MARVEL
NATIONAL CENTRE OF COMPETENCE IN RESEARCH

**Pillar 4**

EPFL

# koopmans

## *an open-source package for accurately and efficiently predicting spectral properties with Koopmans functionals*

**Edward Linscott,[1] Nicola Colonna,[2,3] Riccardo De Gennaro,[1] and Nicola Marzari[1,2,4]**

## Summary

`koopmans` contains everything you need to run a Koopmans functional calculation without expert knowledge, including implementations of Koopmans functionals in QUANTUM ESPRESSO as well as automated workflows.

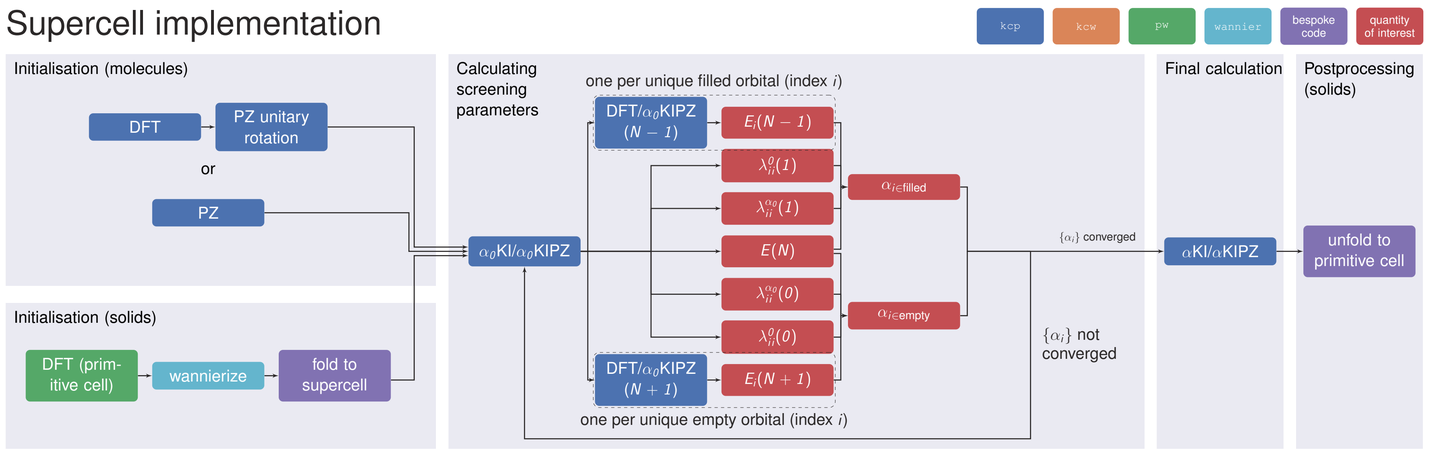
## 1. What are Koopmans functionals?

Koopmans functionals are a class of functionals that aim to reproduce spectral properties (charged excitations) and total energies on the same footing, enforcing a generalised piecewise linearity condition:

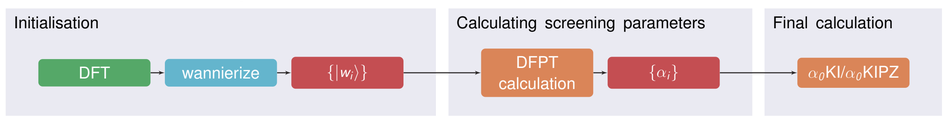
$$E_{\text{Koopmans}} = E_{\text{DFT}} + \sum_i \alpha_i \Big[ \underbrace{-\left(E_{\text{DFT}} - E_{\text{DFT}}|_{f_i=0}\right)}_{\text{removes the erroneous curvature}} \underbrace{+f_i\eta_i}_{\text{restores the correct linearity}} \Big]$$

## 2. Workflows

### Supercell implementation




### Primitive cell implementation

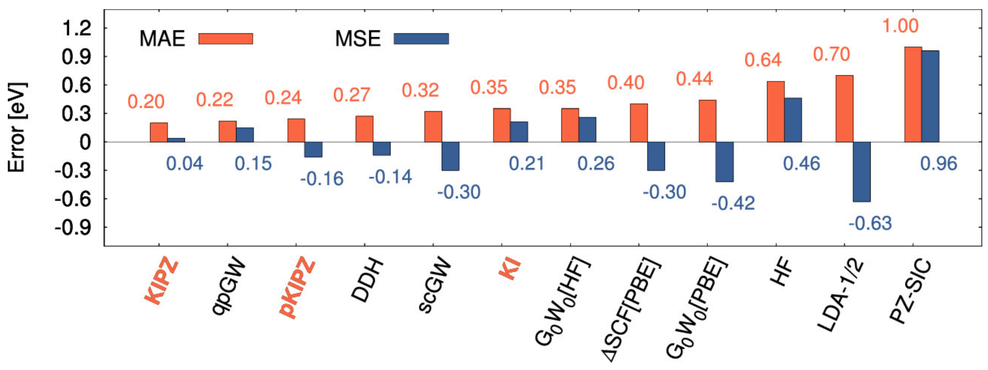


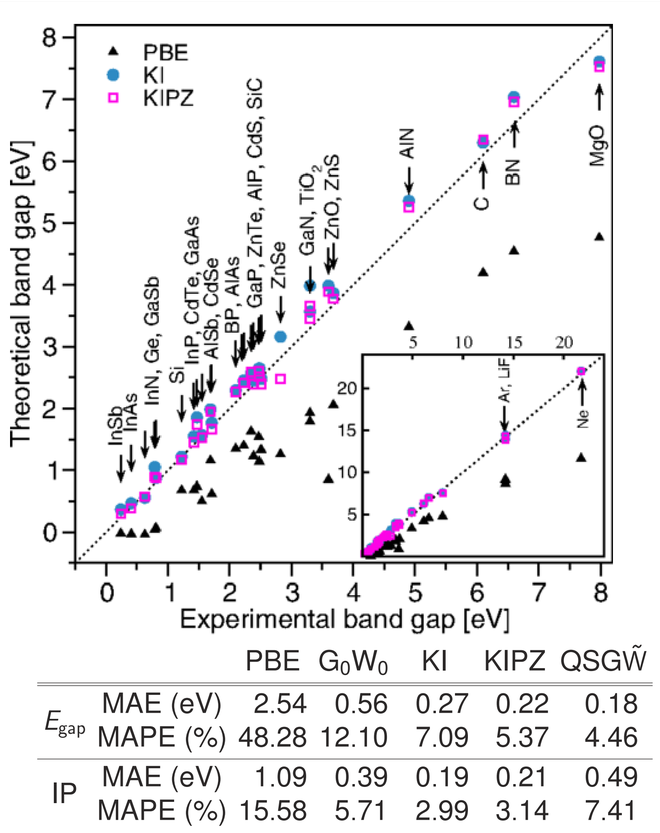

## 3. Example results

**Koopmans functionals give band structures and orbital energies as accurate as state-of-the-art GW, at a fraction of the computational cost.**

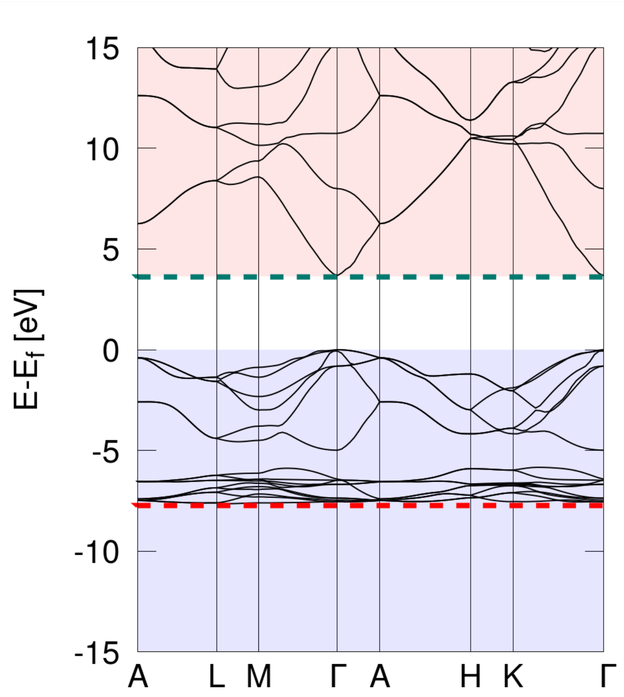
**Semiconductors and insulators** (right) Band gap and ionisation potential of semiconductors and insulators compared to experiment Nguyen *et al.* 2018

**Molecules** (below) The KI, pKIPZ, and KIPZ Koopmans functionals give ionization potentials comparable to qpGW across the GW100 dataset Colonna *et al.* 2019




| | | PBE | $G_0W_0$ | KI | KIPZ | QSGŴ |
|---|---|---|---|---|---|---|
| $E_{\text{gap}}$ | MAE (eV) | 2.54 | 0.56 | 0.27 | 0.22 | 0.18 |
| | MAPE (%) | 48.28 | 12.10 | 7.09 | 5.37 | 4.46 |
| IP | MAE (eV) | 1.09 | 0.39 | 0.19 | 0.21 | 0.49 |
| | MAPE (%) | 15.58 | 5.71 | 2.99 | 3.14 | 7.41 |




The KI band structure of ZnO compared to the experimental band gap and *d*-band position Colonna *et al.* 2022

## 4. What does running `koopmans` look like?

Koopmans takes a single `JSON` file as input e.g. for bulk silicon

```
{
  "workflow": {
    "functional": "ki",
    "base_functional": "lda",
    "method": "dscf",
    "init_orbitals": "mlwfs",
    "pseudo_library": "pseudo_dojo_standard"
  },
  "atoms": {
    "cell_parameters": {
      "periodic": true,
      "ibrav": 2,
      "celldms": {"1": 10.2622}
    },
    "atomic_positions": {
      "units": "crystal",
      "positions": [["Si", 0.00, 0.00, 0.00],
                    ["Si", 0.25, 0.25, 0.25]]
    }
  },
  "kpoints": {"grid": [4, 4, 4], "path": "LGXKG"},
  "calculator_parameters": {
    "pw": {"system": {"nbnd": 20}},
    "w90": {"auto_projections": true}
  }
}
```

Running from the command line looks like this:

```
$ koopmans si.json
  _
 | | _____   ___  _ __  _ __ ___   __ _ _ __  ___
 | |/ / _ \ / _ \| '_ \| '_ ` _ \ / _` | '_ \/ __|
 |   < (_) | (_) | |_) | | | | | | (_| | | | \__ \
 |_|\_\___/ \___/| .__/|_| |_| |_|\__,_|_| |_|___/
                 |_|

 Koopmans spectral functional calculations with Quantum
↪  ESPRESSO

 Written by Edward Linscott, Riccardo De Gennaro, and Nicola
↪  Colonna

  Initialisation of density and variational orbitals
  ==================================================

   Wannierisation
   ==============
    Running wannier/scf... done
    Running wannier/nscf... done
    Running wannier/occ_emp/wann_preproc... done

...

Workflow complete
```

or run with `python`:

```
from ase.build import bulk
from koopmans.kpoints import Kpoints
from koopmans.workflows import SinglepointWorkflow

# Use ASE to create bulk silicon
atoms = bulk('Si')

# Create the workflow
workflow = SinglepointWorkflow(atoms=atoms,
        parameters={'pseudo_library': 'pseudo_dojo_standard', 'base_functional': 'lda', 'init_orbitals': 'mlwfs'}
        kpoints = Kpoints(grid=[4, 4, 4], path='LGXKG', cell=atoms.cell),
        calculator_parameters = {'pw': {'nbnd': 20}, 'w90': {'auto_projections': True}}

# Run the workflow
workflow.run()
```

## References, affiliations, and acknowledgements

I. Dabo *et al.*, 2009, arXiv: 0901.2637.
I. Dabo *et al.*, Phys. Rev. B **82**, 115121 (2010).
G. Borghi *et al.*, Phys. Rev. B **90**, 075135 (2014).
N. Colonna *et al.*, J. Chem. Theory Comput. **14**, 2549–2557 (2018).
N. L. Nguyen *et al.*, Phys. Rev. X **8**, 021051 (2018).
N. Colonna *et al.*, J. Chem. Theory Comput. **15**, 1905–1914 (2019).
R. De Gennaro *et al.*, Phys. Rev. B **106**, 035106 (2022).
N. Colonna *et al.*, J. Chem. Theory Comput. (2022).
E. Linscott *et al.*, 2023, arXiv: 2302.07759.

1. Theory and Simulation of Materials (THEOS), École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland
2. National Centre for Computational Design and Discovery of Novel Materials (MARVEL), École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland
3. Laboratory for Neutron Scattering and Imaging, Paul Scherrer Institute, 5232 Villigen, Switzerland
4. Laboratory for Materials Simulations, Paul Scherrer Institut, 5232 Villigen, Switzerland

This work was supported by the Swiss National Science Foundation (SNSF) through its National Centre of Competence in Research (NCCR) MARVEL and Grants 179138 and 213082.

Swiss National Science Foundation

The National Centres of Competence in Research (NCCRs) are a funding scheme of the Swiss National Science Foundation

**... go to `koopmans-functionals.org` to find out more!**