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

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

**Why?** 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. Then, it requires to translate the wave functions and pseudo-potentials to the Yambo internal format. Only then, we can perform 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 to ensure accurate results. To simplify these workflows and to run calculations more efficiently, we have created a new tool, yambopy, in the spirit of other existing tools for other codes such as AbiPy (for Abinit) and AiiDA (for Quantum Espresso).

Macintosh HD:Users:henrique.pereira:MaxPrize2017:yambopy.pdf

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

**What is?** 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.

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 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.

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 the independent-particle states that build the exciton state, gather de data of 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. 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.

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 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.

* **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 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.

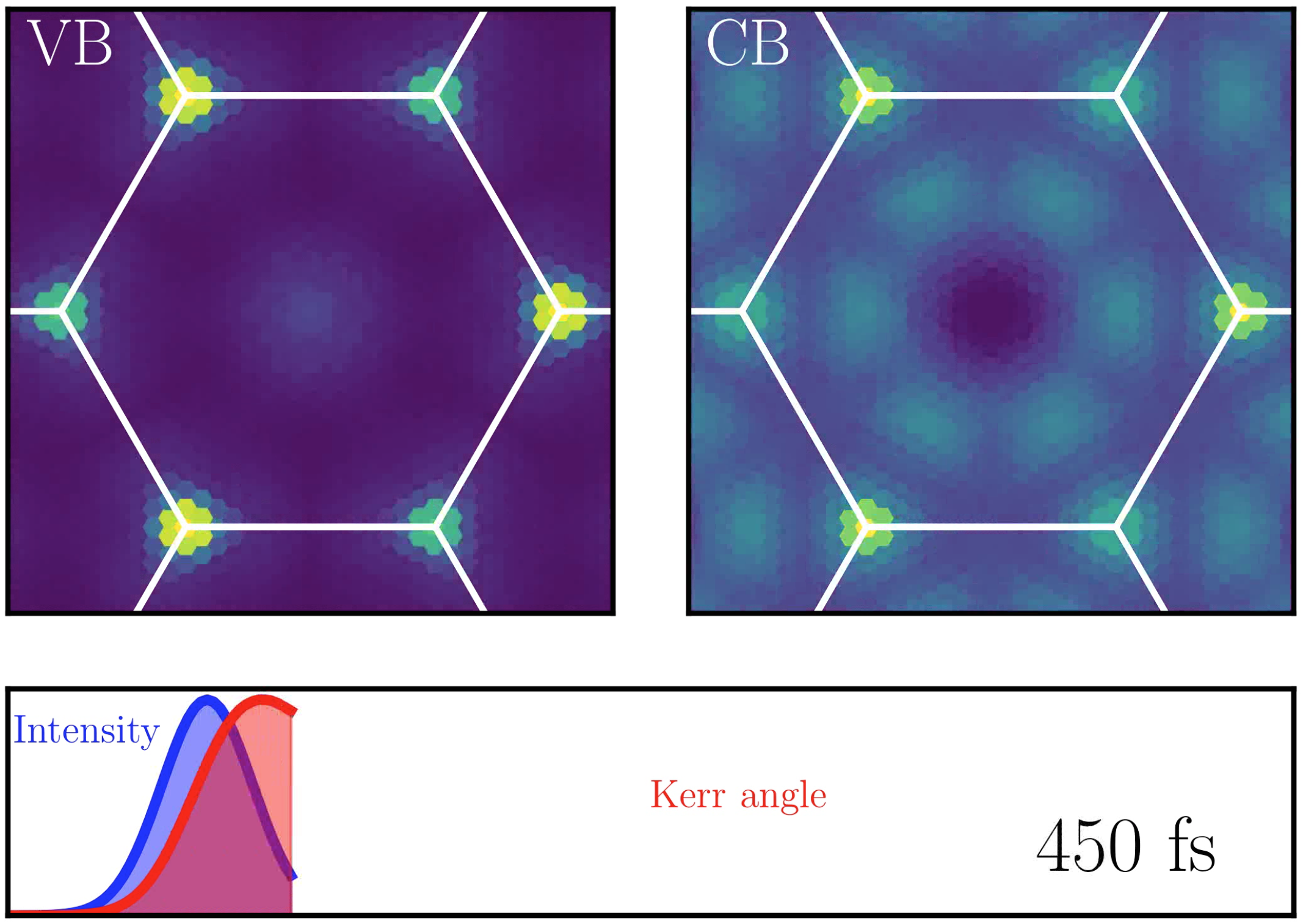


Figure 2: On top: occupations of the valence and conduction band (VB, CB) as a function of time. Lower panel: Intensity of the pump laser (blue) and Kerr angle (red). Link to video:  
<http://pubs.acs.org/doi/suppl/10.1021/acs.nanolett.7b00175>

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