MaX prize for flagship codes application. Report

Advancement in science, technology or in code developments

Motivation A typical workflow of a many-body calculation using the Yambo code (consider as an example a GW + BSE calculation) involves multiple steps. It requires the calculation of the self-consistent and non self-consistent ground state properties using a density functional theory (DFT) code like Quantum Espresso. Then, it requires us to translate the wave functions and pseudo-potentials to the Yambo internal netCDF format. Only then, we can run the many-body perturbation theory calculations as implemented in the Yambo code to calculate the GW quasi-particle energies and the excitonic states with the Bethe-Salpeter equation. The difficulties in handling the different steps are aggravated, as multiple convergence tests are required at the different steps to ensure accurate results. With the goal of simplifying these workflows and to run calculations more efficiently, we have created a new tool, yambopy. With this tool we intend to provide a framework to automatize Yambo calculations using python scripts. This concept has been implemented in other codes such as AbiPy for Abinit and AiiDA for Quantum Espresso and other *ab initio* codes. In the later, a plugin has been developed that also implements some features to run the yambo code from python scripts.

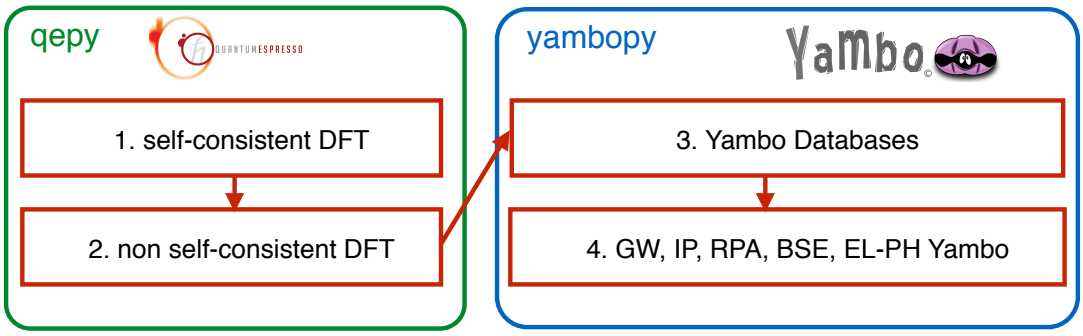

Figure 1: Schematic representation of a typical MBPT calculation workflow using Quantum Espresso and the Yambo code managed by the yambopy package.

Yambopy is a set of classes and scripts written in Python that provide a standard way to automatize and analyze many-body perturbation theory calculations made with the Yambo code. It allows reading, manipulating, and using the different basic quantities that enter or result from the calculations. It is composed of three basic packages that can be used independently: qepy to handly input and output files for Quantum Espresso calculations, yambopy for the yambo workflows and schedulerpy to submit and manage jobs on different cluster environments. Besides of the management of calculations, Yambopy includes the possibility of reading data. The user can easily prototype the implementation of new theoretical approaches using a convenient high-level language such as Python. For example, Yambopy can read the dipoles, the electron-phonon matrix elements, and the electronic states to calculate the Raman spectra in the framework of perturbation theory. Moreover, to ensure the quality of the code and its usability by the community we rely on three principles: open-source, documentation, and testing, We keep a public git repository hosted on Github where the users can get the latest version of the code as well as contribute with patches and new features. We also created a series of tests that are executed at each modification of the code in the Github repository using the Travis-CI platform. This ensures the reliability of the code despite its continuous development.

Visualization of simulations. Another tool intensively developed in Yambopy is the representation of the outcomes of simulations. Thus, the understanding of the results of excited state calculations is simplified if we have access to intermediate quantities obtained during the simulations. Therefore, Yambopy handles netCDF and output files of Yambo to visualize and analyze the data. For instance, in the case of the Bethe-Salpeter, we can analyze the independent-particle states that build the exciton state, gather the data of the exciton wave function represented at the reciprocal and real space. Thus, we can associate the peaks of the optical spectra to the excitonic wave functions. To better visualize the excitonic wave function, we created an interactive exciton website where the user can visualize the electronic density distribution with the hole fixed in a certain position. For GW calculations, Yambopy is able to read quasi-particle energies, represent the band structure, and plot the dielectric screening. Yambopy also contains a set of scripts to analyze carrier dynamics simulations, representing the occupation of the energy levels as a function of time and the occupations of selected bands in the Brillouin zone as a function of time.

Outlook. In the near future we plan to implement a series of additional features that will improve the usability of the code. One of these developments is to provide a framework to create "flows" for the different calculations like in the AbiPy or AiiDA codes. A "flow" consists of a series of interdependent tasks necessary to obtain the final result of a simulation. These tasks can be performed with the same or different codes. This allows the user to write a single Python script with all the steps of the calculation and the inter-dependencies of the multiple stages. Then it is possible to monitor the current status of the job and fix any problem that might occur at any stage of the calculation using a Python interface. Using this we can split the calculation into the maximum number of steps possible and optimize the parallelization for the different stages independently. Another aim is to automatize convergence tests.

Codes

Yambopy is a python code that provides a set of python classes to read and write Yambo input files and databasese. Included in the package is qepy wich handles the input and output files of Quantum Espresso.
Yambopy uses python 2.7 and some standard python libraries such as numpy, matplotlib, and netCDF4. We are currently adapting to code to be compatible with python 3.X.

References

Quantum interference effects in resonant Raman spectroscopy of MoTe2 from first principles, *Nano Letters* **17**, 2381 (2017).

Ab Initio Calculations of Ultrashort Carrier Dynamics in Two-Dimensional Materials: Valley Depolarization in Single-Layer WSe_2, *Nano Letters* **17**, 4549 (2017).

Documentation and tutorials of Yambopy: A detailed documentation of the classes, features, and a tutorial are available in: http://yambopy.readthedocs.io/en/latest/

GitHub repository of Yambopy: https://github.com/henriquemiranda/yambopy

Exciton website: http://henriquemiranda.github.io/excitonwebsite/

Links to images and videos

All movies can be found in this link:

https://github.com/alexmoratalla/MaxPrize2017/tree/master/movies

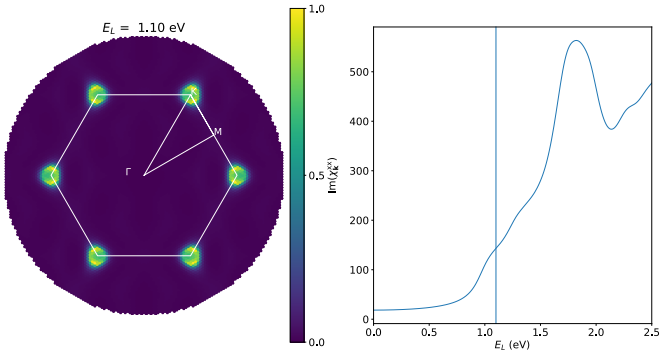

Figure 2: Visualization of the k-resolved contributions to the imaginary part of the dielectric susceptibility for single-layer $MoTe_2$ as a function of laser energy. Use the link above to watch the movie for different laser energies for single-layer $MoTe_2$ and for triple-layer $MoTe_2$.

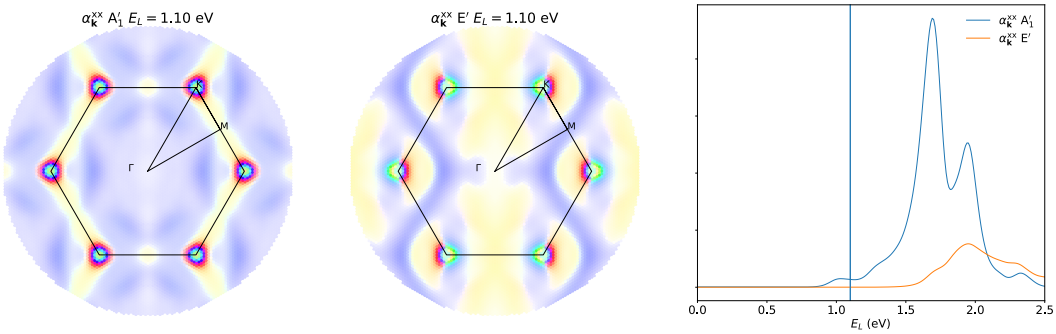

Figure 3: Visualization of the k-resolved contributions to the Raman susceptibility for single-layer $MoTe_2$ for the A'_1 and E' modes as a function of laser energy. Use the link above to watch the movie for different laser energies for single- layer $MoTe_2$ and to watch the movie for the two Raman active A'_1 modes in triple-layer $MoTe_2$.

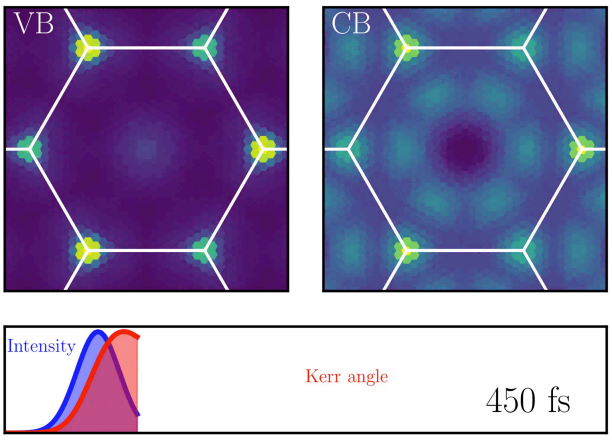

Figure 4: On top: occupations of the valence and conduction band (VB, CB) as a function of time. Lower panel: Intensity of the pump laser (blue) and Kerr angle (red).

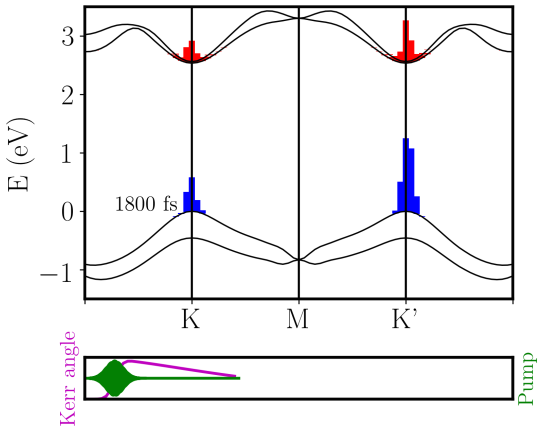

Figure 5: Animation of the occupations of the energy levels in the band structure of single-layer WS2.