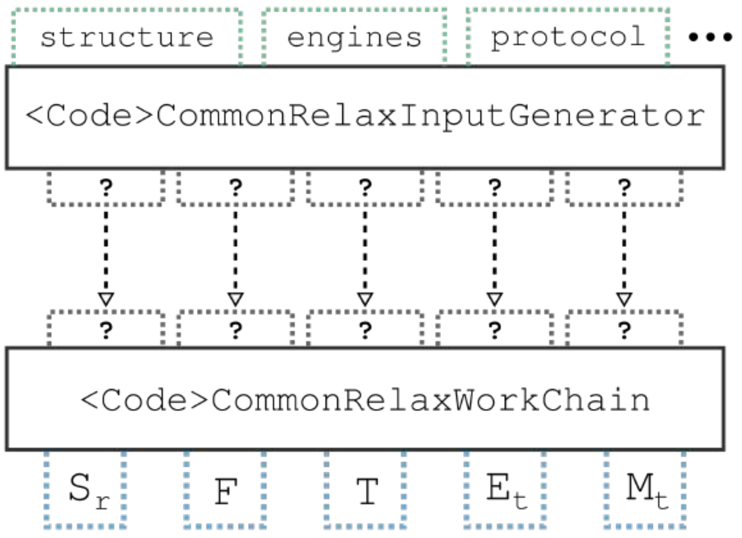


Fig. 1: Schematic diagram of the common interface for the relax workflow. See the main text for a full explanation.

2.2 Equation of State

Based on the common relax workflow, a set of higher-level workflows can be constructed for calculating physical properties in a code-agnostic way. A relatively straightforward but important example here is the `EquationOfStateWorkflow` (Fig. 2), which takes an input structure S_0 and scales its volume N_i number of times. It continues by first executing a "reference" common relax workflow on the "central" structure. This is also used as an additional input for the common relax workflows to calculate the energy of structure for each scaling factor. The goal of providing this reference structure is to ensure that the other relax workflows can use numerical parameters (k-points, cutoffs, …) compatible with it, so that energy differences between calculations are meaningful and well defined.

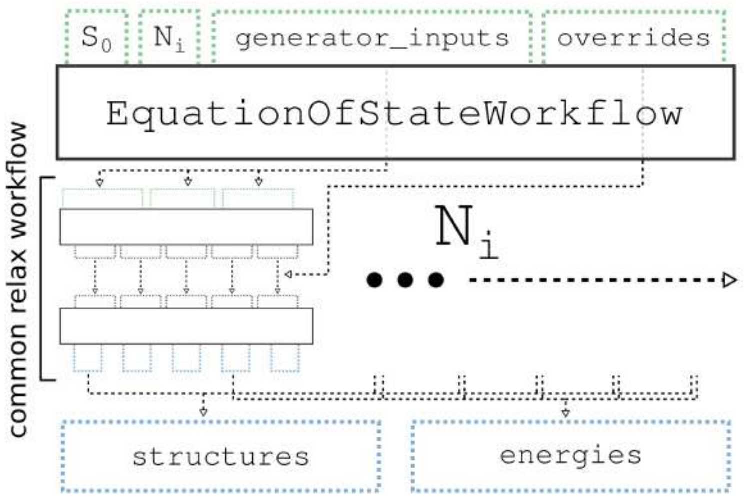


Fig. 2: Schematic diagram of the code-agnostic equation of state workflow.

Fig. 3 shows the equation of state of some simple structures, calculated using the `EquationOfState` work chain described above. At each volume, the atomic positions are optimized but the volume is kept fixed. For Si and Al, this means the cubic unit cell is kept fixed, however for the tetragonal phase of GeTe the cell is allowed to optimize using the `positions_shape` relaxation type (i.e. both cell vectors and internal atomic positions are free to move, with the constraint of constant volume), which is only supported by five of the quantum engines. Although there is a good match between the various codes, there are some slight but visible differences. This highlights the importance of facilitating verification studies via common interfaces. Currently we are working on calculating the equation of state for a large class of 500+ oxide structures to have a more comprehensive cross-verification of the supported quantum engines. The results will be discussed in D5.7.

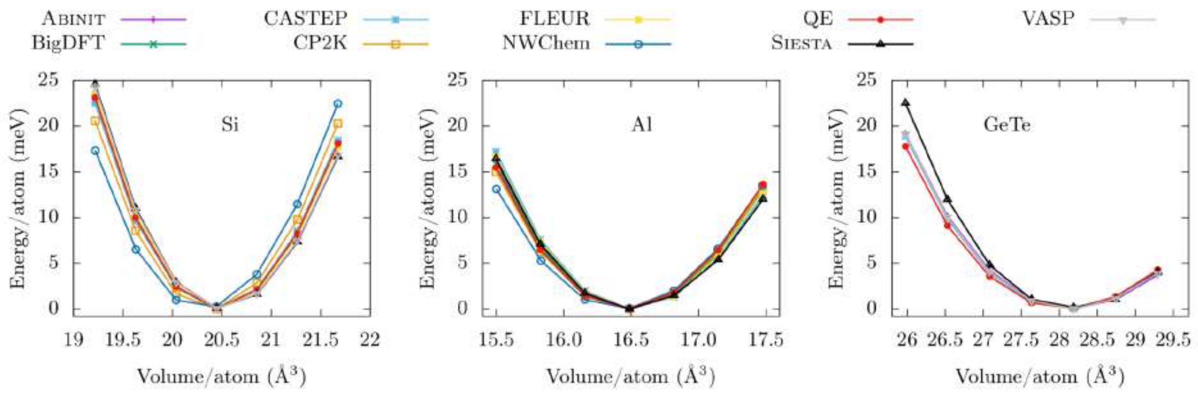


Fig. 3: Equation of state for Si, Al and GeTe, calculated with the code-agnostic `EquationOfState` work chain.

2.3 Dissociation curve

As a demonstration of the extensibility of the common relax interface to quantum chemistry problems, we have used the common relax interface to build a higher-level workflow to calculate the dissociation curve of a diatomic molecule. This `DissociationCurveWorkChain` allows the user to specify a range and number of distances between the sites in the diatomic module for which the energy will be calculated using the common relax workflow and relaxation type “none”. As an example, Fig. 4 shows the dissociation curve of the H_2 molecule calculated with the `DissociationCurveWorkChain`.

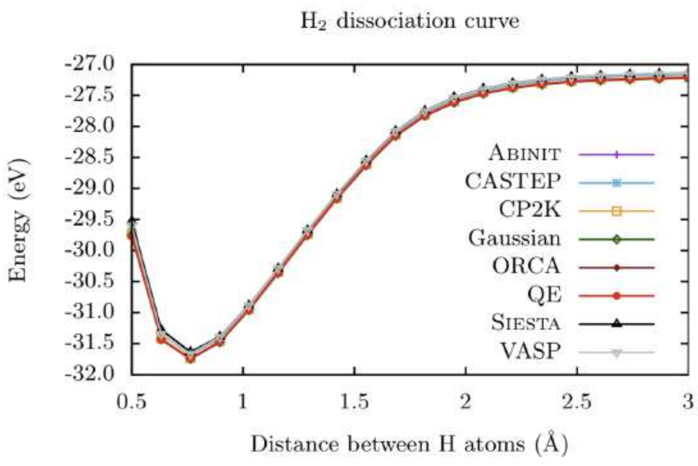


Fig. 4: Dissociation curve for the H2 molecule calculated for 8 different quantum engines using the code-agnostic workflow.

3 Improvements in AiiDA core

After the release of AiiDA 1.0 with all the major changes involved in it (the complete redesign of the engine and the database optimizations to scale to hundreds of thousands of concurrent workflows), most of the incorporations along the 1.x series of releases have been focused on increasing the robustness of the code, improving the accessibility to the platform, and implementing features that facilitate the creation and utilization of workflows and plugins.

3.1 Configuration Setup

An important example of the increase of robustness can be seen in the improvements in the way AiiDA manages the configuration file. First of all, the code will now create a backup of this file every time it needs to migrate it (due to changes in its structure during the addition or modification of configurable options). Moreover, these backups are not overwritten in successive migrations, but instead each one is given a unique name and kept as a safety measure. On the other hand, at the moment of being loaded, this JSON file is validated against a schema reference that ensures the integrity of its content. All in all, these new modifications dramatically increase the protection of the AiiDA environment against errors and transient bugs in the code.

Moreover, not only has the configuration setup become more stable thanks to this increased robustness of how the file is handled, but there has also been an increase in the options offered to the user. The daemon, for example, now allows to set the logging level, as well as the default number of workers to be started by default and the maximum number of concurrent tasks each of them can handle. The connection parameters of the RabbitMQ service can now be changed (e.g. to connect to external servers), whereas for the PostgreSQL database, peer-authentication support has been added. Finally, the ProxyJump SSH option now allows to set up an arbitrary number of proxy jumps without additional processes explicitly launched, thus providing an improved control over the lifetime of the different connections.

3.2 Direct interfaces: CLI, ORM and REST-API

When it comes to assessing the user interaction with AiiDA, the two most direct points are through the verdi command line interface (CLI) and the ORM objects. A third more indirect path is through the REST API, which allows to expose the AiiDA setup as a service via HTTP(S) requests (for example, this is how the Materials Cloud platform interacts with the underlying AiiDA instances). All of these points of access received important upgrades throughout the latest releases.

The verdi CLI incorporated a command for dumping the content of the file repository of a node directly in a given location of the filesystem (more easily accessible to the user). The options of all the setup commands were expanded so that they can not only read local YAML files, but they can also understand URLs and fetch the corresponding files when hosted online. A command for generating a useful summary of the database now provides users with further information about their setups, as well as aiding developers when troubleshooting problems.

The ORM, on the other hand, recently made available to the users some of the tools that were previously only used internally by the code, or that could only be accessed in a limited way. The function to delete nodes, for example, was only available through the respective CLI command, but now it is exposed through the public ORM so that it can be called from the shell or used in scripts. A powerful utility that extends the QueryBuilder so that it can also perform graph traversal was also recently incorporated. The AiiDA Graph Explorer (AGE) is a generic tool that enables new complex searches, including advanced elements such as infinite recursion of the search criteria.

Finally the REST API also received various improvements. Some of these include having a list of endpoints to show up at the base URL for requests, or including a complement to the endpoint that returned the different types that existed in the database, so that it would also show a count of instances for each, to generate statistics in an easy fashion. A major extension worth pointing out is the inclusion of the querybuilder endpoint; this is the first POST method included so far. Like its name suggests, this POST endpoint needs to be provided with a proper AiiDA querybuilder dictionary, and it returns as a response the results of the corresponding QueryBuilder query, initialized with said dictionary and executed.

3.3 AiiDA Groups

The ability of AiiDA to organize nodes into "groups" is a general underlying feature of the code that can be accessed both through the CLI (the group commands) and the ORM (a "group" as a python object). As it has proven to be of great utility to help users interact with the content on their databases in a simple and expeditious manner, it became an interesting target for improvement.

Some of these improvements were structural, such as making groups sub-classable through entry points so that plugin developers could create their custom "group types". Examples of this are the "pseudo-families" groups in the aiida-pseudo plugin (see section 4), which have their own internal methods to automatically select pseudopotentials and run parameters when given a crystal or molecular structure. Others, such as the "GroupPath" utility, involved making the user interface more intuitive. With this second one, groups could now be structured and navigated as if they were in a folder hierarchy.

3.4 Calculations and Workflows

As the main idea of AiiDA is the automation of complex workflows and the underlying management, data files manipulation is mostly performed under the hood without the user having to worry about copying or syncing files. As in some cases this started to become a limitation, two new tools were added to allow users more control over how to handle their data.

- The "Transfer Calcjob" is a calculation that allows the user to manually select and transfer between computers files associated to nodes normally handled by AiiDA. This enables, for example, the retrieval of files from the folder where a calculation was run even after the process has finished.
- The "stashing" option is an addition to the underlying calcjob class that allows users to specify a location on the remote cluster where files will be copied after the calculation finishes. This allows to move data from the transient directory where runs are typically executed to a more long-term storage path.

The advantage of the "Transfer Calcjob" is its versatility, as it can be used to retrieve files from already finished calculations, set up remote folders to be used as restart sources for other processes, etc. However, users need to manually execute it or insert it in their scripts, and plugin designers need to incorporate it into their workflows. On the other hand, while the "stashing" option is more specific, its inclusion on the base calcjob class means it is immediately available in all existing AiiDA workflows without the need for the plugin developers to do any adaptations.

The other important addition to the toolbox of base processes was the “BaseRestartWorkChain”. Originally designed specifically for controlling and restarting calculations from Quantum ESPRESSO, the model was soon adopted by many other plugin developers. It was therefore decided in the latest releases of the code to try to abstract it into a general purpose workchain that could wrap any type of calculation. This is now available thus directly in AiiDA core. It was also necessary to develop and introduce the concept of “process handlers”: a general way for users and plugin designers to describe how to handle specific errors of a code in a way that can later be automated by the workchain.

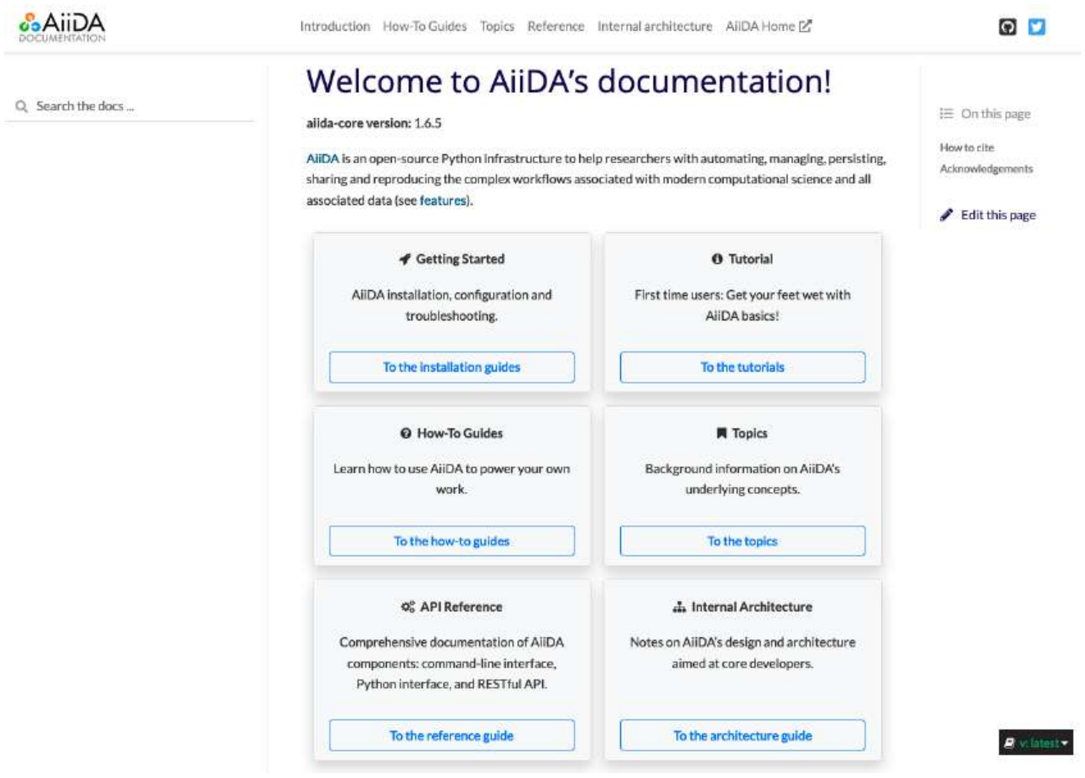
3.5 Improved Documentation

Another important step in increasing the user friendliness of the code was the major restructuring of the online documentation. This process involved modifications of many types, such as:

- Conceptual improvements and cleanup of the topics presented (for example, deciding on a consistent use of "plugin" vs "plugin package" terminology).
- Incorporating new technologies that allow to have a cleaner presentation or simpler underlying code (like the use of panels or links via intersphinx mapping).
- Adding new sections for topics which were complicated to comprehend for users and/or recurring problems we noticed in our support channels (new instructions on “How to extend workflows”, troubleshooting setup fails in complicated PostgreSQL settings, etc).

The revamped documentation also has an improved overall structure, with a brand new “Getting Started” section that allows new prospective users to very quickly set up an AiiDA environment and try out the code with a very basic tutorial. This tutorial is focused on showcasing the key features of the code and conveying its main usefulness in a very short period of time (less than 15 minutes).

At the same time, the old content was better distributed into a more practical “how-to” section for easy reference, and an in-depth explanation of the concepts and topics so that advanced users can have a better understanding of the tools they are using. Finally, there is also another section which details the internal architecture of the key constructs of the code, reserved for AiiDA core developers or users and plugin designers who wish to have the most precise knowledge of the code.



Welcome to AiiDA's documentation!

aiida-core version: 1.6.5

AiiDA is an open-source Python infrastructure to help researchers with automating, managing, persisting, sharing and reproducing the complex workflows associated with modern computational science and all associated data (see features).

Getting Started
AiiDA installation, configuration and troubleshooting.
To the installation guides

Tutorial
First time users: Get your feet wet with AiiDA basics!
To the tutorials

How-To Guides
Learn how to use AiiDA to power your own work.
To the how-to guides

Topics
Background information on AiiDA's underlying concepts.
To the topics

API Reference
Comprehensive documentation of AiiDA components: command-line interface, Python interface, and RESTful API.
To the reference guide

Internal Architecture
Notes on AiiDA's design and architecture aimed at core developers.
To the architecture guide

Fig.5: landing page of the new AiiDA core documentation (https://aiida.readthedocs.io/). The re-organized sections are cleanly displayed using the new tile objects.

3.6 New file repository in AiiDA 2.0

Finally, during the past few months a substantial amount of effort has been put into developing a new and more efficient data repository for handling all the files and information that AiiDA needs to store outside of the database.

So far AiiDA stored all files inside a normal file repository, in which each node is assigned a folder with a path given by its UUID, and sharding was used to have a 2-level hierarchical structure. This simple design was starting to reach its limitations, as it didn't allow to very easily implement features such as file compression or deduplication. Moreover, it started to show some inefficiencies for high-throughput data storage, with some computers hitting the maximum number of inodes before the actual disk space was filled.

The new data repository has been more cleanly separated from the rest of the AiiDA code and presents a more complex and independent design that allows for a more efficient handling of data. Its underlying structure resembles that of an object storage, it automatically takes care of content duplication, and it allows the user to perform maintenance operations to optimize the disk usage (such as packing and compressing the data). At the same time, AiiDA hides all of this complexity away from the user interface, which presents the same intuitive folder structure when accessing the contents of nodes.

Some of the adaptations on pre-existing AiiDA features to fully take advantage of the new repository have already been introduced in the latest releases of the 1.x series. One example of this was the full refactoring of the tools in charge of exporting and importing AiiDA databases. However, as the change in the repository introduces some critical changes, it was decided that its incorporation would be released as a major revision of the code, AiiDA 2.0. It is therefore being bundled up with other changes that might involve deprecation of old features or some important changes in the codebase. The release is currently being finished and should be available in the next couple of months.

4 AiiDA-pseudo: A library for installing pseudopotentials

To improve the user-friendliness of running calculations in AiiDA which require pseudopotentials, we have developed a new plugin package that relies on the recent improvements on AiiDA groups discussed in section 3.3 to quickly install families of pseudopotentials. These are implemented as their own group type, and come with methods that automatically validate pseudopotential nodes added to them, and make it easy to obtain the required pseudopotentials in the correct format for elements or structures. Moreover, it is possible to store recommended cutoff values for the wave functions and charge density in the extras of such a family, which can then be retrieved via the `get_recommended_cutoffs()` method.

Installation of "established" pseudopotential families is made very simple by providing dedicated install commands via the CLI. A family is considered *established* when it has a set of rigorously tested pseudopotentials with convergence tests which have been published or are publicly available. Examples here include the SSSP and pseudo-dojo libraries, which can be installed in any configuration (version, stringency, pseudopotential format, ...) with a single command that automatically adds the corresponding recommended cutoffs. Other families can be installed as well, of course, and recommended cutoffs can be added manually using the Python API.

All code is publicly available on GitHub[1] and documented on Readthedocs[2]. The `aiida-pseudo` features are already extensively used in the AiiDA-Quantum ESPRESSO plugin, for example in the protocol methods described in section 5.2. The package already supports a wide range of pseudopotential formats though, not just Quantum ESPRESSO's UPF format. Hence, other plugins - especially those involved in the common-workflows project discussed in section 2 - are also in the process of adapting their code to rely on the features of `aiida-pseudo`.

5 AiiDA-Quantum ESPRESSO workflows

5.1 Improvements in workflows

Since the last report, two new workflows have been released in the Quantum ESPRESSO plugin:

`PwBandsWorkChain`

One of the classical examples of results obtained with electronic structure calculations is the band structure. The `PwBandsWorkChain` is a fully automated turn-key workflow for calculating this

[1] "aiidateam/aiida-pseudo - GitHub." https://github.com/aiidateam/aiida-pseudo. Accessed 20 Oct. 2021.
[2] "AiiDA pseudo plugin — aiida-pseudo 0.6.3 documentation." https://aiida-pseudo.readthedocs.io/. Accessed 20 Oct. 2021.

quintessential property using Quantum ESPRESSO. Fig. 6 shows the provenance graph of the work chain. It starts with an (optional) optimization of the geometry, effectively wrapping the PwRelaxWorkChain discussed in section 3.1 of the previous report D5.3. Once the final structure has been determined, the SeeK-path library is used to symmetrize the structure within certain tolerances and then determine the path along which to calculate the band structure. A final SCF calculation is performed to obtain the charge density, after which the band structure is calculated. All of the calculations are based on Quantum ESPRESSO's `pw.x` code and are wrapped in a `PwBaseWorkChain` so errors can be handled on the fly.

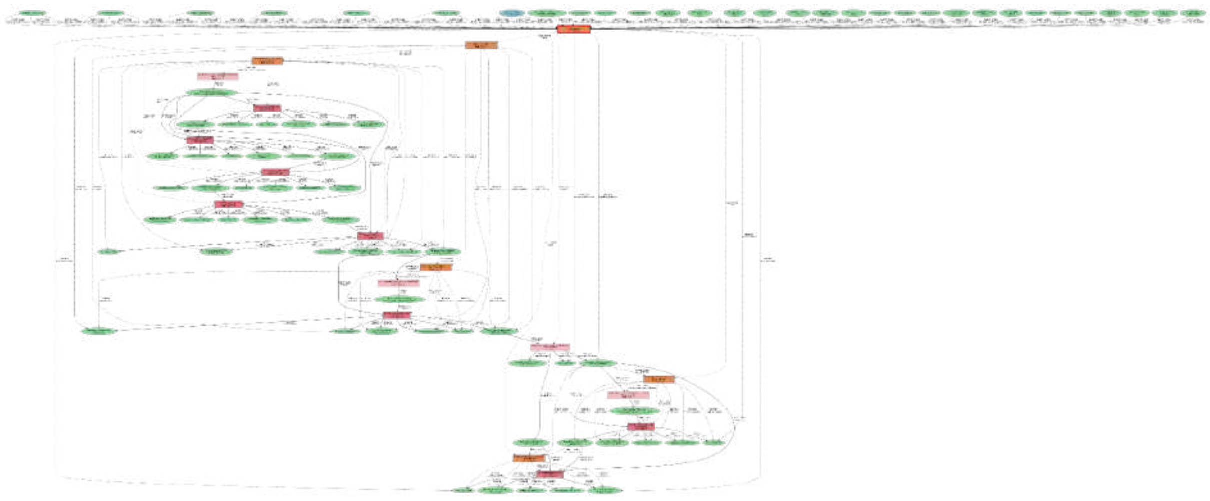


Fig. 6: Provenance graph of a successful run of the `PwBandsWorkChain`. The work chain starts with an (optional) optimization of the geometry, runs an SCF to obtain the charge density and then restarts from this charge density to calculate the band structure along the path determined by the SeeK-path library.

`PdosWorkChain`

Another crucial property to many research questions in materials science is the density of states (DOS), as well as the projections of the Kohn-Sham orbitals on the atomic orbitals of the sites in the structure (projected DOS or PDOS), in order to obtain an idea of the character of the states at each energy level. Fig. 7 shows the provenance of a successfully completed `PdosWorkChain`, designed to obtain both properties with Quantum ESPRESSO via a turn-key workflow. For the structure provided by the user, the work chain starts by calculating the charge density using the `pw.x` code of Quantum ESPRESSO, wrapped in a `PwBaseWorkChain` to benefit from its error-recovery features. The second step is to fix the charge density and use a non-self-consistent field (NSCF) calculation to obtain the Kohn-Sham states and energies on a much finer k-point mesh. Finally, the DOS is calculated using the `dos.x` code via the `DosCalculation` plugin and the projections using the `projwfc.x` code implemented in the `ProjwfcCalculation` plugin.

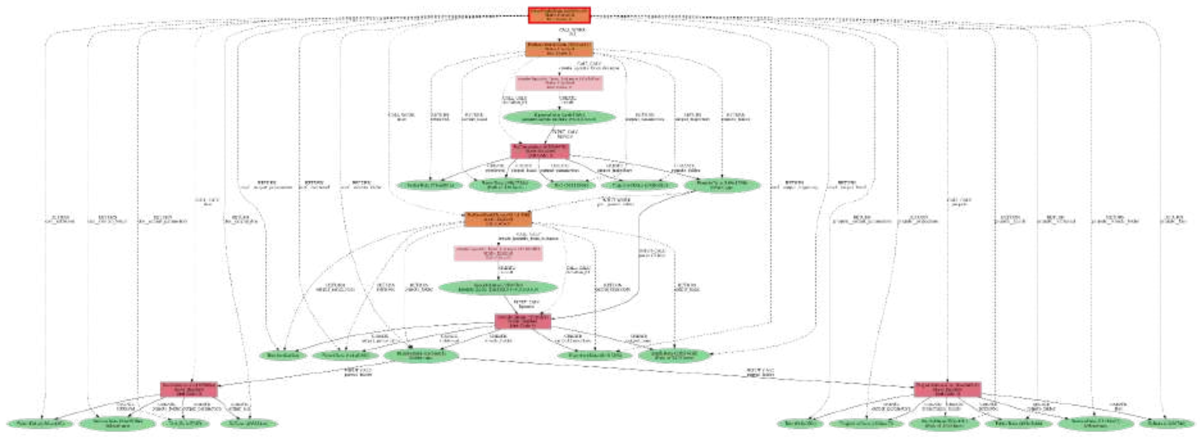
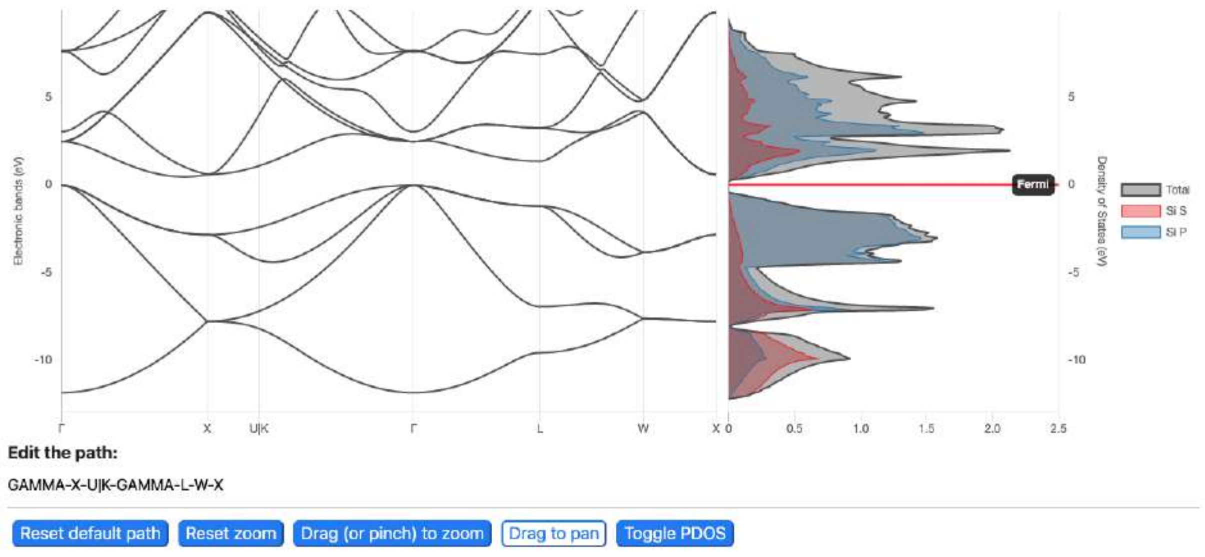


Fig. 7: Provenance graph of the `PdosWorkChain`. For the given input structure, the charge density is calculated first. This is passed to a NSCF calculation with a finer k-point mesh. Using the dos.x and projwfc.x post-processing codes of Quantum ESPRESSO, the DOS and projections are subsequently calculated.

The results for both work chains can be visualized using a widget implemented to be run interactively in Jupyter notebooks, shown in Fig. 8. All of the code used to run the work chains is publicly available on GitHub at the `aiida-quantumespresso` repository[3]. The documentation has also been recently revamped and is available on Readthedocs[4].



Edit the path:

GAMMA-X-U|K-GAMMA-L-W-X

Fig. 8: Interactive visualization of the band structure and (P)DOS results obtained from running the `PwBandsWorkChain` and `PdosWorkChain` for crystalline silicon.

[3] "aiidateam/aiida-quantumespresso - GitHub." https://github.com/aiidateam/aiida-quantumespresso. Accessed 20 Oct. 2021.
[4] "AiiDA plugin for Quantum ESPRESSO - Read the Docs." https://aiida-quantumespresso.readthedocs.io/. Accessed 20 Oct. 2021.

5.2 Protocols

One of the key aspects of running "turn-key" workflows in high-throughput is having a set of input parameters available that can be relied on for running a diverse range of structures with reliable precision. These input parameters, which we refer to as a "protocol", is also one of the key inputs for the common-workflow interface discussed in section 2.1. For Quantum ESPRESSO, the energy cutoff for the wave functions and charge density are provided based on the elements present in the structure and the recommended cutoff from the pseudopotential library used, for example the standard solid-state pseudopotentials[5] (SSSP) or pseudo-dojo[6]. For other parameters, such as the k-point sampling and smearing parameters, a set of 400 binary structures was used to perform an investigation into reliable choice by comparing the results for each setting with a set of high-precision parameters. The results for the k-points density (KPGRID) and smearing width (SMEAR) are shown in Fig. 9.

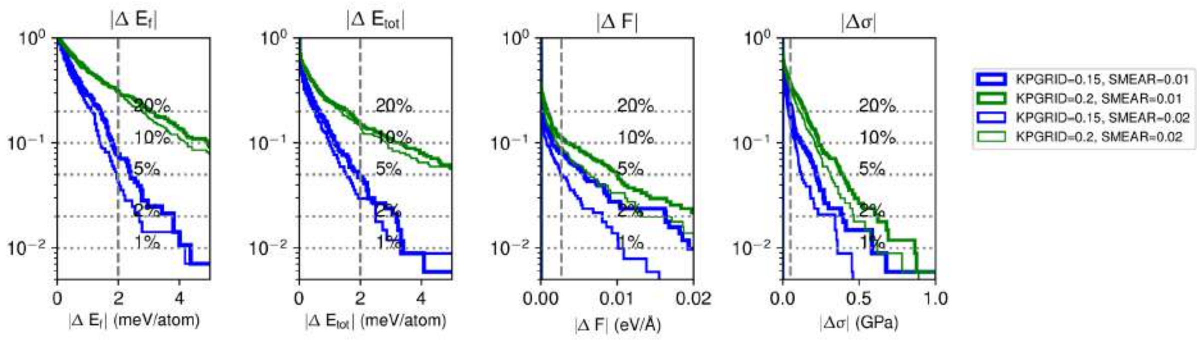


Fig. 9: k-point convergence studies for two different smearing widths; the y-values represent the amount of compounds outside the error-value on the x axis. The quantities shown are all expressed as an absolute difference versus the reference high-precision parameters. From left to right: formation energy E_f, total energy E_{tot}, forces F and stresses σ.

As an example: in terms of the formation energy, a k-point density of 0.15 $Å^{-1}$ yields between 90-95% structures with an error smaller than 2 meV/atom, with only small differences between the two choices in smearing width. Based on similar analyses for other computational parameters, a set of three protocols are constructed:

- **`fast`** uses the efficiency configuration of the SSSP and the following precision settings: k-points distance is 0.5 $Å^{-1}$, self-consistency convergence threshold is $0.4 \cdot 10^{-9}$ Ry per atom,energy threshold for the ionic convergence is $1 \cdot 10^{-4}$ Ry per atom and forces threshold is $1 \cdot 10^{-3}$ Ry/bohr. The convergence on the stress is left to the default of the code which is 0.05 GPa.
- **`moderate`** uses the efficiency configuration of the SSSP and the following precision settings: k-points distance of 0.15 $Å^{-1}$, self-consistency convergence threshold of $0.2 \cdot 10^{-9}$ Ry per atom, energy threshold for the ionic convergence of $1 \cdot 10^{-5}$ Ry per atom and forces threshold of $1 \cdot 10^{-4}$ Ry/bohr. The convergence on the stress is left to the default of the code which is 0.05 GPa.

[5] "SSSP efficiency - Materials Cloud." https://materialscloud.org/sssp. Accessed 20 Oct. 2021.
[6] "Pseudo Dojo." http://www.pseudo-dojo.org/. Accessed 20 Oct. 2021.

- **`precise`** uses the precision configuration of theSSSP and the following precision settings: k-points distance is 0.1 Å^{-1}, self-consistency convergence threshold is $0.1 \cdot 10^{-9}$ Ry per atom, energy threshold for the ionic convergence is $0.5 \cdot 10^{-5}$ Ry per atom and forces threshold is $0.5 \cdot 10^{-4}$ Ry/bohr. The convergence on the stress is left to the default of the code which is 0.05 GPa.

In order to make these protocols easily available to the user of the turn-key workflows, we developed the `get_builder_from_protocol()` method for each of the work chains currently available in the AiiDA-Quantum ESPRESSO plugin that returns a fully populated builder that can then be directly submitted to the AiiDA daemon. This method only requires minimal inputs: the code(s) that the work chain needs to run, the input structure and chosen protocol (with `moderate` as the default). Moreover, there are several optional inputs that allow the user to indicate the type of material (`ElectronicType`, `SpinType`) and every input of the work chain and its sub-work chains can still be changed using the `overrides` input.

5.3 SIRIUS: direct minimization via ensemble DFT

In the previous report, we already discussed how critical robustness is when running the MAX codes in high-throughput, since even with a sensible set of input parameters (as e.g. those described in section 5.2) there are several failure modes that can occur. The implementation of the `BaseRestartWorkChain` in AiiDA core already provided the developer of a work chain with the opportunity to write error handlers that can deal with certain failure modes. However, even with the error recovery features of the `BaseRestartWorkChain`, a significant number of work chains still fail to complete successfully.

The outer ring in Fig. 10 shows the distribution of the initial failure modes of the calculations run for the 3D crystal database (3DCD; more details will be found in D5.7 due at M38). We can see that electronic convergence issues, i.e. the failure of the SCF cycle to converge to the ground state, is one of the most prevalent failure modes for both the `PwBaseWorkChain` and `PwRelaxWorkChain` of the AiiDA Quantum ESPRESSO plugin. Looking at the ratio of these for which the work chain is able to recover with a suitable error handler (inner circle; “Fixed” vs “Failed”), many of these failure modes cannot be recovered at the work chain level. In contrast, although the ionic convergence is the most prevalent issue for the `PwRelaxWorkChain`, many of the initial failures are fixed by the work chain.

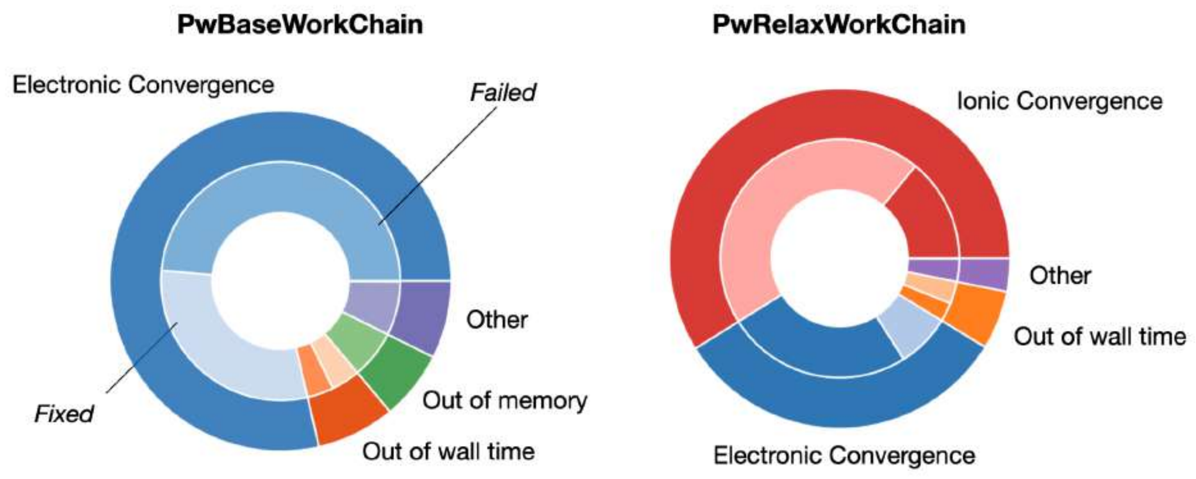


Fig. 10: Failure modes of the work chains run for building the 3DCD database. The outer ring represents the distribution of the failure modes of the initial calculation run within the restart work chain. The inner ring shows what fraction of these initial failures can be "fixed" by the work chain.

Since successful convergence to the ground state is such a critical and prevalent problem, a robust version of a direct minimization method based on ensemble density-functional theory has been implemented within the SIRIUS code, and publicly available via the `nlcglib`[7] package.

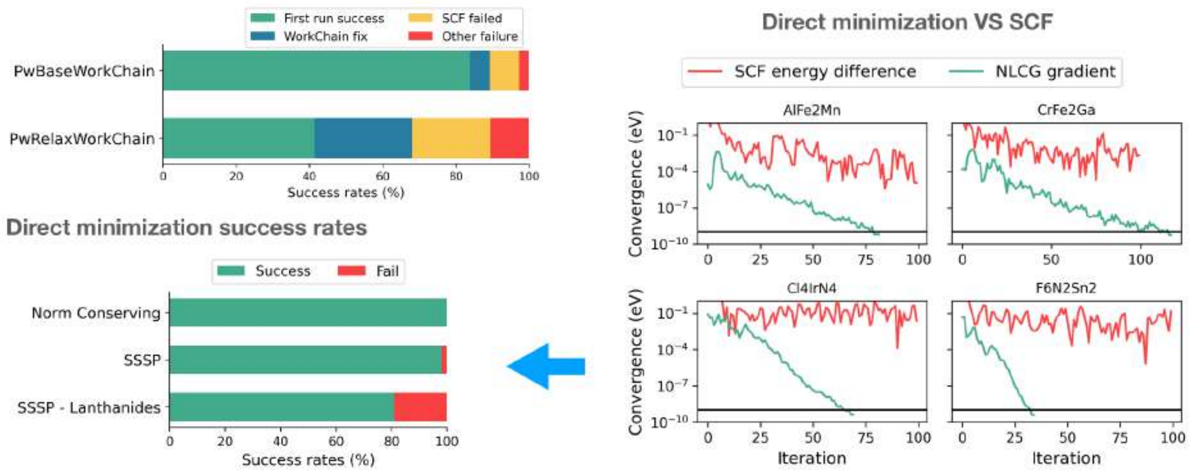


Fig. 11: Results for the direct minimization implementation in the SIRIUS code. The image in the top left shows the success rates of the PwBaseWorkChain and PwRelaxWorkChain, based on the first run only (green) and with error recovery from the work chain (blue). It also shows the fraction of unrecoverable failures related to the lack of SCF convergence (yellow), and other failures (red). On the right are some comparisons between the SCF and direct minimization of representative runs executed on the 100 structures in our test set. The bottom left graph shows the success rates of the direct minimization for these test runs, using norm-conserving pseudopotentials and the SSSP library. Finally, the success rate for the SSSP when running a new test set of only lanthanide-containing structures that fail to converge with the SCF is shown.

[7] "simonpintarelli/nlcglib: Nonlinear CG methods for wave ... - GitHub." https://github.com/simonpintarelli/nlcglib. Accessed 20 Oct. 2021.

This implementation has been extensively benchmarked using AiiDA for a diverse data set based on 100 structures where the self-consistency cycle failed to converge to the ground state for the 3DCD runs, using both norm-conserving and ultrasoft and PAW pseudopotentials.

The results for the direct minimization are shown in Fig. 11. From the work chain success rates, and the fraction of the unrecoverable failures related to the SCF convergence, it is once again clear that a successful implementation of the direct minimization would have a large impact on the robustness of the work chains, driving the success rate of the `PwBaseWorkChain` and `PwRelaxWorkChain` close to 97% and 90% respectively. Comparing the convergence of the SCF with the direct minimization in the image on the right, we can see that for the failed SCF runs shown, the direct minimization *is* able to converge without issue starting from the final charge density. Using norm-conserving pseudopotentials, the direct minimization is able to obtain the converged electronic ground state for all of the problematic structures.

Using the SSSP library, there are 2 failures. Closer inspection reveals that both structures contain a lanthanide, and that the failures are related to the fact that in some cases use of a PAW pseudopotential can result in a negative charge density during the electronic optimization. Rerunning the tests for a new structure set of 100 lanthanide-containing structures for which the SCF failed to converge confirms that this problem is more likely to occur when lanthanides are present. Even so, the direct minimization is able to converge for over 80% of the structures.

6 AiiDA-SIESTA workflows

The workflows and turn-key solutions built around the MAX flagship code SIESTA are collected in the python package `aiida-siesta`, with the full open source code available on GitHub[8]. The package is being continuously updated and improved. In particular, the code has been refactored to use the latest features of the `aiida-core` package (for instance the use of “validators” that perform sanity check on the workchains inputs) and to depend on the newly created `aiida-pseudo` package (see section 4) for the management of pseudopotentials.

In the next subsections we will explain in detail the main new features introduced regarding the workflows of `aiida-siesta`.

6.1 Support for Lua scripting for access to extra Siesta functionality. NEB workflow.

The SIESTA code can be internally scripted through Lua[9] scripts. The mechanism involves direct read/write access from the embedded Lua interpreter to selected Siesta variables. This allows in particular to control the variables related to the position and forces of the atoms, opening up the possibility to use new geometry/cell relaxation schemas and new molecular-dynamics and force-constants algorithms that are not internally implemented in SIESTA. Moreover, Lua scripting allows new types of investigations. For instance, it enables the finding of the Minimum Energy Pathway (MEP) connecting two local minima through the Nudged Elastic Band (NEB) method[10].

[8] "AiiDA Siesta plugins and workflows - GitHub." https://github.com/siesta-project/aiida_siesta_plugin. Accessed 20 Oct. 2021.
[9] "The Programming Language Lua." https://www.lua.org/. Accessed 20 Oct. 2021.
[10] "A climbing image nudged elastic band method for finding saddle ...." https://aip.scitation.org/doi/10.1063/1.1329672. Accessed 20 Oct. 2021.

The `aiida-siesta` package has been extended to support the use of Lua scripting through a dedicated input namespace in the SIESTA plugin. Examples of the use of novel geometry-relaxation algorithms are provided in the package. Moreover, a workchain called `SiestaBaseNEBWorkChain` has been created. It embodies the basic logic to perform MEP optimizations starting from a guessed path (encoded in a native AiiDA `TrajectoryData` object) and passing the appropriate options to the SIESTA plugin. This is very useful for the investigation of reaction paths and energy barriers for defect diffusion (see Fig. 12). For the latter, a dedicated package that uses `aiida-siesta` as a workhorse is being readied.

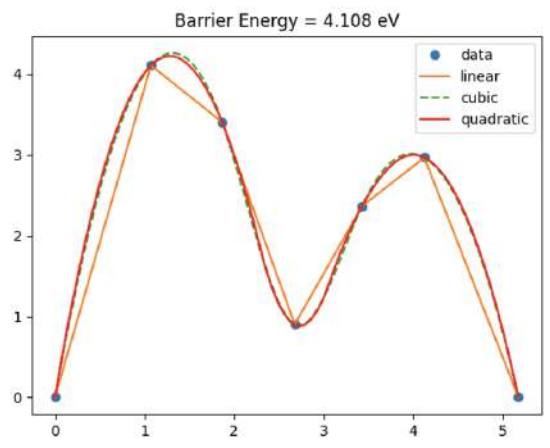


Fig. 12: Energy profile along the MEP (minimum-energy-path) for the diffusion of a H atom in bulk Si between two interstitial sites (model calculation to test the implementation of the NEB workflow in aiida-siesta).

6.2 Iterator, Converger, and Basis-set optimization workflows

A set of three workflows has been introduced with the aim to facilitate the optimization of the SIESTA calculation parameters. It is well known that before starting the study of a new system, it is important to make sure that the computational parameters are adequate for the calculation. This includes, for instance, the need for a convergence study on the k-points mesh and the mesh-cutoff, the search for the optimal basis on which to expand the wave functions, or the use of an accurate pseudopotential.

All these tasks can now be efficiently carried out through AiiDA thanks to three workchains. The first is called `SiestaIterator` and allows to iterate over any input of a siesta calculation in an automatic way. It is useful, for instance, in order to run the same calculation with several pseudopotentials. The second workchain is called `SiestaConverger` and it adds to the iterator logic the possibility to define a monitored quantity and a tolerance. These are checked at the end of every iteration step in order to establish the convergence of the monitored quantity. We foresee this workchain to be used for the choice, for instance, of mesh cutoff and k-points. Both work chains allow iteration over more than a single quantity. Two inputs can be incremented together or one at the time. For example, one can iterate at the same time over the k-points mesh and the broadening of the smearing for the occupation of the states. Sequential convergences are also allowed, enabling the full automation of convergence tests. Finally we defined a `BasisOptimizationWorkChain` that helps the search for an optimal basis. In the current implementation, first an iteration is run to understand the favorable basis size (SZ, DZ, DZP, …) and then the workchain performs a simplex optimization (Nelder–Mead method) with all the cutoff radii as optimization variables. This fully automatic basis selection is paving the way for a seamless use of SIESTA also for non-experts, since the optimization of the basis can be always inserted as a first step of other scientific workflows.

6.3 Optical properties

The Siesta code allows the calculation of some basic optical properties. The `EpsilonWorkChain` retrieves from SIESTA the imaginary part of the dielectric function as a function of the energy. Subsequently it calculates the electronic contribution to the static dielectric constant using Kramers-Kronig relations.

6.4 DOS and PDOS

The Density of States (DOS) and its projections on atomic orbitals (PDOS) are essential quantities for the interpretation of electronic properties of materials. The `DosPdosWorkChain` has been created to facilitate this task. A separate (usually denser) k-points mesh can be specified in input for the calculation of the DOS/PDOS. After the end of the self-consistent cycle (and possible relaxation), the siesta calculation is restarted with the new kpoints mesh and the DOS and PDOS (if requested) are calculated.

6.5 Other Siesta-related refinements

A key aspect of running “turn-key” workflows in high-throughput is providing suggested input parameters to achieve a target precision in calculations. A “protocol” system has been introduced for this purpose in the `aiida-siesta` package. This system was described in previous reports and now it has been extended to cover more use cases and it has been refined to have a more accurate selection of basis sets.

Two other features of the SIESTA code are now available for use in AiiDA workflows: the “floating sites” (ghost-atoms) and the “ion” files. SIESTA allows the inclusion of off-site orbitals (not centered around any specific atom) in the basis set used to expand the wave functions. These orbitals are often called “floating orbitals” and they are centered around a “fake” atom that, in the SIESTA terminology, is called “ghost atom” or “floating site” . The `aiida-siesta` package now allows specifying “floating orbitals” as an additional basis information. Position and kind of the “floating site” must be specified, together with the characteristics of the attached orbitals. The second feature that has been added is the possibility to specify, instead of basis and pseudopotentials, the so-called “ions”. In SIESTA, an “ion” is an entity that packages the set of basis orbitals and KB projectors for a given species. The “ions” specifications are collected in external files that are read at runtime by the code. These files can now be stored in the database thanks to a dedicated data class and they can be passed to the calculations as an alternative to the more conventional use of pseudo+basis. Moreover, several methods have been implemented to extract information on the nature of the “ions”, including the possibility to access (and plot) explicitly the radial part of the basis orbitals.

7 AiiDA-YAMBO workflows

The AiiDA-YAMBO workflows have been greatly improved since the last report. In particular, more sophisticated convergence algorithms have been implemented, and a more robust input/output layer has been developed through the yambopy package integration.

7.1 Integration with the yambopy package

Since the last deliverable, we integrated in the plugin the yambopy package[11], as a pure Python dependency. Yambopy is a Python package that helps to prepare, run and analyze yambo calculations, and is meant to be part of the yambo-core structure and co-developed with the yambo code itself. As explained below, yambopy in turn allows for a more robust and resilient AiiDA-YAMBO interfacing. In the AiiDA-YAMBO plugin, yambopy is mainly used for input creation and output parsing, in order to avoid duplication of code for these specific tasks. Yambo calculations are still performed entirely within the AiiDA engine. In particular, inside the plugin, now yambo inputs are created using the `YamboIn()` class, and the output parsing is done using the class `YamboFolder()`. These two classes are both imported from yambopy. This makes the parsing of yambo outputs straightforward, as the AiiDA-YAMBO parser is greatly simplified. Moreover, yambopy functions allow us to include among the feasible calculations also those addressing optical properties, i.e. Bethe-Salpeter (BSE) calculations: yambopy is able to easily parse excitonic eigenvalues/eigenvectors and classify them, for example finding the brightest exciton of the optical spectra. Then, the plugin stores the information in the AiiDA database. This is particularly useful in the `YamboConvergence` workflow, as we discuss in the next section. In Fig. 13 we report a sketch of the logical interaction between AiiDA-YAMBO and yambopy.

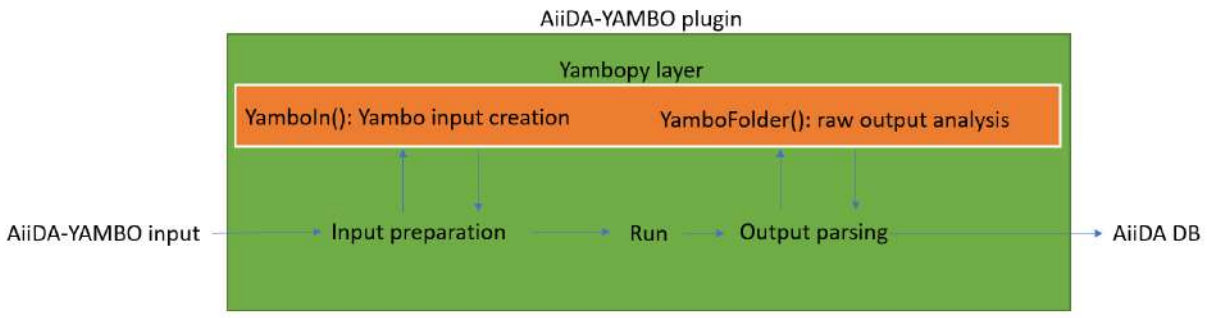


Fig. 13: schematic of the flow for a single yambo calculation performed by the AiiDA-YAMBO plugin. We observe that the yambopy layer is used for both input and output operations, and that the actual run is performed by the plugin.

7.2 Improvement of the YamboConvergence workflow

Convergence algorithm: GW calculations

The `YamboConvergence` workflow is the top-level workflow of the AiiDA-YAMBO plugin, and supports convergence for the various parameters involved in GW calculations. The most crucial

[11] "henriquemiranda/yambopy: Automatize yambo calculations ... - GitHub." https://github.com/henriquemiranda/yambopy. Accessed 20 Oct. 2021.

among them are: cutoffs on Kohn-Sham empty states B and plane wave expansions G, used in the evaluation of the dielectric function and self energy. There is a strong interdependence among these two parameters, i.e. when one converges B and then converges G, for example, the parameter B is found to be once more out of convergence. So one has to re-converge it for the latter converged value of G. Significant improvement has been encoded in the algorithmic convergence procedures, in order to address the interdependence problem on the one hand and maintain a simple convergence logic on the other hand. The algorithm still converges one parameter at the time, as reported for the previous version of the plugin in deliverable D5.3, but now we also exploit more effectively the overall knowledge of all previous calculations, considered in the 2D space. Each parameter optimization is performed using known fitting functions, e.g. A/x + b, to analytically use optimization methods (such as the Newton algorithm) in order to determine the next points to be explored in the parameters space. This allows the workflow to speed up the optimization procedure, taking explicitly into account the interdependence. Fig. 14 shows an example of such convergences, i.e. the calculation of GW corrections in 2D hBN.

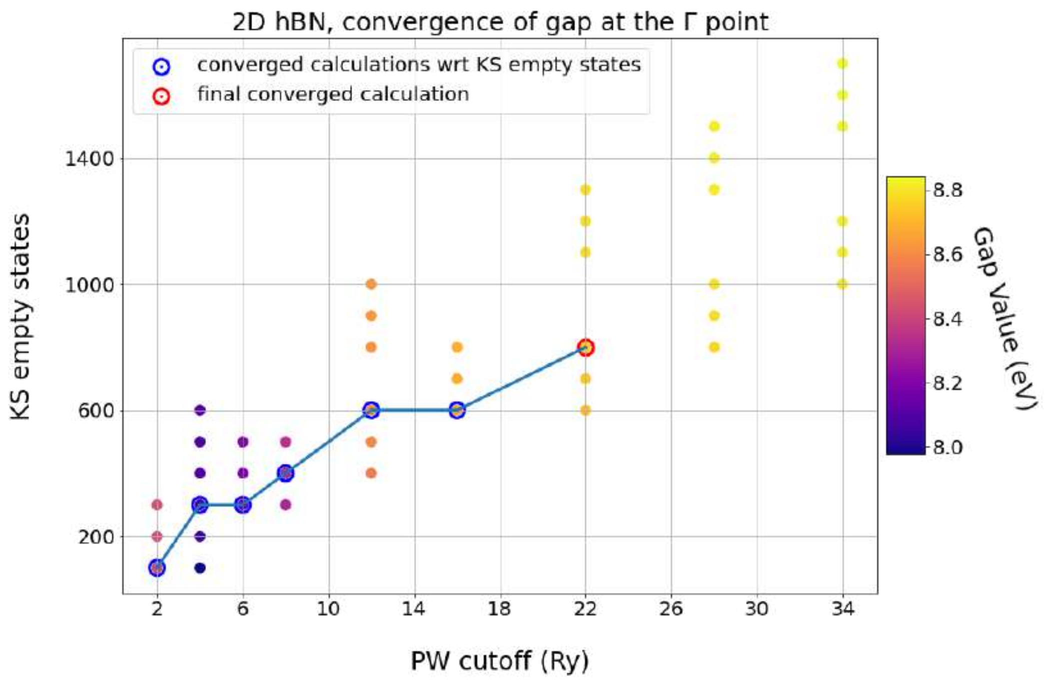


Fig. 14: Example of automatic GW convergence using the newly developed convergence algorithm. The colorbar indicates the value of the computed band gap in eV. The red circle indicates the cheapest fully converged calculation. We observe that the convergence is achieved by performing calculations along parallel lines in the perpendicular direction with respect to the PW cutoff used for the response function (horizontal axis). Note that here we considered 2D hBN, and we have not performed the k-point mesh convergence, as it is a decoupled parameter with respect to the two objects of the convergence study.

BSE convergence

The integration of the yambopy layer, as already mentioned, allowed us to include in the plugin support for Bethe-Salpeter Equation (BSE) calculations. This in turn allows the `YamboConvergence` workflow to perform automatic convergence of excitonic eigenvalues with respect to BSE parameters. Indeed, the quantities to be converged are just provided to the `YamboConvergence` workflow from the `YamboWorkflow` output nodes. Here the most critical parameter is the k-point mesh sampling of the Brillouin Zone.

8 AiiDA-Fleur workflows

8.1 Improvements to the Fleur XML file parsers

Since the last report, a large effort has been undertaken to refactor and improve the file parsers for the Fleur input and output. The Fleur code uses XML files for input and output, from which the information is parsed for the `aiida-fleur` plugin. Until recently, the parsers and methods related to the XML files posed a major source of maintenance for the plugin, since the location of almost all important information was hard-coded in the aiida-fleur plugin and had to be adjusted for newer versions of the Fleur code. This meant that keeping the plugin compatible with multiple versions of Fleur was difficult and required a lot of work.

With the improved file parsers, all information needed in the methods related to the XML files is now directly extracted from the XML Schema files corresponding to the file version of the used code and is stored as an intermediate representation in python dictionaries. This representation enables extraction of information from and modification of the XML files at a higher level, hiding the concrete structure of the XML files with convenience methods. Only with this infrastructure and related improvements to the file parsers as a whole can the current version of the aiida-fleur plugin handle inputs and outputs from the MAX5 release and previous versions of the Fleur code at the same time, since large changes to the input files occurred in this release.

The parts of the improved file parsers not directly related to AiiDA were moved to an independent package called `masci-tools` [12] to make them available outside the context of AiiDA.

8.2 DFT+U calculations

The `aiida-fleur` package previously had no explicit support for performing calculations with the DFT+U method, which aims to improve the description of materials with strongly correlated electrons like e.g. 4f-Lanthanides. The necessary logic for copying the additional files for these calculations with the Fleur was added to the `FleurCalculation`. Furthermore, the `FleurSCFWorkChain` was improved to ensure the correct convergence of DFT+U calculations and methods for easily creating/modifying the files representing the DFT+U density matrix were added.

In addition, a new workchain named `FleurOrbControlWorkChain` was implemented (Fig. 15). This workchain solves the common problem of DFT+U calculations converging into local minima of the total energy by starting multiple calculations with different starting configurations and converging each of them independently using the `FleurSCFWorkChain`.

[12] "masci-tools - Github" https://github.com/JuDFTteam/masci-tools

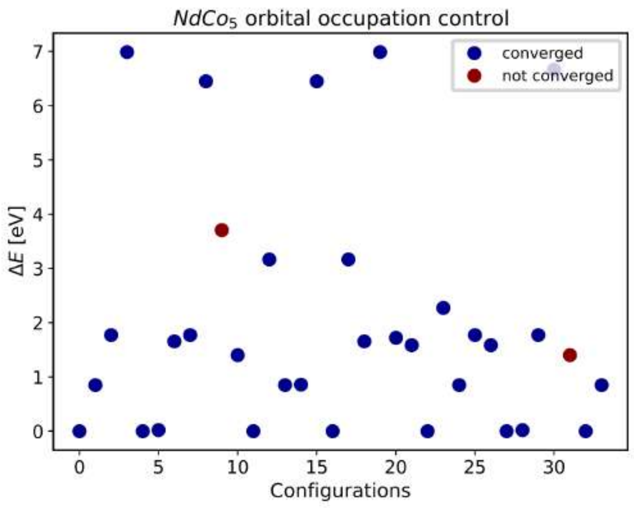


Fig. 15: Example result from a successful run of the FleurOrbControlWorkChain for the compound $NdCo_5$. Each dot shows the final total energy of a given initial starting configuration.

8.3 Band Structure/Density of State workflow

A new workchain for calculating both the density of state and band structure of materials was introduced. It is called `FleurBandDOSWorkChain` and makes use of the introduction of a new HDF5 output file of the Fleur code named `banddos.hdf`. This file contains not only the information necessary for the total density of state and band structure, but also various weights allowing the creation of weighted band structures and projected DOS plots. Such weights currently include the projection of the states or the DOS to the atomic sites, their orbital decomposition, the vacuum contributions, contributions to MCD spectra or information needed for the "backfolding" of the band structure.

The inputs of this workchain are designed to allow the easy manipulation of the used k-point meshes or k-point paths for DOS and band structure calculations respectively. The k-point path can either be calculated using `seekpath`, specified via a list of the high-symmetry points to include or given explicitly as a `KPointsData` input. DOS calculations can also use the explicit k-point input to specify the used mesh. The workchain in addition produces an output in the form of the `BandsData` type for band structures and `XyData` for DOS calculations. This is to prepare this workchain for usage in the extension of the common workflows to these kinds of calculations.

8.4 Summary of aiida-fleur workflows

In summary, the aiida-fleur package currently contains the following workflows to facilitate easy calculations of complex properties in a high-throughput environment. Some of them are already interfaced to the common workflows initiative outlined above.

- SCF workflow: basic task of achieving a self-consistent density
- Equation of state (EOS) workflow: determination of the optimal lattice constant and related properties
- Structure optimization Base workchain: low level structure optimization
- Structure optimization workchain: more advanced structural relaxations

- DOS/Band workflow: as described in detail above
- Orbital occupation control workflow: detailed control of the LDA+U method
- Initial core-level shifts workflow: corel-level shifts in the initial state approximation
- Core-hole workflow: core-level shifts by transition states
- Spin-Spiral Dispersion workchain: spin-spirals using the magnetic force theorem
- Dzyaloshinskii–Moriya Interaction energy workchain: DMI effects in spin-spirals
- Magnetic Anisotropy Energy workflow: MAE from the magnetic force theorem
- Spin-Spiral Dispersion Converge workchain: fully self-consistent spin-spiral dispersions
- Magnetic Anisotropy Energy Converge workchain: MAE by fully self-consistent calculations
- Create Magnetic Film workchain: high level workflows to construct relaxed magnetic film structures

9 AiiDA-CP2K workflows

The `aiida-cp2k` package[13] contains the basic features required to interact with CP2K code through the AiiDA interface. As such, we try to keep it as simple and as small as possible. Mostly, we focus on fixing bugs and implementing the critically required features found to be useful by other higher-level workflows.

Since the time of the previous report we have added optional support for the Gaussian data types available in `aiida-gaussian-datatypes`[14] package. Users can decide whether to use the "standard way" of specifying pseudos/basis sets in the input dictionary or to let the input plugin automatically pick them from the input nodes. To be compatible with the common workflows project described in section 2, we added the possibility to use the standard AiiDA `KpointsData` class as the input for k-points. Also, we added a new output node to `Cp2kBaseWorkChain` that contains the input dictionary used for the final calculation step - a necessary feature for the restarts. Finally, we added a new parser that is based on the cp2k-output-tools project[15] developed by the CP2K team.

To participate in the AiiDA common workflows initiative, we have implemented a `Cp2kCommonRelaxWorkChain` that provides three levels of accuracy: fast, moderate and precise. Additionally, the workflow supports different relaxation types, electronic types, and spin types. The supported structure relaxation types are 'none' (means no relaxation), 'positions' (only atomic coordinates), 'positions_cell' (atomic coordinates and the cell size). The 'electronic_type' argument supports the values 'metal' and 'insulator'. For an 'insulator', the calculation is performed using the orbital transformation method with fixed occupations. The default total magnetic moment is equal to 0 for an even number and 1 for an odd number of electrons. For a 'metal' a diagonalization with Fermi-Dirac smearing is used at an electronic temperature of 500 K and 20 molecular orbitals are added for each spin. In case smearing is employed, the total magnetization is flexible. The 'spin_type' argument supports the values 'none' and 'collinear'. In the case of the 'collinear' option, the spin-polarized calculations are enabled. Additionally, the user can specify the

[13] "The CP2K plugin for the AiiDA workflow and provenance engine.." https://github.com/aiidateam/aiida-cp2k. Accessed 21 Oct. 2021.
[14] "dev-zero/aiida-gaussian-datatypes - GitHub." https://github.com/dev-zero/aiida-gaussian-datatypes. Accessed 21 Oct. 2021.
[15] "cp2k-output-tools - GitHub." https://github.com/cp2k/cp2k-output-tools. Accessed 21 Oct. 2021.

'magnetization_per_site', which would be ignored in the 'none' case. If the user provides the 'magnetization_per_site', the total multiplicity is derived from it.

In the next two sections, we provide an overview of AiiDA-CP2K workflows implemented and used in the LSMO laboratory at EPFL and in the nanotech@surfaces laboratory at Empa.

9.1 Workflows implemented in the LSMO laboratory at EPFL

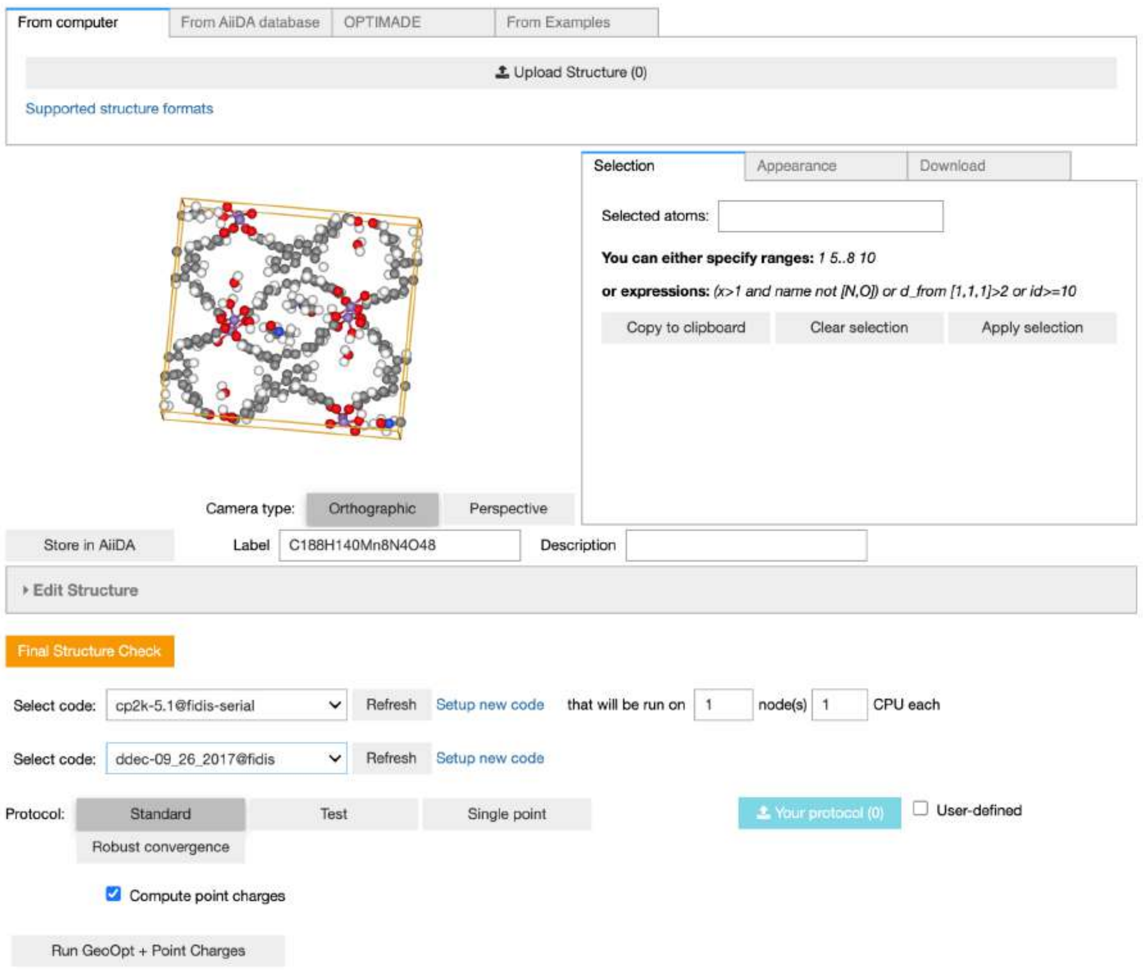
All the workflows are available within the aiida-lsmo package[16]. They are optimized to work with porous materials such as covalent organic frameworks, metal organic frameworks, and zeolites. Below we list the available workflows and describe their functionality.

`Cp2kBindingEnergyWorkChain`: this workflow submits a `Cp2kBaseWorkChain` for a molecule in periodic structure, first optimizing the geometry of the molecule and later computing the corrected interaction energy taking into account the basis set superposition error (BSSE). The workchain is very similar to the `Cp2kMultistageWorkChain` as it first does the structure optimization using the settings provided in a protocol file.

`Cp2kMultistageWorkChain`: this workflow is meant to optimize the structure combining geometry optimization, cell optimization and molecular dynamics. The exact sequence is defined in the YAML protocol files that are shipped in the package. In case the standard protocols do not match the users requirement, they can provide their own protocol file as a separate input. Additionally, users can specify the minimal cell size: if the provided cell is too small, it will be increased automatically to fulfill the provided criterion.

`Cp2kMultistageDdecWorkChain`: this workflow performs the structure optimization and then computes DDEC points charges derived from the electronic charge density computed by CP2K. The output of the workflow is the crystal structure in CIF format with point charges assigned to atoms. It is a prerequisite for subsequent simulations using classical molecular dynamics or Monte Carlo methods. In addition we provide a visual interface to that work chain in the form of AiiDAlab application to give access to the simulations to non-experts (see Fig. 16).

[16] "AiiDA workflows for the LSMO laboratory at EPFL - GitHub." https://github.com/lsmo-epfl/aiida-lsmo. Accessed 21 Oct. 2021.


From computer | From AiiDA database | OPTIMADE | From Examples

Upload Structure (0)

Supported structure formats

Selection | Appearance | Download

Selected atoms:

You can either specify ranges: *1 5..8 10*

or expressions: *(x>1 and name not [N,O]) or d_from [1,1,1]>2 or id>=10*

Copy to clipboard | Clear selection | Apply selection

Camera type: Orthographic | Perspective

Store in AiiDA

Label: C188H140Mn8N4O48

Description:

Edit Structure

Final Structure Check

Select code: cp2k-5.1@fidis-serial | Refresh | Setup new code | that will be run on 1 node(s) 1 CPU each

Select code: ddec-09_26_2017@fidis | Refresh | Setup new code

Protocol: Standard | Test | Single point | Robust convergence

Your protocol (0) | User-defined

☑ Compute point charges

Run GeoOpt + Point Charges

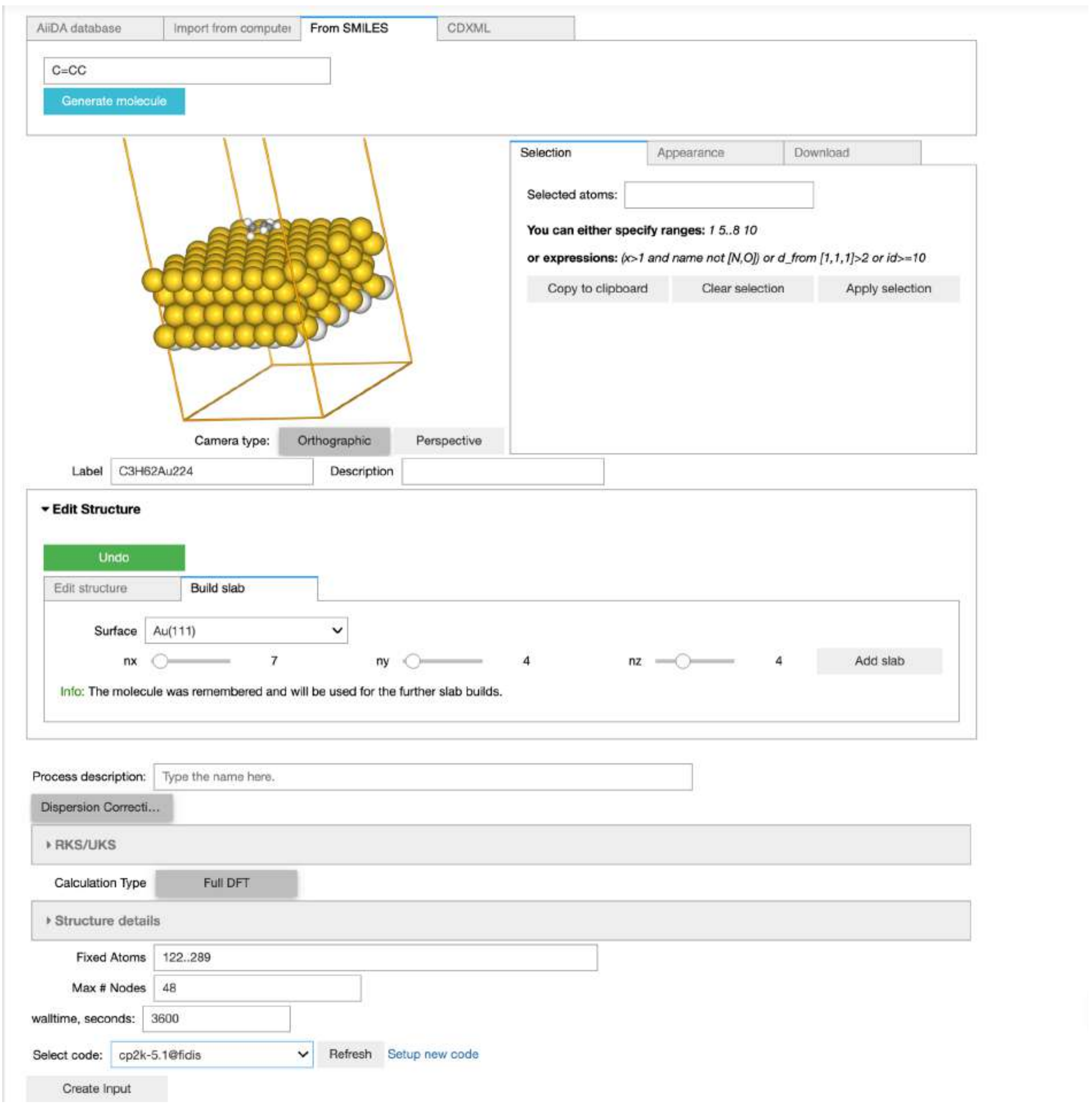
Fig. 16: Example of the LSMO AiiDAlab application that helps setup the Cp2kMultistageDdecWorkChain. Additionally, it provides the possibility to perform the structure check for the Metal-Organic Frameworks (MOFs).

`Cp2kPhonopyWorkChain`. The workflow computes phonon frequencies using CP2K and Phonopy. It takes as an input the structure and reference CP2K calculation as the optional parameter. In case the latter is not provided, the workflow searches for an ancestor calculation of the input structure. Additionally, the user can specify the calculations to run in 'serial' (default) mode to save computational resources or 'parallel' mode to reduce the computation time.

9.2 Workflows implemented in the nanotech@surfaces laboratory at Empa

All the workflows are available within the aiida-nanotech-empa package[17]. The workflows are adapted to work with bulk materials (such as metals or intermetallic compounds), slabs, molecules, and molecules adsorbed on material surfaces. All the workflows described below are available through the user-friendly On-Surface Chemistry application available on AiiDAlab (see Fig. 17).

[17] "AiiDA workflows for the nanotech@surfaces group at Empa - GitHub." https://github.com/nanotech-empa/aiida-nanotech-empa. Accessed 21 Oct. 2021.



C=CC

Selected atoms:

You can either specify ranges: *1 5..8 10*

or expressions: *(x>1 and name not [N,O]) or d_from [1,1,1]>2 or id>=10*

Label: C3H62Au224 Description:

Surface: Au(111)

nx 7 ny 4 nz 4

Info: The molecule was remembered and will be used for the further slab builds.

Process description: Type the name here.

Calculation Type: Full DFT

Fixed Atoms: 122..289

Max # Nodes: 48

walltime, seconds: 3600

Select code: cp2k-5.1@fidis

Fig. 17: Example of the On-Surface Chemistry AiiDAlab application that helps to set up geometry optimization with CP2K. The application also provides the possibility to generate the molecule from SMILES, build a metallic slab and place the molecule on it.

`Cp2kBulkOptWorkChain`: this workflow allows to perform geometry optimization of slabs with the possibility for fine-tuning through the input parameters. Users can specify whether they want to perform cell optimisation as well and if they want to keep the cell symmetry along the optimization process. Additionally, users can provide a list of fixed atoms that won't be moved during the optimization process. Users can switch on and off the vdW interactions, set the total charge of the system, and provide the total multiplicity as well as the magnetization per site.

`Cp2kMoleculeGwWorkChain`: this workflow allows to perform GW calculations using CP2K. It supports the following protocols: 'gapw_std' (Gaussian Augmented-Plane Waves method with standard settings), 'gapw_hq' (Gaussian Augmented-Plane Waves method with high-precision settings), 'gpw_std' (Gaussian-Plane Waves method with standard settings). Additionally, users can specify whether to run the image charge correction calculation. Finally, the possibility to provide total multiplicite and magnetization per site is also there.

`Cp2kSlabOptWorkChain`: this workflow is very similar to the `Cp2kBulkOptWorkChain`, but it doesn't provide the possibility to optimize the unit cell.

10 AiiDA-BigDFT workflows

The BigDFT AiiDA plugin, developed during the course of the MAX project was completed and improved since the last report. The plugin has been kept up to date with the latest developments of both AiiDA and PyBigDFT. Compatibility with AiiDA 2.0 has been already preliminarily tested, and convenience features, such as common crash reasons (memory, walltime, common BigDFT errors...) have been added, to allow easy identification of the root of the issue, or even automatic restarting of failed jobs.

10.1 Post processing and complex workloads

The size and complexity of the workloads used in the current studies using BigDFT might make it difficult to transfer large log files and output data of intermediary jobs to the AiiDA controlling system. AiiDA fully supports handling of large data residing only on the remote system, which led to build upon such features to address the complex post processing needs for some workflows.

A step for post-processing tasks was added in the BigDFT plugin, in order to run dedicated scripts on the remote system after a successful BigDFT job, using the data of this previous run as input.

The need for even more flexibility for such pre- or post-processing tasks executed on the remote system has led to the implementation inside PyBigDFT of a remote runner construct, in order for complex tasks to be performed around one or several executions of the BigDFT main code, without any data being exchanged between the controlling node and the supercomputer/remote system. This construction can then be seen from the AiiDA point of view as a remote code by itself and will be treated as such in a new version of the plugin. In this case the called code is no longer just the BigDFT executable but a full script using PyBigDFT to process the input and output data, run the code and output the data for AiiDA to gather and collect to the controlling node. Some sub-tasks may be generated and submitted to the batch scheduler by this remote runner in order to parallelize computation, as PyBigDFT is able to generate submission scripts by itself.

10.2 Common Workflows with BigDFT, and Sirius use case

As the BigDFT plugin was designed during the course of this project, integration with the AiiDA common workflows initiative has been simplified, as BigDFT-AiiDA workflows were designed from the beginning to comply with the interfaces of the project. The NLCC Pseudo potentials available in PyBigDFT are used by default (pending the creation of a proper AiiDA family) for the supported atoms. The relaxation work chain makes use of the relaxation capacities ithin BigDFT, and is actually a single computation, and the PDOS computation will be handled the same way in the next release.

Sirius integration in BigDFT has been part of the recent developments, and the common workflow interface has been used to validate the implementation and its results. Indeed, the pseudopotentials used by BigDFT (HGH format) are available as UTF format, and Sirius provides converting tools to translate these UTF pseudopotentials as Json. It has then been possible to use the common workflows interface to generate BigDFT and BigDFT+Sirius versions of relaxation experiments (Fig. 18), to verify that the results are consistent, and to tune the parameters in order to provide the same results with both computation engines. Sirius integration is an important step, as it will lead to support of non-orthorhombic cells in a coming release of BigDFT.

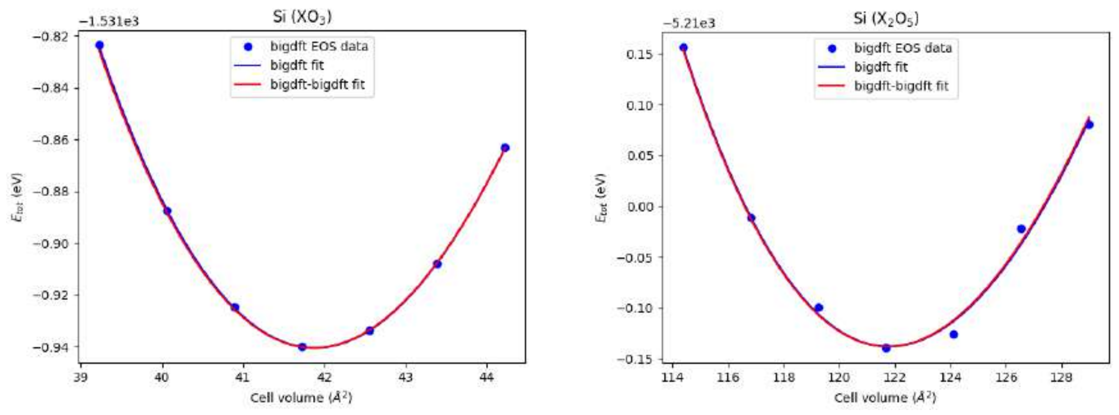


Fig. 18: example of fitting two EOS workchains with the common workflow interface, showing that BigDFT alone (blue) fits perfectly with the results of the computations using BigDFT+Sirius (red), when the same pseudopotentials are being used. Resulting energies have been shown to be the same at 6 significant digits in these tests.

11 Conclusions

All MaX flagship codes are now tightly integrated with AiiDA: they not only provide plugins to create the inputs and parse the outputs of the codes via the AiiDA workflow engine, but now they also provide a large number of workflows covering a broad set of materials properties (bands structures, DOS, PDOS, optical properties, DFT+U, GW and automatic convergence of parameters, …). In addition, workflows help significantly improve the robustness of the codes, reducing the failure rate of the simulations, which in turn also means more reliable simulations and a reduction in the overall costs of the simulations on HPC resources.

Furthermore, MaX already achieved one of its key goals: providing a common robust interface to compute materials properties, independent of the underlying quantum code. This has been achieved in the form of common interfaces to AiiDA workflows for single-point and relaxation calculations, where 11 different codes (with very different basis sets and algorithms, and including all MaX codes that perform total-energy calculations) can now be driven with the very same interface to compute the relaxed cells and atoms, as well as forces and stresses. These workflows, in addition, provide robust recipes that implement error correction and recovery in case of code failures, as well as automatic parameter choice for the inputs of the simulations. We expect that these common workflow interfaces will be the building blocks for fully code-agnostic workflows (e.g., we already implemented general workflows to compute dissociation curves and the equation of state of materials, and plan to implement more, e.g. to compute phonon dispersions, stability, energetics of defects, open-circuit voltages of battery materials, ..) and that they will be complemented in the future by even more common workflow interfaces for advanced properties (band structures, DOS, dielectric constant, Born effective charges, …).