**MaX prize for flagship codes application. Report**

* **Advancement in science, technology or in code developments**

**Why?** A typical workflow of a many-body calculation (consider as an example a GW + BSE calculation) involves multiple steps. It requires the calculation of the ground state properties using a DFT code. The next step is the translation of the wave functions and pseudo-potentials to the appropriate format. Finally, we can use a many-body perturbation theory as Yambo to calculate the GW quasi-particle energies and the exciton states with the Bethe-Salpeter equation. The difficulties in handling the many steps involved in the calculation are aggravated as convergence tests are unavoidable to ensure accurate results. In order to simplify the workflow of Yambo calculations and to run calculations more efficiently, we have created a new tool, Yambo\_Py, in the spirit of existing tools for ground state calculations as abipy (for abinit) and Aiida (for Quantum Espresso).

**What is?** Yambopy is a set of classes and scripts written in Python that aim to provide a standard way to automatize and analyze many-body perturbation theory calculations made with the Yambo code as well as to read, manipulate, and use the different basic quantities that enter the calculations. To ensure the quality of the code and its usability by the community we rely on three principles: open-source, documentation, and testing. A detailed documentation of the classes, features, and a tutorial are available in (http://yambopy.readthedocs.io/en/latest/). 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.

**What will be?** 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 allow the user to define and run “flows” like in the fireworks of abipy or aiida codes. A “flow” consists of all the tasks necessary to obtain a simulation result. These tasks can be performed with the same or different codes. This allows us to write one 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.

The understanding of the results of excited state calculations is simplified if we have access to intermediate quantities obtained during the simulations. In addition to the workflow management, Yambopy reads netCDF and output files of Yambo to visualize, analyze and handle the data. For instance, in the case of the Bethe-Salpeter, we can analyze all the partial information such as the independent-particle states that build the exciton state, the exciton wave function represented at the reciprocal and real space. We can associate the peaks of the optical spectra to the excitonic wave functions. Currently, for GW calculations, Yambopy is able to read quasi-particle energies, represent the band structure, plot the dynamical 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.

Another outcome of reading the data with Yambopy is the possibility for the user to easily prototype the implementation of new theoretical approaches using a convenient high-level language 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.

* **Codes**

Yambopy is a python code interfaced with Yambo (yambopy.py) and Quantum Espresso (qepy.py). Yambopy uses the version python2.7 and libraries such as matplotlib, netCDF4 (I don’t know if we need to say that). We are currently working on the code to adapt to python3.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 WSe2**, *Nano Letters* **17**, 4549 (2017).

**Documentation and tutorials of Yambopy**

(<http://yambopy.readthedocs.io/en/latest/>)

**GitHub repository of Yambopy**

[https://github.com/henriquemiranda/yambopy](https://github.com/henriquemiranda/yambopy/commit/f09cf8e2684777693d654a3f070b3ec437bc3922)

* **Links to images and videos**

- Exciton k-space mapping

- Raman spectra and visualization of the contributions on the Brillouin zone as a function of the laser energy

- Animation of the occupations on the BZ as a function of time.

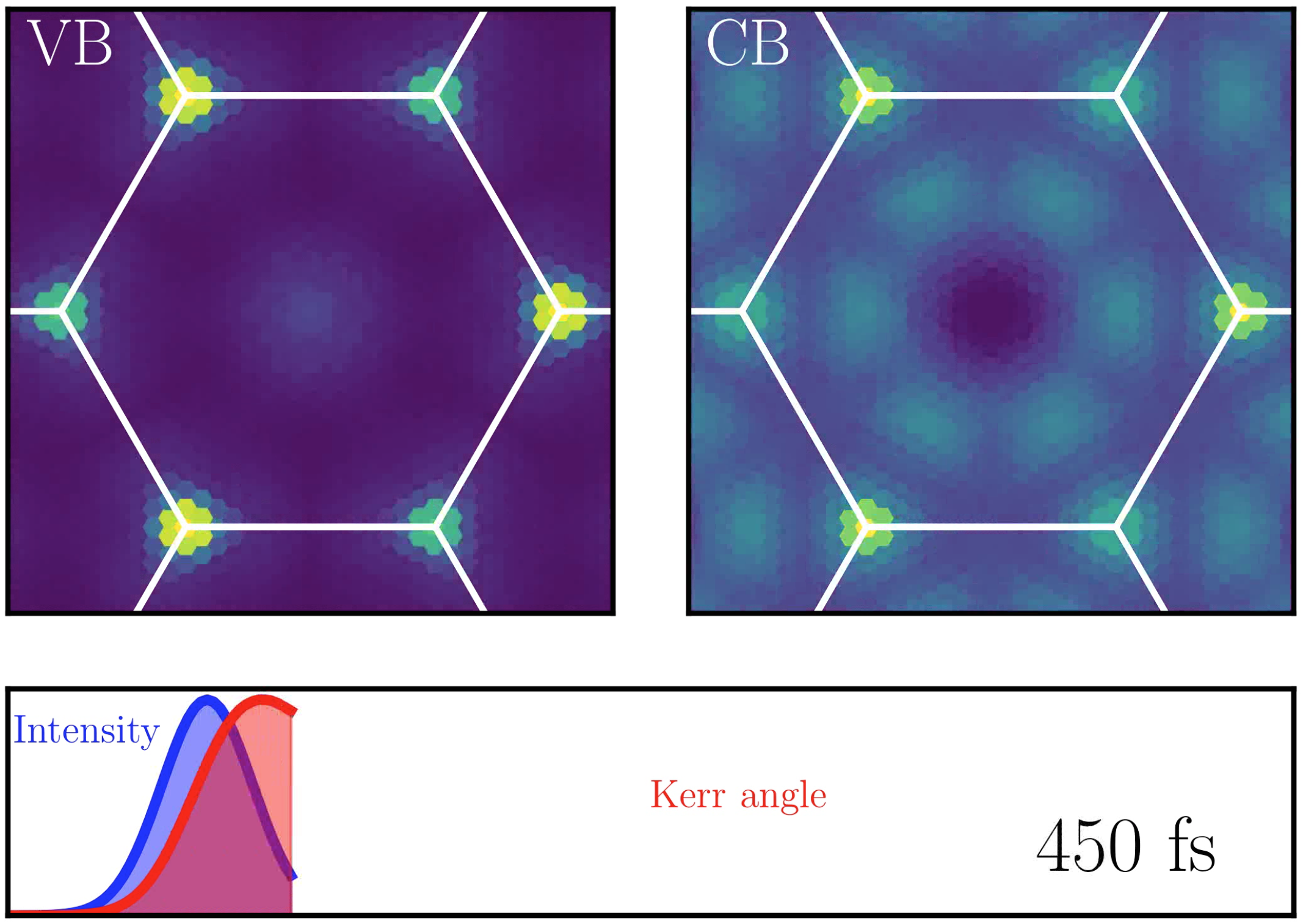


Figure 1: On top: occupations of the valence and conduction band (VB, CB) as a function of time. Lower panel: Intensity of the pump laser (blue) and Kerr angle (red). Link to video:  
<http://pubs.acs.org/doi/suppl/10.1021/acs.nanolett.7b00175>

- Animation of the occupations on the band structure as a function of time.