



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



AiiDA Lab implementation of IR spectrum calculations for carbon based nanomaterials

Semester Thesis

Diego Renner

`diego.renner@student.ethz.ch`

Computational Science and Engineering
Seminar of Applied Mathematics
ETH Zürich

Supervisors:

Dr. C. A. Pignedoli – `carlo.pignedoli@empa.ch`

August 25, 2020

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 2 | Physical Background and Computational Realization | 3 |
| 2.1 | Physical Background | 3 |
| 2.1.1 | Normal modes of vibration | 5 |
| 2.2 | Computational Realization | 6 |
| 2.2.1 | Full DFT approach | 6 |
| 2.2.2 | Mixed DFT | 7 |
| 2.2.3 | DFTB | 9 |
| 2.2.4 | Parallelization of Vibrational Analysis | 9 |
| 3 | Setup and Application Structure | 10 |
| 3.1 | AiiDa | 10 |
| 3.2 | AiiDa Lab | 10 |
| 3.3 | Application Structure | 11 |
| 3.3.1 | GUI | 12 |
| 3.3.2 | Submit Button | 16 |
| 3.3.3 | Workflow | 16 |
| 3.3.4 | Get CP2K input | 17 |
| 3.3.5 | Gas phase and on slab version | 18 |
| 3.3.6 | Visualization | 19 |
| 4 | Example | 21 |
| 4.1 | Submission | 21 |
| 4.2 | Visualizing | 24 |
| 5 | Future work | 27 |
| 5.1 | Raman spectroscopy | 27 |

| | |
|--|-----------|
| CONTENTS | ii |
| 5.2 Application structure optimisation | 27 |
| Bibliography | 29 |

Introduction

This research project was conducted at the Swiss Federal Laboratories for Materials Science and Technology in the nanotech@surfaces laboratory. Here in 2010 a technique to fabricate atomically precise carbon based 1D nanostructures was demonstrated. The technique is based on self-assembly of specifically designed molecular precursors on noble metal substrates [1]. The method has since then demonstrated high flexibility and diverse new nanomaterials with particular focus on graphene nanoribbons (GNRs) have been fabricated and characterized by using it [2].

To understand or predict the fundamental electronic properties of the GNRs, simulations mainly relying on Density Functional Theory [3] are conducted in parallel to the experiments in the laboratory. The ability to predict gas phase properties of GNRs and to reproduce microscopy and spectroscopy features for the nanomaterials supported on the noble metals, is a prerequisite to drive research in this field. Thus the fabrication and analysis techniques established in the laboratory in the last few years have been refined considerably to achieve a "standard" simulation procedure. At the present, the "human factor" in highly repetitive tasks, such as preparation of inputs for calculations, monitoring of running calculations, retrieving and analysing of simulation data, is a considerable limiting factor to the further discovery of new nanomaterials.

Since 2018 the nanotech@surfaces laboratory has been developing computational workflows within the AiiDA platform [4], in collaboration with EPFL researchers, in order to considerably reduce the possibility of errors in dealing with repetitive simulation tasks and the amount of human input needed. The work is conducted within the NCCR MARVEL project financed by the Swiss National Research Foundation. In working towards this goal an automated infrastructure was created in which GNRs can be analyzed, archived and managed based on a chemical sketch of their structure [5, 6]. The infrastructure is comprised of modular "Apps" which allows external contributors to easily add on their own work. In order to be standardized and accessible the infrastructure is run in a Docker [7] container on a virtual machine that is hosted by CSCS. From there other dockers in the HPC infrastructure can be accessed. The AiiDa platform is

setup in the Docker on the virtual machine and an interface to the framework is given using AiiDa lab. The AiiDa lab interface is based on Jupyter Notebooks [8] and allows app developers to install and create apps that grant an easier access to AiiDa. To create new apps Jupyter Notebooks are employed as well.

At the moment with the help of this infrastructure one can compute electronic properties of ideal nanoribbons in gas phase, and adsorbed on a substrate and simulate scanning probe experiments to unravel the atomistic details of the different stages of the fabrication process. Also chemical reactions on a surface can be studied with apps that rely on the Nudged Elastic Band Method [9] for transition state search. While these techniques are quite efficient at complementing the experiments to understand the complex process of synthesis, absorption spectra such as IR and Raman, that are also employed experimentally to unequivocally identify fingerprints of reactants, products and intermediate products, still belong to the class of "non-automated" calculations. The goal of this project is to add a solution for efficiently computing infrared spectra not only for molecules in gas phase but also for molecules adsorbed on a noble metal substrates. Attention will be devoted to the possibility of using different levels of theory in the simulation, from approaches relying on a full DFT treatment of the atomistic model, to approaches meant for a rapid screening of materials' properties where different theoretical backgrounds are employed to treat the adsorbate system and the substrate.

This report is organized as follows: In chapter two it is described how to use the change in polarization as well as the modes of a structure to compute an IR spectra. Furthermore the computational methods based on atomistic approaches and DFT used to do these calculation are described as well as the manner in which CP2K parallelizes them. Chapter three goes over the structure of the application that was developed in this project. Here starting from the AiiDa framework, over the AiiDa lab interface to the specific building block of the application itself all components are described. In chapter 4 an example is demonstrated on how to submit an IR computation job and visualize the results using the application. Finally in chapter 5 there is an outlook on future work. Specifically the Raman spectroscopy method that could be added as a feature to the application is mentioned and optimizations of the code structure are discussed.

Physical Background and Computational Realization

The application created in this project serves as an interface to specific parallelized CP2K calculations. In the following sections the physics on which these calculations are based will be introduced as well as the computational methods applied to realize them. This will be split into a short introduction to IR spectroscopy and an overview of the methods DFT, mixed DFT and DFTB which can be applied using the application. At the end of this chapter there will be a description of the parallelization scheme applied by CP2K to solve the IR spectroscopy problem.

2.1 Physical Background

In infrared spectroscopy, light of wavelengths in the range $0.78 \mu m - 300 \mu m$ is shed on the sample system and peaks of absorption are measured. Light absorption will be governed by the interaction with varying polarization of the sample. At wavelengths typical for IR, light is interacting with the vibrational modes of the system. More precisely, if the spectrum of possible normal modes of vibration for the target system is known, IR absorption will be possible only for vibrational modes that induce a variation of the systems polarization. In an harmonic approximation, IR spectras can be computed knowing the normal modes of vibration of the modeled system and the variation of polarization associated to each specific vibrational mode. To compute the normal modes of vibration of a system, within the harmonic approximation, we can rely on finite difference methods or on linear response methods [10]. Within this research project we will be dealing with the finite differences approach to compute normal modes of vibration. Since in general, for the case of molecular systems adsorbed on a substrate, we deal with atomistic models with imposed periodic boundary conditions, to compute the variation of polarization associated with a specific vibrational mode, one has to rely on modern theory of polarization [11, 12]. Without entering into the de-

tails of Berry's phase and the modern theory of polarization, it is instructive to recall an intuitive example where effective charges are associated to atoms in a crystal for a specific vibrational mode and from these effective charges the variation of polarization for each specific phonon can be computed. We recall that, in the case of vibrational modes not associated with a variation of the polarization of the sample, no IR features are observed for that specific mode.

As described in [11] the change in polarization due to a transformation of the Hamiltonian H is given by

$$\Delta P = -\frac{ifq_e}{8\pi^3} \sum_{n=1}^M \int_{BZ} d\mathbf{k} \int_0^1 d\lambda [\langle \partial u_{\mathbf{k}n}^{(\lambda)} / \partial k_\alpha | \partial u_{\mathbf{k}n}^{(\lambda)} / \partial \lambda \rangle - \langle \partial u_{\mathbf{k}n}^{(\lambda)} / \partial \lambda | \partial k_\alpha | \partial u_{\mathbf{k}n}^{(\lambda)} / \partial k_\alpha \rangle]. \quad (2.1)$$

Here f is the number of states in the valence band, q_e is the electron charge and M is the number of occupied bands. The integral over \mathbf{k} extends over any primitive cell in reciprocal space and $\partial/\partial k_\alpha$ is the derivative along the Cartesian coordinate α . λ is the parametrization of the transformation of H so that at $\lambda = 0$ and $\lambda = 1$ is the initial and final state of the transformation of H respectively. $u_{\mathbf{k}n}^{(\lambda)}$ are cell periodic functions with phases chosen so that they are analytic in both \mathbf{k} and n .

Introducing the Born effective charge tensor for the i th atom in the unit cell of volume V , ΔP can be written as

$$\Delta P = \frac{q_e}{V} \sum_{i=1}^N Z_i^* \cdot \Delta x_i. \quad (2.2)$$

The Born effective charge tensor gives the macroscopic electric response of a crystal due the displacement of an atom Δx_i . Hence we can compute our change in polarization with the displacement given by a vibrational mode using equation 2.2 [13].

We can go from 2.1 to 2.2 using Wannier functions. Wannier functions are an alternate representation to the extended Bloch eigenfunctions for the electronic structure of periodic solids. They can be constructed via a unitary transformation of the Bloch states belonging to an isolated band or to a composite group of bands. Wannier functions are useful for describing states in the valence bands since they provide an intuitive, "chemical-like" localized picture of bonding and dielectric properties of insulators [14].

As mentioned to compute the IR spectra the normal modes of vibration of a system have to be computed. Once the vibrational modes are known the change in polarization along a specific mode can be computed with the aforementioned method. A description on how to compute the modes of a structure is given in the next section.

2.1.1 Normal modes of vibration

To compute the normal modes of a given structure, the first step is to determine the equilibrium geometry of the system with respect to the atomic coordinates $(x_{1,0}, x_{2,0}, \dots, x_{3N,0})$. Here $x_{i,0}$ refers to one coordinate of one atom when in the aforementioned minimum. If we do not have this assumption we will see later that the analysis would become unfeasible. For simplicity we also introduce mass weighted coordinates:

$$\xi_i = \sqrt{m_i}(x_i - x_{i,0}) \quad \forall \quad i = 1, \dots, N. \quad (2.3)$$

Where we are using m_i , x_i as the mass and position respectively of one coordinate of one atom of our structure. Using this convention we will expand the potential up to the second order around the minimum. This is reasonable since around the minimum the potential will behave quadratically. The Lagrangian of our system can then be written as:

$$\mathcal{L} = \frac{1}{2} \sum_{i=1}^{3N} \dot{\xi}_i^2 - \frac{1}{2} \sum_{i,j=1}^{3N} \left(\frac{\partial^2 V}{\partial \xi_i \partial \xi_j} \right)_0 \xi_i \xi_j \quad (2.4)$$

Applying this to the equations of motion and setting the mass-weighted Hessian to k_{ij} gives:

$$\ddot{\xi} + \sum_{k=1}^{3N} k_{ik} \xi_k = 0. \quad (2.5)$$

Due to the form of the equation we choose $A_k e^{i\omega t}$ as an ansatz. Substituting this back into 2.5 we get $3N$ equations:

$$\sum_{k=1}^{3N} (-\omega^2 \delta_{ik} + k_{ik}) A_k = 0. \quad (2.6)$$

For this linear system of equations to have a non-trivial solution we need the determinant of coefficients to be zero:

$$|k_{ik} - \omega^2 \delta_{ik}| = 0. \quad (2.7)$$

Solving this equation for an ω gives us the frequency of an eigenmode. This frequency can be substituted back into 2.6 in order to find a vector corresponding to the eigenmode. The eigenmodes and the frequency are finally given by:

$$\Theta_k(t) = b_k \cos(\omega_k t + \phi_k) \quad (2.8)$$

$$\xi_j = \sum_{k=1}^{3N} A_{jk} \Theta_k(t). \quad (2.9)$$

This tells us that our structure can vibrate along an eigenmode A with the frequency ω .

2.2 Computational Realization

To obtain the normal modes of vibration from an atomistic model, we need to compute the dynamical matrix (also referred to as the mass-weighted Hessian). This is usually a computationally expensive task but it can be efficiently parallelized. Two approaches are commonly used to derive the dynamical matrix: linear response [10] and finite differences. In this work we rely on the finite differences approach to compute vibrational modes as implemented in the CP2K code. In the finite differences approach, at each step one atom of the simulation cell is displaced from its minimum position in either x, y or z direction. Then the energy for this configuration is computed. Using this information one can apply finite differences to gather information about the mass-weighted Hessian. Finally diagonalizing the dynamical matrix gives us the eigenmodes. For each new configuration defined by the displacement of an atom, the energy associated to the perturbed system has to be computed. We recall here that our context is “molecules on metallic substrates”. Atomistic models that realistically represent this class of systems can easily require more than a thousand atoms thus, a brute force treatment of the problem at a high level of theory is not always the first choice. In this work we provide the user with an interface to select three possible levels of theory for their calculations: full DFT, DFT (for the adsorbate) combined with an embedded atom model (EAM) for the substrate and DFTB (for the adsorbate) combined with EAM [15, 16]. We describe briefly below the main differences between the three approaches.

2.2.1 Full DFT approach

The derivation of DFT goes beyond the aim of this short report. We just recall here that, due to the Hohenberg–Kohn theorem, we can derive the ground state properties of a system of interacting electrons subject to an external potential (in our case the ionic potential from the nuclei of the atoms in our model) if the ground state electron charge density is known [3]. The success of DFT when related to problems in materials science relies on the one hand on the efficiency and accuracy with which approximations to the exact Kohn-Sham theory are implemented in computer codes and, on the other hand, on the wide spectrum of properties that can be derived from DFT even beyond its formal validity (we refer here, for example, to the calculation of band structures for solids).

The Kohn-Sham ansatz replaces the interacting problem with an auxiliary independent-particle problem. Formally this is reflected in the Kohn-Sham equations:

$$(H_{KS}^\sigma - \epsilon_i^\sigma)\psi_i^\sigma(r) = 0. \quad (2.10)$$

Here ϵ_i are the eigenvalues and H_{KS} is the effective Hamiltonian (in Hartree atomic units)

$$H_{KS}^\sigma(r) = -\frac{1}{2}\nabla^2 + V_{KS}^\sigma(r) \quad (2.11)$$

with

$$V_{KS}^\sigma(r) = V_{ext}(r) + V_{Hartree}(r) + V_{xc}^\sigma(r). \quad (2.12)$$

These equations represent independent-particle equations with a potential that has to be found self-consistently [17]. In many cases it suffices to only take the valence electrons into account when doing the wave function optimization for this problem. This means we can replace the ionic potential by pseudopotentials. In our CP2K calculations we use GTH pseudopotentials [18] and we expand the Kohn-Sham orbitals in a localized basis set of contracted gaussians [19]. Since we typically deal with metal/adsorbate systems, it is important to include corrections to account for dispersive forces. We rely on the Grimme parametrization for Van der Waals interactions [20].

2.2.2 Mixed DFT

Since a full DFT computation is quite expensive to do because of the wave function optimization for each configuration one often uses a mix of DFT and empirical force fields for larger systems. In our example this can be used when simulating the slab together with the molecule. The molecule would then be treated with a full DFT calculation while the energy coming from the slab would be computed via an empirical force field. A schematic description of this approach can be seen in figure 2.1

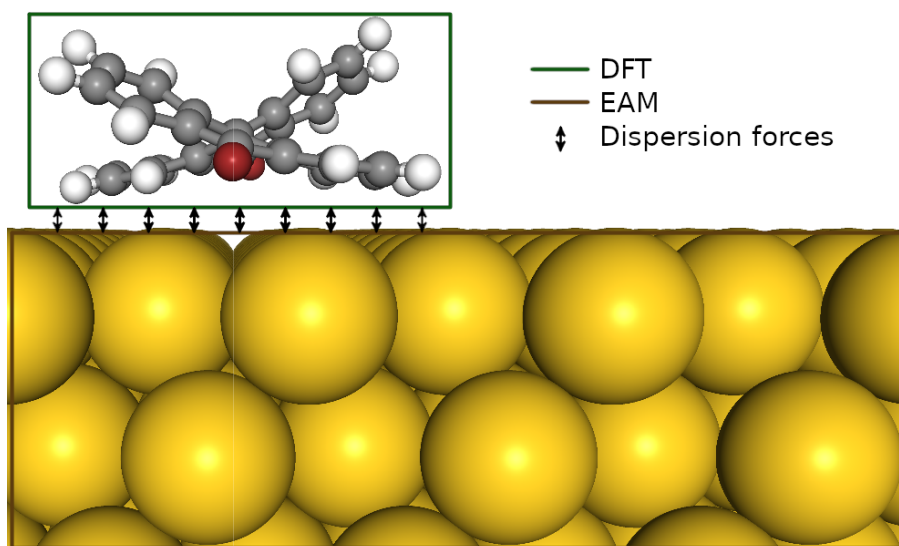


Figure 2.1: Since the slab is close to static it can be treated with an empirical force field. The molecules complex dynamics require a more accurate approach and are therefore treated with a DFT calculation. The interaction between slab and molecule are taken into account using dispersion forces.

In our case we assume that the molecule is only interacting via physisorption with the surface of the slab. Therefor we only use non-bonding potentials to describe the interactions between the slab and itself and between the slab and the molecule. Depending on the atoms in question we either use a general potential or a Leonard Jones potential. An example for the parametrization in the two cases can be seen below.

Code Listing 2.1: This excerpt of a CP2K input file shows the parametrization of a general potential between gold and hydrogen and a Lenard Jones potential between carbon and hydrogen.

```
&GENPOT
atoms Au H
  FUNCTION A*exp(-av*r)
           +B*exp(-ac*r)-C/(r^6)/(1+exp(-20*(r/R-1)))
  VARIABLES r
  PARAMETERS A av B ac C R
  VALUES 0.878363 1.33747 24.594164
           2.206825 32.23516124268186181470 5.84114
  RCUT 15
END GENPOT
LENNARD-JONES
atoms C H
```

```
EPSILON 0.0  
SIGMA 3.166  
RCUT 15  
END LENNARD-JONES
```

2.2.3 DFTB

DFTB is a tight binding approach which is parametrized directly using DFT. It can be used when the system size grows too big for DFT's computational scaling. However even though DFTB has its origin in DFT it is still less accurate. This is due to the fundamental starting point of DFTB being tightly bound electrons and the interactions being treated as perturbations. [16]

2.2.4 Parallelization of Vibrational Analysis

Since the energy for each configuration of the IR calculation can be computed independently it is reasonable to have parallel units taking care of these tasks. In CP2K we can do this by defining the variable NPROC_REP. This is the number of processors used per replica. A replica is one unit that does an energy calculation on it's own. Hence NPROC_REP should be large enough to take care of the wave function optimization used to compute the energy of a configuration. The number of units doing the computations in parallel are given by how many processors the CP2K code is run on overall. CP2K will use as many replicas with NPROC_REP processors as possible until the total number of processors is used up. Furthermore each replica can be internally divided for mixed DFT calculations. Like this the energy stemming from the empirical force field can be computed with only a fraction of the ones used for the entire replica while the rest is used for the wave function optimization. [21]

Setup and Application Structure

The application was built using Jupyter Notebooks and ipywidgets within an AiiDa lab instance. The Jupyter Notebooks were employed to access the AiiDa framework which was then used to submit calculations to the HPC system and retrieve and store the results.

3.1 AiiDa

AiiDa manages data and calculations in order to ensure preservation and searchability. It does this using an acyclic graph in which nodes can be data, calculations, or instructions and their relations are described by vertices. Through the encoding of sequences of calculations as workflows one can operate using the nodes as inputs and outputs. This allows a high level of automation while still keeping track of the provenance of the data and the calculations. Remote computational resources can be setup to be accessed through AiiDa making it an ideal tool for submitting high throughput calculations in an automated and simplified fashion. [4] For this application AiiDa was used to do just this. A GUI allows the users to select the parameters for the calculation. Upon submission a predefined workflow is called that takes the input from the GUI, gathers the information necessary for starting the job and submits it. This calculation with all its input is immediately registered by AiiDa. The same goes for the results once the job has finished executing. Hence the application only has to take care of preparing the input and picking it up at the right place once the job has finished. Later we will see how the AiiDa database can be queried to do this.

3.2 AiiDa Lab

AiiDa lab is designed to reduce the complexity of the AiiDa interface. It does this using the AiiDa lab app. All available apps are listed in a central app registry called AiiDa lab registry. Here there is also the AiiDa lab home app which is the

core application of the service. It provides a Homepage, an App Store, a Linux-like terminal and the interface to access apps installed by the user from the App Store. In the cloud where AiiDa lab is hosted every user runs its own isolated Docker environment with the AiiDa lab docker stack repository containing the docker file used for the AiiDa lab setup. [5]

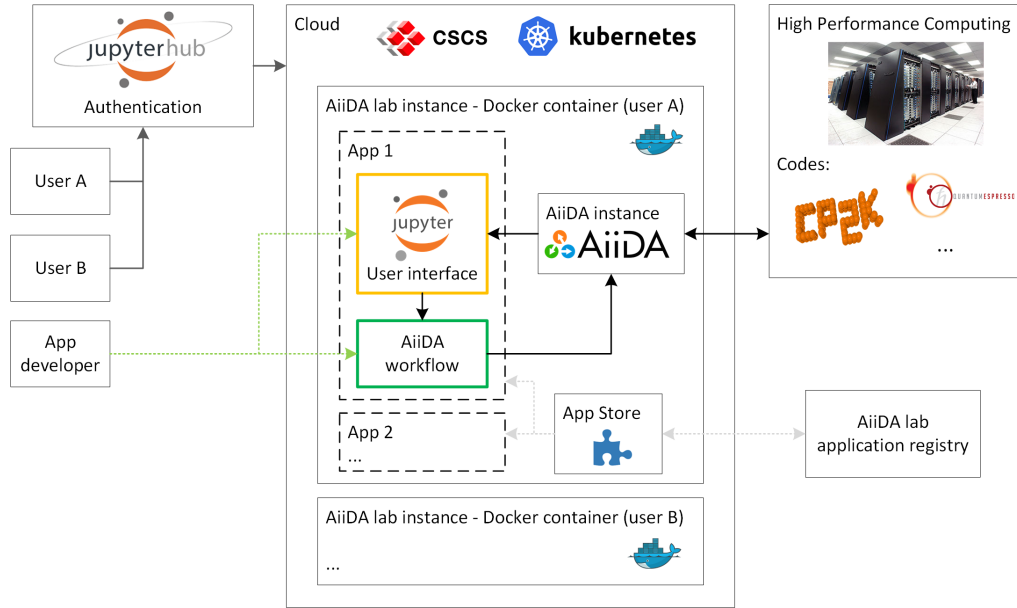


Figure 3.1: On the cloud multiple instances of AiiDa lab are run. They are accessible through a jupyter hub for standard users or directly for App developers. By an AiiDa instance each AiiDa lab instance is connected to the HPC infrastructure. Source: [5]

Using AiiDa lab allows users to focus on the part of the calculation submission process which they are interested in. One can develop Jupyter Notebook based apps which use preexisting workflows to issue instructions to AiiDa, one can design these workflows, or one can use preexisting apps to analyze data that is stored in the AiiDa database. In the project such an app was designed using the AiiDa lab interface and a workflow was adapted to take care of the job submission. Furthermore an app was developed to visualize the results of these calculations.

3.3 Application Structure

The structure of the application can be seen in figure 3.2. The main GUI of the application is used to let the user set the parameters of an IR calculation. These parameters are then gathered and preprocessed by the submit button.

The submit button hands on the workflow and the processed input to the AiiDa framework. Here the workflow is used to create the actual CP2K input file and submit the job on the HPC cluster. There are two versions of the GUI that allow you to run these calculations. One is for a molecule on a slab and the other is for a molecule in gas phase. Finally the results of the computations submitted with one of these two GUIs can be retrieved and visualized using another GUI. The components mentioned here will be more closely described in the next subsections.

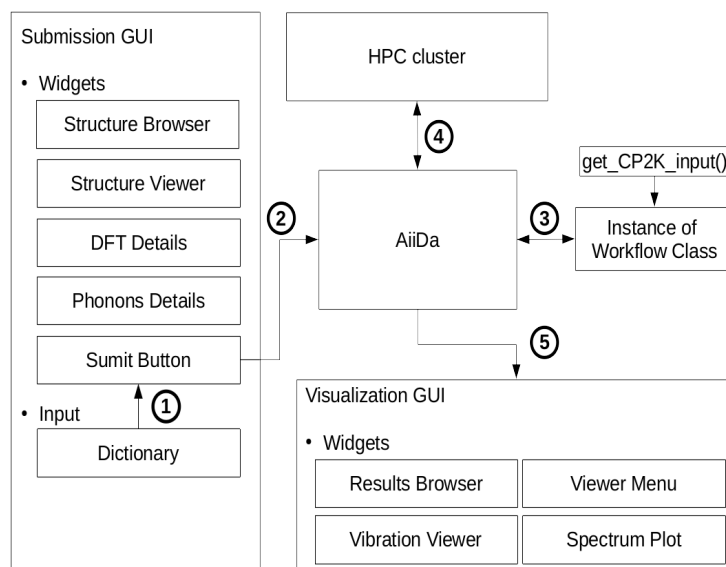


Figure 3.2: 1) The users input is collected and handed off to the submit button widget. 2) The input is parsed and submitted to AiiDa. 3) AiiDa creates an instance of the predefined workflow class with the input gotten from the submit button. The workflow generates a CP2K input file and submits the job using AiiDa. 4) AiiDa waits for the job to finish and collects the results. 5) The results can be queried and visualized using the visualization GUI.

3.3.1 GUI

The GUI allows the user to set the input for their IR calculation. We will first go over the version for a molecule on a slab since the GUI for a molecule in gas phase is very similar.

The information to be gathered from the GUI is organized in a dictionary. While the user enters parameters for the calculation the dictionary is continually updated until it is finally submitted with the submit button. There are different components of the GUI used to enter parameters. Each of these components is an ipywidget sometimes comprised of many ipywidgets. The first four components are the structure browser, the viewer, the computer and code dropdown, and the

DFT details. Each of these components have been adapted from previous work at nanotech@surfaces. The structure browser allows the user to browse uploaded, edited and computed .xyz files stored in the AiiDa database as shown in figure 3.3.

Submit Phonons Calculation (On Slab)

Figure 3.3: The time frame in which the structure was created and how the structure was created can be set in the structure browser. If the structure was computed the process that computed it can be selected as well.

Once a structure has been selected with the structure browser it will automatically be visualized in the viewer. Also the parameters for the DFT details ipywidget get set to a best guess based on the loaded structure. This is done by using the `observe` function of ipywidgets as shown in 3.1. The `observe` function then calls a custom function defined inside the Jupyter Notebook of the GUI.

Code Listing 3.1: Initialization of the structure browser. The function passed to `observe()` is called anytime a new structure is selected.

```
struct_browser = StructureBrowser()
struct_browser.results.observe(on_struct_change, names='value')
```

The computer and the code dropdown allow the user to select the code and select where to run it. For the time being this application only supports the CP2K code on Daint. In the DFT details section of the application the values for Dispersion Correction and "MGRID_CUTOFF" are set to standard values from the beginning on. The "Dispersion Correction" can be either activated or deactivated. When activated it adds an empirical pair potential to the standard PBE exchange functional that we use. "MGRID_CUTOFF" defines the cutoff for the plane waves used to represent the charge density [21]. The fields "# Nodes" and "Fixed Atoms" are set once a structure is loaded using an external function `suggested_parameters`. "# Nodes" is set to be as many nodes as are

necessary to do one energy calculation. This is not the total number of nodes requested when the job is submitted. This number can be seen in the next section of the application in the field "Total # Nodes" and it is computed from the "# Nodes" and the "# Replicas" fields. "Fixed Atoms" is either set to the slab without the top two layers for full DFT or the entire slab for Mixed DFT and DFTB by default. For the average calculation in this section it suffices for the user to choose a name for their calculation and the method with which they want to compute the energies. The rest of the parameters are either set to a default or suggested upon selection of a structure and the calculation type. All inputs in the application outside of the structure that can be set are shown in figure 3.4.

The screenshot displays the CP2K application setup interface. At the top, there is a "Select computer" dropdown menu with the text "Please select a computer" and a downward arrow. Below this is a "Select code:" dropdown menu, also with a downward arrow. The main section contains several input fields and controls:

- # Nodes:** A numeric input field set to "1".
- Calculation Name:** A text input field containing "A great name."
- Fixed Atoms:** A text input field set to "1..10", accompanied by a "show" button.
- Calculation Type:** Three radio buttons labeled "Mixed DFTB", "Mixed DFT", and "Full DFT". The "Full DFT" button is currently selected.
- Dispersion Corrections:** A greyed-out section with a label "Dispersion Corrections".
- MGRID_CUTOFF:** A slider control ranging from 0 to 600, with a circle marker at approximately 300.
- # Processors per ...:** A numeric input field set to "12".
- # Processors per r...:** A numeric input field set to "12".
- # Replicas:** A numeric input field set to "1".
- Total # Nodes:** A numeric input field set to "1".
- Submit:** A greyed-out button.
- walltime:** A numeric input field set to "86000".

Figure 3.4: Once the structure has been loaded all necessary parameters to finalize the input of a calculation are listed in this figure. Most of them can be estimated by the application from the selected structure. Some have to be set individually depending on how exact the calculation has to be (Calculation Type) and how many resources are available (Number of Replicas).

The next section of the application is solely for defining input relevant to the vibrational analysis performed by CP2K. Here the field "# Processors per node" is also set to a standard value for Daint. "Processors per replica" is updated automatically whenever "# Processors per node" or "# Nodes" is updated. This is again done using the observe function of ipywidgets as well as the custom

function passed to observe.

Code Listing 3.2: Initialization of the DFT and the vibrational analysis section. A lot of the widgets contained inside of these widgets will set new values for other parameters once their value has changed. This is always done by calling `observe()`.

```
##DFT
dft_details_widget = DFTDetails(job_details = job_details
                                ,
                                widgets_disabled = {
                                    'max_force' : True
                                }
                                )
dft_details_widget.btn_fixed_atoms.on_click(
    on_fixed_atoms_btn_click)
dft_details_widget.calc_type.observe(guess_calc_params,
                                     names='value',
                                     )
dft_details_widget.num_machines.observe(update_nproc_rep,
                                       names='value',
                                       )

##PHONON
phns_details_widget = PHNSDetails(job_details=job_details)
phns_details_widget.proc_node.observe(update_nproc_rep,
                                     names='value',
                                     )
phns_details_widget.num_rep.observe(update_nproc_rep,
                                   names='value',
                                   )
```

Code Listing 3.3: Function keeping track of the number of processors per replica. It is called by different widgets as can be seen in the previous code snippet. This is because it's suggested value depends on different parameters set by the user or the internal logic of the application.

```
def update_nproc_rep(c):
    num_nodes = dft_details_widget.num_machines.value
    proc_node = phns_details_widget.proc_node.value
    phns_details_widget.proc_rep.value=num_nodes*proc_node
    phns_details_widget.tot_num_nodes.value=str(num_nodes*
                                                job_details['nreplicas'])
```

In the code 3.3 we see that the value of "# Replicas" is simply kept at the number of processors per node times the number of nodes. This information will be given on to CP2K to inform it how many processors should be used per energy calculation. The `update_nproc_rep` function also takes care of keeping the field "Total # Nodes" current. The field "# Replicas" can be set to choose how many energy calculations can be done in parallel. Entering a number here will update the last field in 3.4 "Total # Nodes" through the same pathways as shown in 3.2 and 3.3. This is solely an output to inform the user how many nodes their job

submission will be requesting. The last section of the application lets the user choose a wall time for their job and submit the job via the submit button. The submit button will be looked at in more detail in the next subsection.

3.3.2 Submit Button

The submit button is an ipywidget that inherits from the class `ipywidgets.VBox`. On submission it collects the dictionary created through the input of the user and checks that there are no discrepancies in it. For our case specifically this means checking if the number of calculations that have to be done is divisible by the number of replicas chosen. If this is not the case the submission is canceled and the user is alerted to alter his input. This has to be done since otherwise CP2K does not know how to split up the calculations onto the replicas and it will abort. There are also other checks being done that have been taken over from previous work at nanotech@surfaces. These include checking that a structure has been selected or if a name has been set for the calculation. If the checks go through the input dictionary from the GUI is parsed into a dictionary with three entries: The structure, the code and a `ParameterData` file for handing over the parameters to AiiDa. `ParameterData` is an AiiDa data type. This dictionary together with the workflow, which is also retrieved from the input dictionary is handed over to the `aiida.work.run.submit()` function as shown in 3.4. Which commands are encoded in the workflow will be described in the following subsection.

Code Listing 3.4: This is a excerpt from the `on_btn_submit_press` function of the submit button class. It is defined when the submit button widget is initialized and called when the button is pressed. Here the three-part dictionary is created and passed along to AiiDa together with the workflow.

```
arg_dict = {'code': self.code, 'structure': self.structure, '
           parameters': self.submit_details}
if self.struc_folder is not None:
arg_dict['struc_folder'] = self.struc_folder
outputs = submit(self.the_workchain, **arg_dict)
print(outputs)
```

3.3.3 Workflow

The workflow class, which inherits from the AiiDa `WorkChain` class, used in this application was adapted from previous work at nanotech@surfaces. The important part for our application is defined in the `run_phonons` function that can be seen in 3.5.

Code Listing 3.5: This section takes care of finalizing the input with `build_calc_inputs`, starting the calculation with the submit function and in-

forming the AiiDa engine that it has to wait for the calculation to finish by returning an instance of the ToContext class.

```
def run_phonons(self):
    self.report("Running CP2K GW")

    parameters_dict = self.inputs.parameters.get_dict()
    inputs = self.build_calc_inputs(code = self.
                                    inputs.code,
                                    parent_folder = None,
                                    structure = self.
                                    inputs.structure,
                                    input_dict =
                                    parameters_dict )

    self.report("inputs: "+str(inputs))
    self.report("parameters: "+str(inputs['parameters'].
                                       get_dict()))
    self.report("settings: "+str(inputs['settings'].get_dict())
               )

    future = submit(Cp2kCalculation.process(), **inputs)
    return ToContext(phonons_opt=Calc(future))
```

Here the information that was before parsed into a `ParameterData` file by the submit button is retrieved through the AiiDa framework. This together with the code and the structure, also retrieved from AiiDa, is used to build the input for the calculation in the function `build_calc_inputs`. In `build_calc_inputs` there are important modifications made to the input before submitting it. For instance the `.xyz` structure file gets converted to an AiiDa data type `SingleFileData`. Also if the calculation is not run with full DFT then an additional file has to be created from the initial `.xyz` file. The new file will only contain the molecule without the slab. The actual CP2K input file gets written in the external `get_cp2k_input` function which will be looked at more closely in the next subsection.

3.3.4 Get CP2K input

In the `get_cp2k_input` function a dictionary is created that contains all the keywords and values that have to be set for the CP2K calculation. Many standards were already set here through previous work at nanotech@surface. These include the basis sets used for the wave function optimization in DFT and the potentials used for the empirical force fields used in Mixed DFT. For this application the `VIBRATIONAL_ANALYSIS` section for the CP2K input had to be defined.

Code Listing 3.6: The vibrational analysis parameters are mostly set to default values. However the `NPROC_REP` parameter is set to the user input from the GUI referred to earlier as the number of processors per replica.

```
if self.workchain == 'PhononsWorkchain':
```

```

self.inp['VIBRATIONAL_ANALYSIS'] = {
    'NPROC_REP': '%d' % self.inp_dict['nproc_rep'],
    'DX': '0.005',
    'INTENSITIES': 'T',
    'PRINT': {
        'PROGRAM_RUN_INFO': {'_': 'ON'}
    }
}

```

In 3.6 the value of `NPROC_REP` is being set to let CP2K know how many processors to use per energy calculation. The value of `DX` determines how much the atoms get perturbed in the finite difference approach. As mentioned in the Physics section of this report we need to know if the computed modes are IR active due to a change in polarization along that mode. For this we need the intensity of each vibration so `INTENSITIES` is set to `T` for true.

3.3.5 Gas phase and on slab version

As mentioned in the beginning of this section there is a version of the application that allows you to do IR calculations on a molecule in gas phase.

| | | |
|------------------------|---------------------------|--------|
| # Nodes | 1 | ^ v |
| Calculation Name: | A great name. | |
| Dispersion Corrections | | |
| MGRID_CUTOFF: | <input type="range"/> 600 | |
| # Processors per ... | 12 | ^ v |
| # Processors per r... | 12 | ^ v |
| # Replicas | 1 | ^ v |
| Total # Nodes | 1 | |

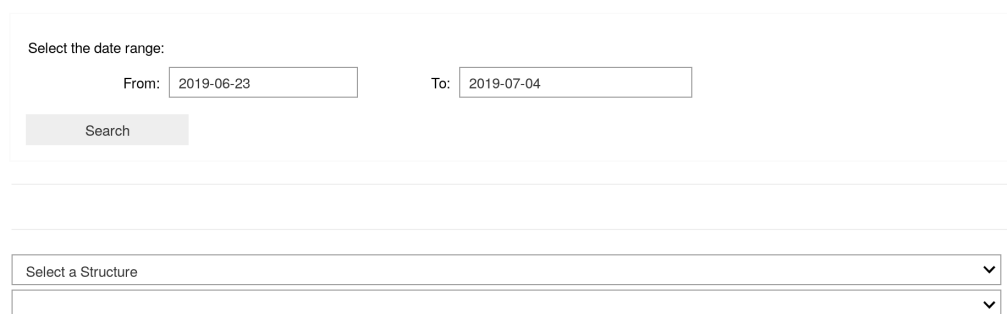
Figure 3.5: Next to the DFT type that does not have to be set for this case the field "Fixed Atoms" is left out as well since it does not make sense to fix any atoms of a molecule in these kinds of calculations. When simulating with a slab a few atoms can be fixed only because their interactions with the molecule are negligible.

The main difference in the GUI is the fact that here you can not choose between different versions of DFT since it would not make sense conceptually to split a molecule into parts that should be treated by DFT and other that shouldn't. To avoid confusion a check has been implemented to stop molecules from being submitted to the on slab version of the GUI and vice versa.

3.3.6 Visualization

The visualization is a standalone GUI. It employs an adapted version of the structure browser mentioned in the GUI used to submit calculations as seen in 3.6. This browser allows the user to browse the molden files generated by CP2K vibrational analysis computations.

Visualize Results



Select the date range:

From: 2019-06-23 To: 2019-07-04

Search

Select a Structure

Figure 3.6: The queries to the AiiDa database are hidden behind the interface allowing the user to find results simply by entering the time window in which they were created.

It queries the AiiDa database for nodes that have the **FolderData** AiiDa data type as an output and that were created by the previously mentioned workchain. This is done using an instance of the AiiDa **QueryBuilder** class as seen in 3.7. A time range in which to search for these nodes is set as well and can be altered by the user.

Code Listing 3.7: To find the results of a calculation an instance of the AiiDa class **QueryBuilder** is used. First nodes of the type `data.folder.FolderData` are identified. Then the nodes are selected that have the previously found nodes as an output.

```
qb = QueryBuilder()
qb.append(Node, filters={'type': 'data.folder.FolderData.', 'ctime':
                        :{'and': [{'<=': self.end_date}, {'
                        >': self.start_date}]}}}, tag='
                        output')
qb.append(Node, filters={'label': 'phonons_opt'}, input_of='output')
```

After these nodes have been found it is checked that they actually have the .mol file which we are interested in. The ones that do are added to the option that the user can choose from when selecting the calculation to visualize. Once a calculation has been chosen the GUI checks how many modes are available to be visualized and passes this information on to the structure browser which

then allows the user to choose one. Upon selection of the mode the vibration is visualized together with an interactive GUI provided by NGLview. Below this the actual infrared spectrum is plotted as well. All of this is done in an adapted version of the visualizer used for the input GUI.

Example

This chapter is a demonstration on how to submit and visualize an infrared computation using the application created in this project.

4.1 Submission

In the figure 4.1 the first step of submitting a calculation can be seen. Here the parameters to search for the structure we want to do a computation on can be set. In this case an uploaded structure was selected. With the "Process Label" field the type of calculation the structure should come from can be selected. That is not necessary for uploaded structures. The time frame in which to search for the structure also didn't have to be altered, since it was uploaded recently. Based on the set parameters a list structure to choose from is returned. In this case the structure is labeled by the line starting with "PK:516". This is the ID of the node in the AiiDa database representing the structure we are loading.

Submit Phonons Calculation

Select the date range:

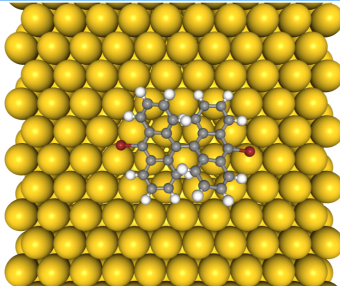
From:

To:

☒ all
☐ uploaded
☐ edited
☐ calculated

Process Label

PK: 516 | 2019-06-21 14:54 | C28H136Au480Br2 | upload(C28H136Au480Br2)_all_opt



Slab contains:
bottom H: 47..166
slab atoms: 167..646
slab layer 1: 167..286
slab layer 2: 287..406
slab layer 3: 407..526
slab layer 4: 527..646
#1 molecules: 1..46

Figure 4.1: Using few parameters the number of structure to choose from can be limited. A selected structure is directly visualized and can interactively be checked.

Once a structure is selected it is visualized and can be moved and inspected to check that the right structure is loaded. Below the visualization of the structure there is also a summary of the loaded structure printed. It shows in which logical fragments the structure is split up into internally by the application. The information can be used when fixing atoms later on in the application if the default settings are not sufficient.

Once the structure has been loaded and checked using the visualization the parameters for the calculation need to be set, as seen in figure 4.2. First the computer and the code to run on it have to be chosen. In any case the code to run will be CP2K. The computer would have to be setup via ssh in AiiDa and needs to be able to run CP2K. In this the case the computer to run on is Daint. After the computer and the code the parameters for the DFT calculations have to be set. For this structure the value of "# Nodes" and fixed atoms has been set to 14 and 47..646 respectively by the `suggested_parameters` function called

when a structure is loaded.

The calculation name can be chosen freely. For this calculation mixed DFT to compute the energies due to the size of the structure was chosen. Dispersion correction and the value for "MGRID_CUTOFF" are set to default values and left for this calculation.

Select computer

daint-s904

Select code:

cp2k_6.1_18464@daint-s904

Nodes 14

Calculation Name: IR example

Fixed Atoms 47..646 show

Calculation Type Mixed DFTB Mixed DFT Full DFT

Dispersion Corrections

MGRID_CUTOFF: 600

Processors per ... 12

Processors per r... 168

Replicas 4

Total # Nodes 56

Submit

walltime 86000

Figure 4.2: The options for the computer and the code are limited. The rest of the parameters in this section let the user set how to parallelize the phonon calculation and set the type of DFT calculation to be used.

The last parameters to be set relate to the vibrational analysis. "# Processors per node" is set to the default value of twelve for Daint. "# Processors per replica" is computed from the number of nodes and the number of processors per node. The main parameter that has to be chosen by the user and that is not set automatically is the value of "# Replicas". Depending on the size of the calculation this may vary strongly. One should keep in mind that for each free atom 6 energy calculations have to be computed. The more replicas that are available the more of these calculations can be done in parallel but each replica requires as many nodes as defined in "# Nodes". Here the calculation is run with 4 replicas. Still it will take approximately 2-3 hours to run this calculation. The "#Nodes" field helps keep track of how many nodes will be requested for the job once submitted. It is only an output and there is nothing to set here. The

wall time is set to a default value of 24 hours or 86000 seconds. If certain that the execution of the calculation will occur faster the wall time should be lowered as to get a higher priority in the queue of jobs submitted to the HPC cluster. Finally only the submit button is left to be pressed. When the submit button is pressed and all the parameters were set correctly the application will print out a summary of the job submission as seen in figure 4.3

```
SUBMITTING workchain <class 'apps.phonons.phonons_work.PhononsWorkchain'>

nreplicas 4
num_cores_per_mpiPROC 1
walltime 86000
nproc_rep 168
proc_node 12
num_machines 14
fixed_atoms 47..646
workchain PhononsWorkchain
tot_num_nodes 56
vdw_switch True
ncalcs 276
calc_type Mixed DFT
elements ['H', 'C', 'Au', 'Br']
calc_name IR example
mgrid_cutoff 600
RunningInfo(type=<RunningType.PROCESS: 0>, pid=657)

DONE
```

Figure 4.3: In this output mainly the pid is important for the average user. For developers it can be helpful for checking the state of the dictionary at the point of submission.

The summary consists of the work chain that this application uses to submit the job, the dictionary that has been created based on the input and some information on the process that has started to run. This last part is helpful to check the status of your calculation using the given pid. Once the job has finished the result can be visualized as described in the next section.

4.2 Visualizing

The part of the GUI for selecting the result of a phonon calculation in 4.4 is similar to structure browser in the GUI for submitting the calculations. A date range can be chosen in which the result was produced and a result within this date range can then be chosen through a dropdown menu. Here it is important to know the ID of the calculation of which the result is wanted. As mentioned in the last section this information is among the output given by the submission GUI when the submit button is pressed. Once a result has been chosen from

the dropdown menu a second dropdown menu allows you to choose a mode to visualize from this result. In this example mode number twelve was chosen to be visualized.

Visualize Results

Select the date range:

From:

To:

PK: 673 | 2019-07-01 10:00

Mode # 12

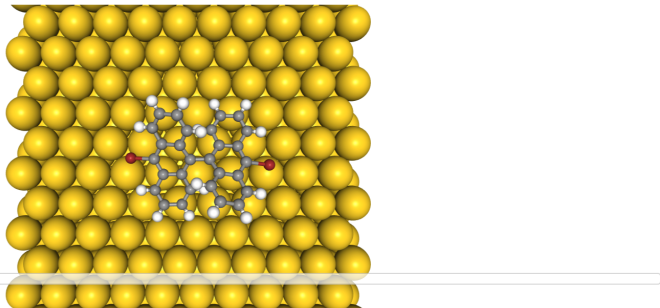


Figure 4.4: The component to select the wanted result is reduced compared to the structure browser that we saw in the submission GUI. This is because we already know that we are only looking for outputs from phonon calculations.

When the mode is selected as well it is visualized together with a pause/play button for controlling the animation. Below the visualization of the mode there is a menu with which one can edit the visualization on the go as seen in figure 4.5.



Figure 4.5: The menu of the visualization can help modify the representation of the mode in cases where the default representation is not clear enough. Another useful feature is the screenshot button. It lets you save a picture of the mode you are visualizing as a PNG.

Here for instance the step size of the animation or the representation of the atoms can be altered. Finally below the visualization of the mode there is also the IR spectra of the entire structure plotted as seen in 4.6. The x-axis is the energy of the impinging wave and the y axis is the absorption.

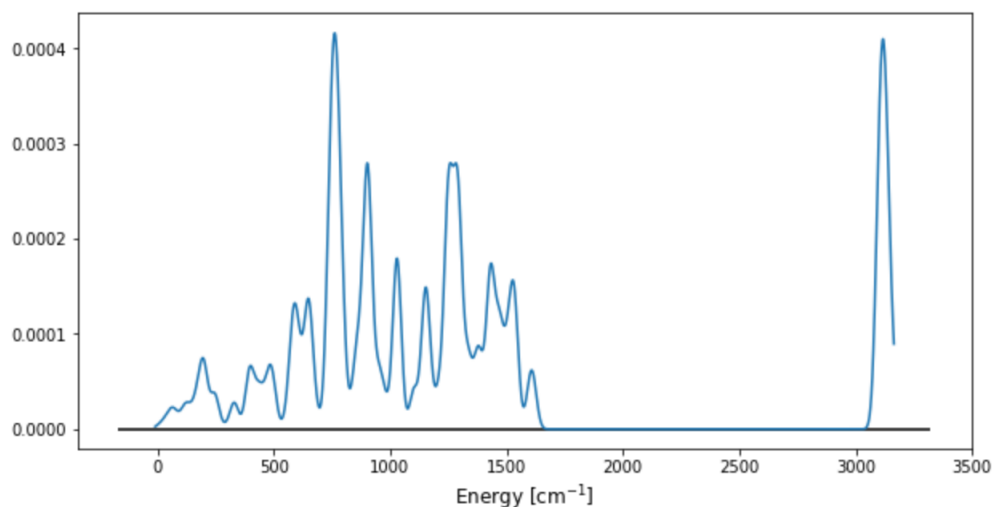


Figure 4.6: An experimental IR spectrum as the one plotted here can help identify structures and their states in experiments.

Future work

The application can successfully run IR calculations using DFT, mixed DFT and DFTB as well as visualizing the resulting modes and the spectra. The application could be expanded to do Raman calculations and some of the apps structure could be optimised by keeping the code for the GUI and processing of data more separated as well as avoiding code repetition.

5.1 Raman spectroscopy

Raman spectroscopy could be added as a feature to the application. Raman is another commonly used spectroscopy method. In our case it is very useful in helping to detect if a GNR remains intact. It is similar to IR spectroscopy but relies on a change in polarizability of a structure opposed to the change in dipole moment (polarization). In order to compute a Raman spectrum post processing of the vibrational modes would have to be done in order to detect the change in polarizability.

5.2 Application structure optimisation

Some optimisation to avoid code repetition could be done on the application. For instance the function `get_nfixed` is a slight alteration of a part of the function `analyze_slab`.

Code Listing 5.1: A function used for finding the number of fixed atoms. It is used to compute the number of calculations that have to be made.

```
def get_nfixed(slab_analyzed):
    method = dft_details_widget.calc_type.value
    valid_slab, msg = slab_is_valid(slab_analyzed, method)
    if method in ['Mixed DFTB', 'Mixed DFT']:
        full_slab = slab_analyzed['slabatoms'] + slab_analyzed['
                                bottom_H']
    full_slab = [i for i in full_slab]
```

```
    nfixed = len(full_slab)

    if method in ['Full DFT']:
        partial_slab = slab_analyzed['bottom_H'] + slab_analyzed['
                                slab_layers'][0] +
                                slab_analyzed['
                                slab_layers'][1]

        partial_slab = [i for i in partial_slab]
        nfixed = len(partial_slab)

    return nfixed
```

Integrating `get_nfixed` into `analyze_slab` and making it more general could help avoid code repetition.

The function `update_nproc_rep` depicted at 3.3 could probably be integrated into the ipywidgets that control the fields that it updates. When initializing these ipywidget one could add the observe functions for keeping track of and updating the necessary fields of the GUI directly inside the widgets instead of inside the code for the GUI. This would cause a cleaner separation between the logical setup of the GUI and the processing of the input in the background.

Bibliography

- [1] J. Cai *et al.*, “Atomically precise bottom-up fabrication of graphene nanoribbons.” *Nature*, 2010.
- [2] L. Talirz, P. Ruffieux, and R. Fasel, “On-surface synthesis of atomically precise graphene nanoribbons,” *Advanced Materials*, vol. 28, no. 29, pp. 6222–6231, 2016. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.201505738>
- [3] R. M. Martin, *Density functional theory: foundations*. Cambridge University Press, 2004, p. 119–134.
- [4] G. Pizzi, A. Cepellotti, R. Sabatini, N. Marzari, and B. Kozinsky, “Aiida: automated interactive infrastructure and database for computational science,” *Computational Materials Science*, vol. 111, pp. 218 – 230, 2016. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0927025615005820>
- [5] O. Schütt, E. Ditle, A. Yakutovich, D. Passerone, and C. A. Pignedoli, “Jupyter@materialscloud – an ecosystem to develop, execute and share scientific workflows.” *Comput. Mat. Sci. (in preparation)*, 2019.
- [6] O. Schütt *et al.*, “aiida-cp2k: Cp2k plugin for the aiida provenance and workflow engine.” [Online]. Available: <https://github.com/cp2k/aiida-cp2k>
- [7] “docker,” <https://www.docker.com/resources/what-container>, accessed: March 25th 2018.
- [8] “Project jupyter,” <http://www.jupyter.org>, accessed: March 25th 2018.
- [9] H. Jonsson, G. MILLS, and K. W. JACOBSEN, *Nudged elastic band method for finding minimum energy paths of transitions*, 06 1998, pp. 385–404.
- [10] S. Baroni, S. de Gironcoli, A. Dal Corso, and P. Giannozzi, “Phonons and related crystal properties from density-functional perturbation theory,” *Rev. Mod. Phys.*, vol. 73, pp. 515–562, Jul 2001. [Online]. Available: <https://link.aps.org/doi/10.1103/RevModPhys.73.515>
- [11] R. D. King-Smith and D. Vanderbilt, “Theory of polarization of crystalline solids,” *Phys. Rev. B*, vol. 47, pp. 1651–1654, Jan 1993. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevB.47.1651>

- [12] D. Xiao, M.-C. Chang, and Q. Niu, “Berry phase effects on electronic properties,” *Rev. Mod. Phys.*, vol. 82, pp. 1959–2007, Jul 2010. [Online]. Available: <https://link.aps.org/doi/10.1103/RevModPhys.82.1959>
- [13] X. Zhao and D. Vanderbilt, “Phonons and lattice dielectric properties of zirconia,” *Physical Review B*, vol. 65, 08 2001.
- [14] I. Souza, R. M. Martin, N. Marzari, X. Y. Zhao, and D. Vanderbilt, “Wannier-function description of the electronic polarization and infrared absorption of high-pressure hydrogen,” *Physical Review B*, vol. 62, no. 23, pp. 15 505–15 520, 2000. [Online]. Available: <http://infoscience.epfl.ch/record/179358>
- [15] M. S. Daw, S. M. Foiles, and M. I. Baskes, “The embedded-atom method: a review of theory and applications,” *Materials Science Reports*, vol. 9, no. 7, pp. 251 – 310, 1993. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/092023079390001U>
- [16] G. Seifert, “Tight-binding density functional theory: An approximate kohn-sham dft scheme,” *The journal of physical chemistry. A*, vol. 111, pp. 5609–13, 08 2007.
- [17] R. M. Martin, *The Kohn–Sham auxiliary system*. Cambridge University Press, 2004, p. 135–151.
- [18] S. Goedecker, M. Teter, and J. Hutter, “Separable dual-space gaussian pseudopotentials,” *Phys. Rev. B*, vol. 54, pp. 1703–1710, Jul 1996. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevB.54.1703>
- [19] J. VandeVondele and J. Hutter, “Gaussian basis sets for accurate calculations on molecular systems in gas and condensed phases,” *Journal of Chemical Physics*, vol. 127, no. 11, p. 114105, September 2007. [Online]. Available: <https://doi.org/10.5167/uzh-3160>
- [20] S. Grimme, “Density functional theory with london dispersion correction,” *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 1, pp. 211 – 228, 03 2011.
- [21] “Cp2k input manual,” <https://manual.cp2k.org/>, accessed: June 27th 2019.