

Koopmans functionals in practice

workflows, automation, and more...

- What does a Koopmans calculation actually involve?
- What tools are there for performing these calculations?
- Where can I find out more?

$$E_{\text{Koopmans}}[\rho, \{f_i\}, \{\alpha_i\}] = E_{DFT}[\rho] + \sum_i \alpha_i \left(- \underbrace{\int_0^{f_i} \varepsilon_i(f) df}_{\text{removes curvature}} + \underbrace{f_i \eta_i}_{\text{restores linearity}} \right)$$

$$E_{\text{Koopmans}}[\rho, \{f_i\}, \{\alpha_i\}] = E_{DFT}[\rho] + \sum_i \alpha_i \left(- \underbrace{\int_0^{f_i} \varepsilon_i(f) df}_{\text{removes curvature}} + \underbrace{f_i \eta_i}_{\text{restores linearity}} \right)$$

Differences to semi-local functionals:

$$E_{\text{Koopmans}}[\rho, \{f_i\}, \{\alpha_i\}] = E_{DFT}[\rho] + \sum_i \alpha_i \left(- \underbrace{\int_0^{f_i} \varepsilon_i(f) df}_{\text{removes curvature}} + \underbrace{f_i \eta_i}_{\text{restores linearity}} \right)$$

Differences to semi-local functionals:

- different flavours

$$E_{\text{Koopmans}}[\rho, \{\mathbf{f}_i\}, \{\alpha_i\}] = E_{DFT}[\rho] + \sum_i \alpha_i \left(- \underbrace{\int_0^{f_i} \varepsilon_i(f) df}_{\text{removes curvature}} + \underbrace{f_i \eta_i}_{\text{restores linearity}} \right)$$

Differences to semi-local functionals:

- different flavours
- orbital-density dependence

$$E_{\text{Koopmans}}[\rho, \{f_i\}, \{\alpha_i\}] = E_{DFT}[\rho] + \sum_i \alpha_i \left(- \underbrace{\int_0^{f_i} \varepsilon_i(f) df}_{\text{removes curvature}} + \underbrace{f_i \eta_i}_{\text{restores linearity}} \right)$$

Differences to semi-local functionals:

- different flavours
- orbital-density dependence
- screening

$$E_{\text{Koopmans}}[\rho, \{f_i\}, \{\alpha_i\}] = E_{DFT}[\rho] + \sum_i \alpha_i \left(- \int_0^{f_i} \varepsilon_i(f) df + f_i \eta_i \right)$$

One degree of freedom: what should be the gradient of this linear term?

$$E_{\text{KI}}[\rho, \{f_i\}, \{\alpha_i\}] = E_{DFT}[\rho] + \sum_i \alpha_i \left(- \int_0^{f_i} \varepsilon_i(f) df + f_i \int_0^1 \varepsilon_i(f) df \right)$$

One degree of freedom: what should be the gradient of this linear term?

- the base functional → “KI” (Koopmans integral). Enforces IP theorem. Does not affect energy/density!

$$E_{\text{KIPZ}}[\rho, \{f_i\}, \{\alpha_i\}] = E_{DFT}[\rho] + \sum_i \alpha_i \left(- \int_0^{f_i} \varepsilon_i(f) df + f_i \left\{ \int_0^1 \varepsilon_i(f) df - E_{Hxc}[n_i] \right\} \right)$$

One degree of freedom: what should be the gradient of this linear term?

- the base functional → “KI” (Koopmans integral). Enforces IP theorem. Does not affect energy/density!
- with a PZ correction → “KIPZ”

$$E_{\text{KIPZ}}[\rho, \{f_i\}, \{\alpha_i\}] = E_{DFT}[\rho] + \sum_i \alpha_i \left(- \int_0^{f_i} \varepsilon_i(f) df + f_i \left\{ \int_0^1 \varepsilon_i(f) df - E_{Hxc}[n_i] \right\} \right)$$

One degree of freedom: what should be the gradient of this linear term?

- the base functional → “KI” (Koopmans integral). Enforces IP theorem. Does not affect energy/density!
- with a PZ correction → “KIPZ”

You might also see...

$$E_{\text{KIPZ}}[\rho, \{f_i\}, \{\alpha_i\}] = E_{DFT}[\rho] + \sum_i \alpha_i \left(- \int_0^{f_i} \varepsilon_i(f) df + f_i \left\{ \int_0^1 \varepsilon_i(f) df - E_{Hxc}[n_i] \right\} \right)$$

One degree of freedom: what should be the gradient of this linear term?

- the base functional → “KI” (Koopmans integral). Enforces IP theorem. Does not affect energy/density!
- with a PZ correction → “KIPZ”

You might also see...

- “pKIPZ” = KIPZ Hamiltonian evaluated on the KI solution

$$E_{\text{KIPZ}}[\rho, \{f_i\}, \{\alpha_i\}] = E_{DFT}[\rho] + \sum_i \alpha_i \left(- \int_0^{f_i} \varepsilon_i(f) df + f_i \left\{ \int_0^1 \varepsilon_i(f) df - E_{Hxc}[n_i] \right\} \right)$$

One degree of freedom: what should be the gradient of this linear term?

- the base functional → “KI” (Koopmans integral). Enforces IP theorem. Does not affect energy/density!
- with a PZ correction → “KIPZ”

You might also see...

- “pKIPZ” = KIPZ Hamiltonian evaluated on the KI solution
- “K” = an earlier iteration based off half-filling rather than integer endpoints (no longer used)

$$-\int_0^{f_i} \varepsilon_i(f) df + f_i \int_0^1 \varepsilon_i(f) df = E_{\text{Hxc}}[\rho] + E_{\text{Hxc}}[\rho - \rho_i] + f_i (-E_{\text{Hxc}}[\rho - \rho_i] + E_{\text{Hxc}}[\rho - \rho_i + n_i])$$

$$-\int_0^{f_i} \varepsilon_i(f) df + f_i \int_0^1 \varepsilon_i(f) df = E_{\text{Hxc}}[\rho] + E_{\text{Hxc}}[\rho - \rho_i] + f_i (-E_{\text{Hxc}}[\rho - \rho_i] + E_{\text{Hxc}}[\rho - \rho_i + n_i])$$

Potential is given by $v_i(\mathbf{r}) = \frac{\delta E}{\delta \rho_i(\mathbf{r})}$. After some derivation...

$$v_i/\alpha_i = v_{\text{scalar}} + \delta_{ij} v_{\text{diag}}(\mathbf{r}) + (1 - \delta_{ij}) v_{\text{nondiag}}(\mathbf{r})$$

$$-\int_0^{f_i} \varepsilon_i(f) df + f_i \int_0^1 \varepsilon_i(f) df = E_{\text{Hxc}}[\rho] + E_{\text{Hxc}}[\rho - \rho_i] + f_i (-E_{\text{Hxc}}[\rho - \rho_i] + E_{\text{Hxc}}[\rho - \rho_i + n_i])$$

Potential is given by $v_i(\mathbf{r}) = \frac{\delta E}{\delta \rho_i(\mathbf{r})}$. After some derivation...

$$v_i/\alpha_i = v_{\text{scalar}} + \delta_{ij} v_{\text{diag}}(\mathbf{r}) + (1 - \delta_{ij}) v_{\text{nondiag}}(\mathbf{r})$$

For filled orbitals with KI:

$$v_i^{\text{KI}}/\alpha_i = -E_{\text{H}}[n_i] + E_{\text{xc}}[\rho] - E_{\text{xc}}[\rho - n_i] - \int d\mathbf{r}' v_{\text{xc}}(\mathbf{r}', [\rho]) n_i(\mathbf{r}')$$

Initialise with MLWFs, then (optionally) solve with CG minimisation:

Initialise with MLWFs, then (optionally) solve with CG minimisation:

outer loop: $\varphi_i^{(n+1)} = \varphi_i^{(n)} + \Delta_i$

Initialise with MLWFs, then (optionally) solve with CG minimisation:

outer loop: $\varphi_i^{(n+1)} = \varphi_i^{(n)} + \Delta_i$

inner loop: $\varphi_i^{(n+1)} = \sum_j U_{ij} \varphi_j^{(n)}$

Initialise with MLWFs, then (optionally) solve with CG minimisation:

outer loop: $\varphi_i^{(n+1)} = \varphi_i^{(n)} + \Delta_i$

inner loop: $\varphi_i^{(n+1)} = \sum_j U_{ij} \varphi_j^{(n)}$

For more details see G. Borghi et al. *Phys. Rev. B* 91.15 (2015), 155112

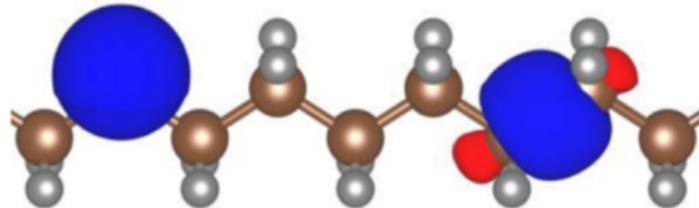
Initialise with MLWFs, then (optionally) solve with CG minimisation:

outer loop: $\varphi_i^{(n+1)} = \varphi_i^{(n)} + \Delta_i$

inner loop: $\varphi_i^{(n+1)} = \sum_j U_{ij} \varphi_j^{(n)}$

For more details see G. Borghi et al. *Phys. Rev. B* 91.15 (2015), 155112

Gives rise to a set of minimising orbitals (localised/variational)



(a) variational

Orbital-density-dependence

Initialise with MLWFs, then (optionally) solve with CG minimisation:

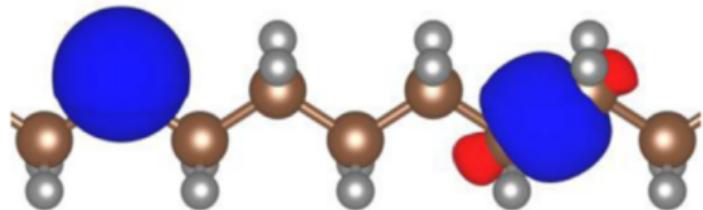
outer loop: $\varphi_i^{(n+1)} = \varphi_i^{(n)} + \Delta_i$

inner loop: $\varphi_i^{(n+1)} = \sum_j U_{ij} \varphi_j^{(n)}$

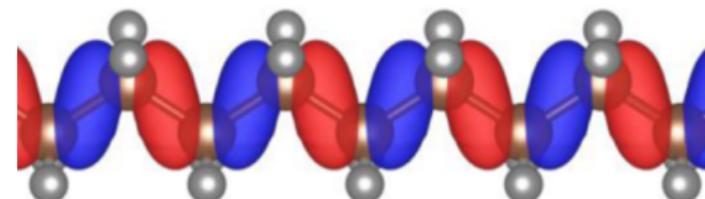
For more details see G. Borghi et al. *Phys. Rev. B* 91.15 (2015), 155112

Gives rise to a set of minimising orbitals (localised/variational)

Diagonalising at the minimum gives rise to diagonalising orbitals (delocalised/canonical)



(a) variational



(b) canonical

- What is screening (in the context of Koopmans functionals)?

$$\frac{dE}{df_i} \xleftarrow{?} \frac{\partial E}{\partial f_i}$$

including relaxation excluding relaxation

- What is screening (in the context of Koopmans functionals)?

$$\frac{dE}{df_i} \xleftarrow{?} \frac{\partial E}{\partial f_i}$$

including relaxation excluding relaxation

- In Hartree-Fock (the original “Koopmans’ theorem”):

$$E_{ee}^{HF} = \frac{1}{2} \sum_{ij} f_i f_j \int d\mathbf{r} d\mathbf{r}' \frac{|\psi_i(\mathbf{r})|^2 |\psi_j(\mathbf{r}')|^2}{\mathbf{r} - \mathbf{r}'} - \frac{\psi_i^*(\mathbf{r}) \psi_j^*(\mathbf{r}') \psi_i(\mathbf{r}') \psi_j(\mathbf{r})}{\mathbf{r} - \mathbf{r}'}$$

- What is screening (in the context of Koopmans functionals)?

$$\frac{dE}{df_i} \xleftarrow{?} \frac{\partial E}{\partial f_i}$$

including relaxation excluding relaxation

- In Hartree-Fock (the original “Koopmans’ theorem”):

$$E_{ee}^{HF} = \frac{1}{2} \sum_{ij} f_i f_j \int d\mathbf{r} d\mathbf{r}' \frac{|\psi_i(\mathbf{r})|^2 |\psi_j(\mathbf{r}')|^2}{\mathbf{r} - \mathbf{r}'} - \frac{\psi_i^*(\mathbf{r}) \psi_j^*(\mathbf{r}') \psi_i(\mathbf{r}') \psi_j(\mathbf{r})}{\mathbf{r} - \mathbf{r}'}$$

- Account for screening post-hoc:

$$\frac{dE}{df_i} \approx \alpha_i \frac{\partial E}{\partial f_i}$$

- What is screening (in the context of Koopmans functionals)?

$$\frac{dE}{df_i} \xleftarrow{?} \frac{\partial E}{\partial f_i}$$

including relaxation excluding relaxation

- In Hartree-Fock (the original “Koopmans’ theorem”):

$$E_{ee}^{HF} = \frac{1}{2} \sum_{ij} f_i f_j \int d\mathbf{r} d\mathbf{r}' \frac{|\psi_i(\mathbf{r})|^2 |\psi_j(\mathbf{r}')|^2}{\mathbf{r} - \mathbf{r}'} - \frac{\psi_i^*(\mathbf{r}) \psi_j^*(\mathbf{r}') \psi_i(\mathbf{r}') \psi_j(\mathbf{r})}{\mathbf{r} - \mathbf{r}'}$$

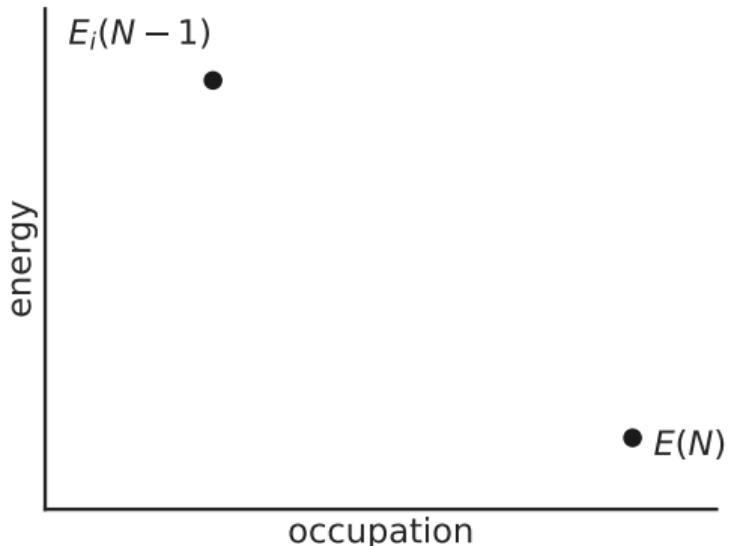
- Account for screening post-hoc:

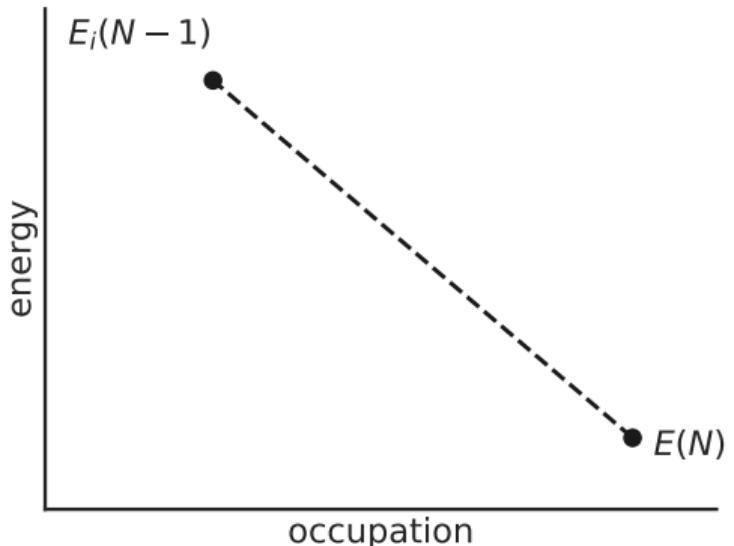
$$\frac{dE}{df_i} \approx \alpha_i \frac{\partial E}{\partial f_i}$$

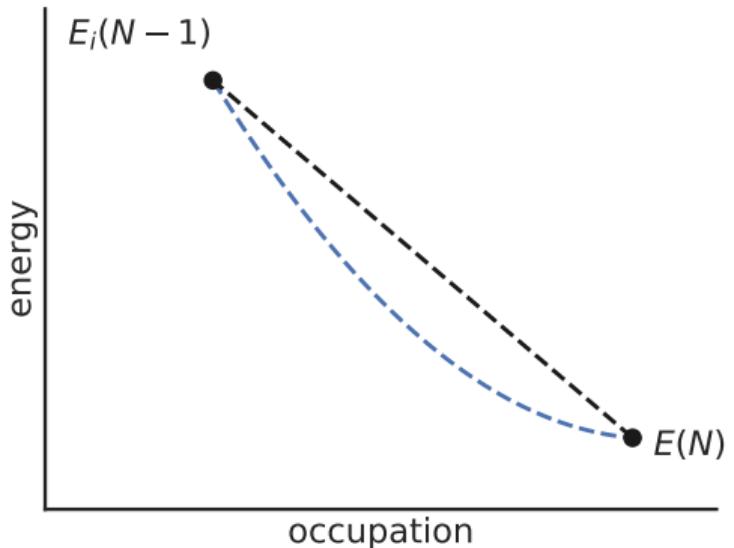
- How to choose an appropriate value for α_i ? Return to the original idea of Koopmans functionals:

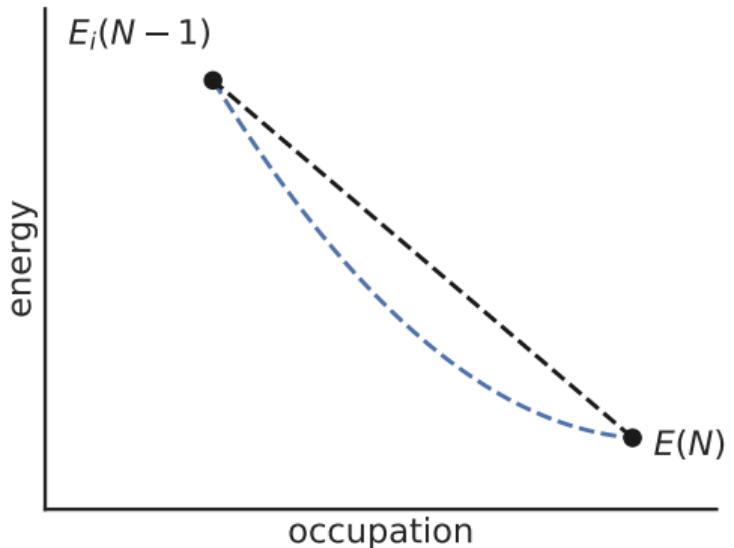
$$\varepsilon_i^{\text{Koopmans}} = E_i(N-1) - E(N)$$

Screening

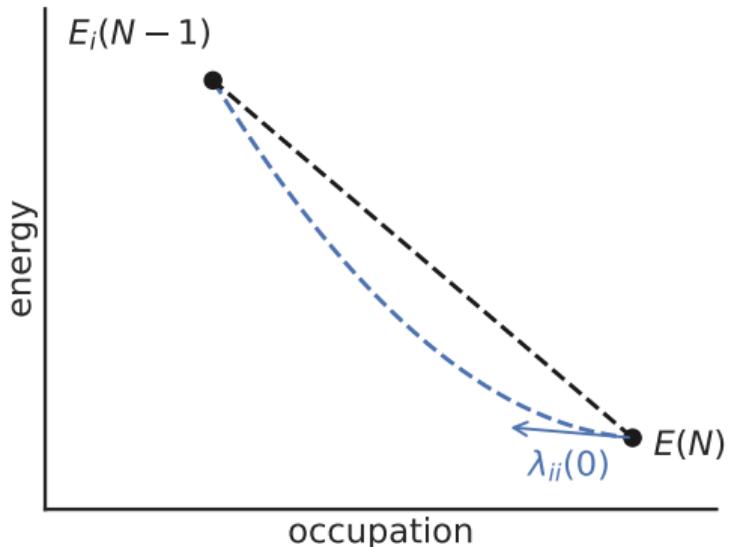




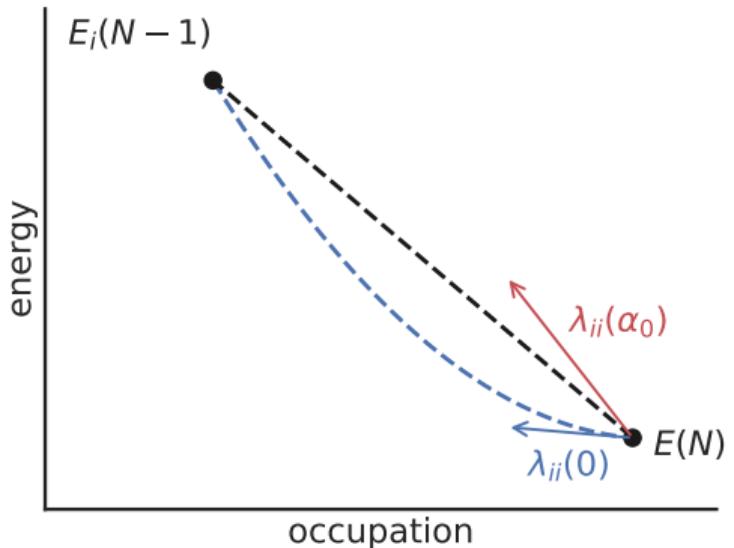




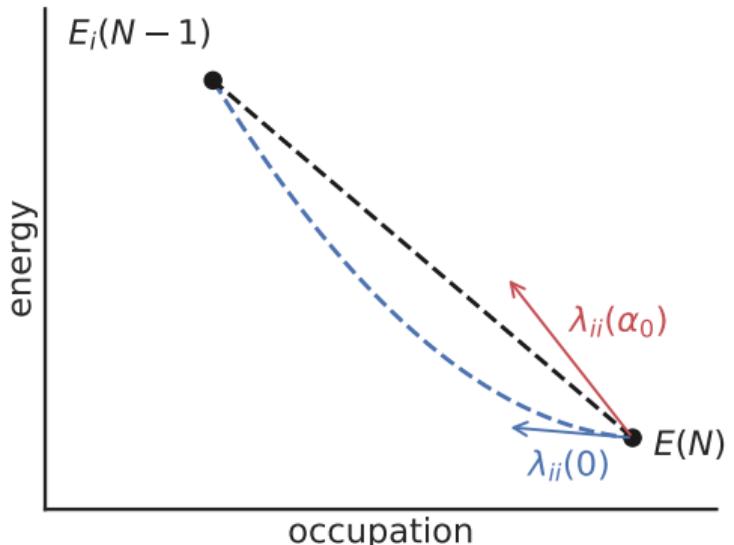
$$\lambda_{ii}(\alpha) \equiv \langle \varphi_i | \hat{h}^{\text{DFT}} + \alpha \hat{v}^{\text{Koopmans}} | \varphi_i \rangle = \frac{dE^{\text{Koopmans}}}{df_i} \Big|_{f_i=s}$$



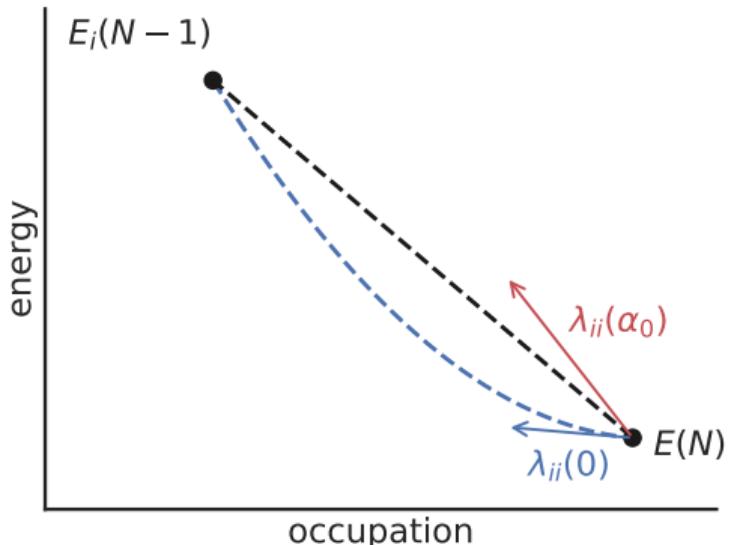
$$\lambda_{ii}(\alpha) \equiv \langle \varphi_i | \hat{h}^{\text{DFT}} + \alpha \hat{v}^{\text{Koopmans}} | \varphi_i \rangle = \frac{dE^{\text{Koopmans}}}{df_i} \Big|_{f_i=s}$$



$$\lambda_{ii}(\alpha) \equiv \langle \varphi_i | \hat{h}^{\text{DFT}} + \alpha \hat{v}^{\text{Koopmans}} | \varphi_i \rangle = \frac{dE^{\text{Koopmans}}}{df_i} \Big|_{f_i=s}$$



Given this, how to work out the ideal α ?



Given this, how to work out the ideal α ?

$$\alpha^{n+1} = \alpha^n \frac{E_i(N-1) - E(N) - \lambda_{ii}(0)}{\lambda_{ii}(\alpha^n) - \lambda_{ii}(0)}$$

$$\alpha^{n+1} = \alpha^n \frac{E_i(N-1) - E(N) - \lambda_{ii}(0)}{\lambda_{ii}(\alpha^n) - \lambda_{ii}(0)}$$

$$\alpha^{n+1} = \alpha^n \frac{E_i(N-1) - E(N) - \lambda_{ii}(0)}{\lambda_{ii}(\alpha^n) - \lambda_{ii}(0)}$$

↑
expectation value
of $\hat{H}^{\text{Koopmans}}$

$$\alpha^{n+1} = \alpha^n \frac{E_i(N-1) - E(N) - \lambda_{ii}(0)}{\lambda_{ii}(\alpha^n) - \lambda_{ii}(0)}$$

↑
expectation value
of \hat{H}^{DFT}
↑
expectation value
of $\hat{H}^{\text{Koopmans}}$

$$\alpha^{n+1} = \alpha^n \frac{E_i(N-1) - E(N) - \lambda_{ii}(0)}{\lambda_{ii}(\alpha^n) - \lambda_{ii}(0)}$$

total energy
of neutral system

↑
↑
↑

expectation value
of \hat{H}^{DFT}

expectation value
of $\hat{H}^{\text{Koopmans}}$

$$\alpha^{n+1} = \alpha^n \frac{E_i(N-1) - E(N) - \lambda_{ii}(0)}{\lambda_{ii}(\alpha^n) - \lambda_{ii}(0)}$$

total energy with electron removed from orbital i

total energy of neutral system

expectation value of \hat{H}^{DFT}

expectation value of $\hat{H}^{\text{Koopmans}}$

Original formulation requires explicit charged defect calculations in a supercell

¹ N. Colonna et al. *J. Chem. Theory Comput.* 15.3 (2019), 1905.

² N. Colonna et al. *J. Chem. Theory Comput.* (2022).

Original formulation requires explicit charged defect calculations in a supercell
Now reformulated in terms of DFPT¹...

$$\alpha_i = 1 + \frac{\langle v_{\text{pert}}^i | \Delta^i n \rangle}{\langle n_i | v_{\text{pert}}^i \rangle}.$$

¹ N. Colonna et al. *J. Chem. Theory Comput.* 15.3 (2019), 1905.

² N. Colonna et al. *J. Chem. Theory Comput.* (2022).

Original formulation requires explicit charged defect calculations in a supercell
Now reformulated in terms of DFPT¹...

$$\alpha_i = 1 + \frac{\langle v_{\text{pert}}^i | \Delta^i n \rangle}{\langle n_i | v_{\text{pert}}^i \rangle}.$$

... in reciprocal space²

$$\alpha_{0i} = 1 + \frac{\sum_{\mathbf{q}} \langle v_{\text{pert},\mathbf{q}}^{0i} | \Delta_{\mathbf{q}}^{0i} n \rangle}{\sum_{\mathbf{q}} \langle n_{\mathbf{q}}^{0i} | v_{\text{pert},\mathbf{q}}^{0i} \rangle}.$$

(See Nicola Colonna's talk)

¹ N. Colonna et al. *J. Chem. Theory Comput.* 15.3 (2019), 1905.

² N. Colonna et al. *J. Chem. Theory Comput.* (2022).

When performing a Koopmans calculation, you must decide...

When performing a Koopmans calculation, you must decide...

- which flavour (KI, KIPZ)?

When performing a Koopmans calculation, you must decide...

- which flavour (KI, KIPZ)?
- how are we treating the orbital-density-dependence?

When performing a Koopmans calculation, you must decide...

- which flavour (KI, KIPZ)?
- how are we treating the orbital-density-dependence?
 - how are we initialising our variational orbitals? (N.B. depends on the flavour!)

When performing a Koopmans calculation, you must decide...

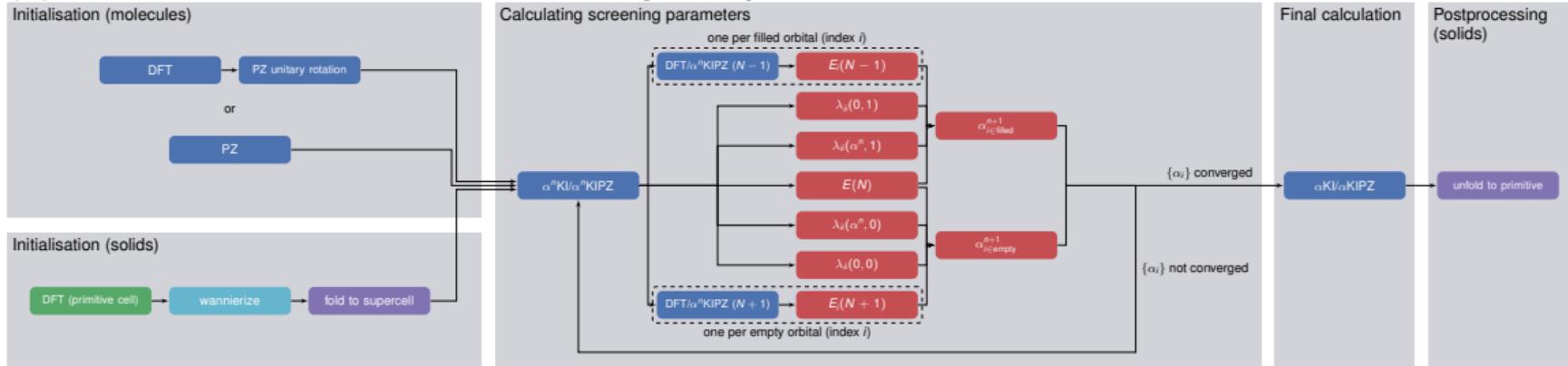
- which flavour (KI, KIPZ)?
- how are we treating the orbital-density-dependence?
 - how are we initialising our variational orbitals? (N.B. depends on the flavour!)
 - are we going to explicitly minimise the ODD?

When performing a Koopmans calculation, you must decide...

- which flavour (KI, KIPZ)?
- how are we treating the orbital-density-dependence?
 - how are we initialising our variational orbitals? (N.B. depends on the flavour!)
 - are we going to explicitly minimise the ODD?
- how are we calculating the screening parameters? (finite differences, DFPT, ML...)

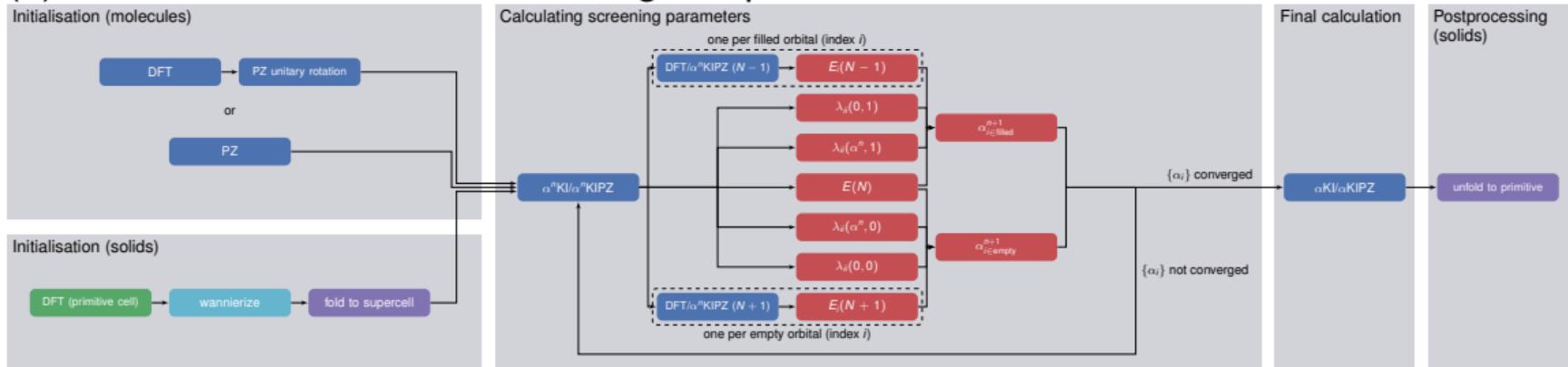
The workflows

(a) finite difference calculations using a supercell

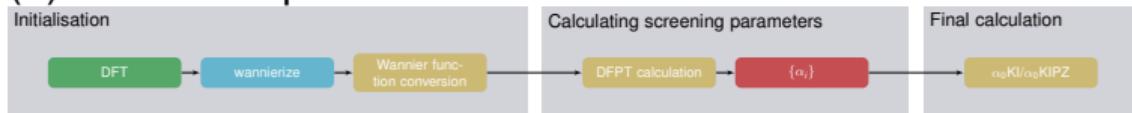


The workflows

(a) finite difference calculations using a supercell

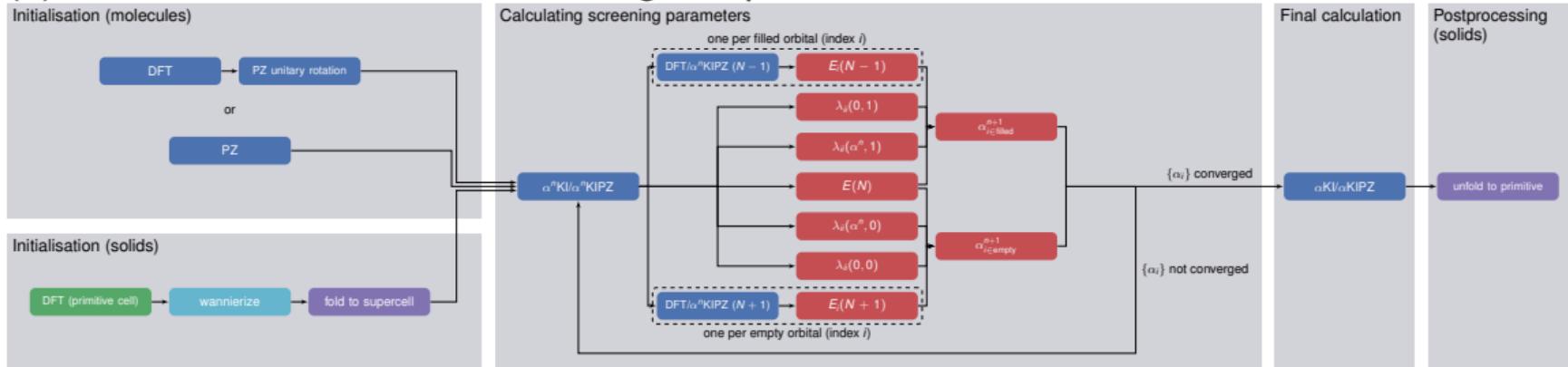


(b) DFPT in a primitive cell



The workflows

(a) finite difference calculations using a supercell



(b) DFPT in a primitive cell



(c) via machine learning

`kcw.x` (DFPT implementation) is distributed in Quantum ESPRESSO v7.1 onwards

`kcp.x` (supercell implementation) is available publically

`kcw.x`

`kcp.x`

`kcw.x` (DFPT implementation) is distributed in Quantum ESPRESSO v7.1 onwards

`kcp.x` (supercell implementation) is available publically

	<code>kcw.x</code>	<code>kcp.x</code>
available flavours	KI only	KI, KIPZ, pKIPZ, PZ

`kcw.x` (DFPT implementation) is distributed in Quantum ESPRESSO v7.1 onwards

`kcp.x` (supercell implementation) is available publically

	<code>kcw.x</code>	<code>kcp.x</code>
available flavours	KI only	KI, KIPZ, pKIPZ, PZ
screening parameters...	DFPT	finite differences

`kcw.x` (DFPT implementation) is distributed in Quantum ESPRESSO v7.1 onwards

`kcp.x` (supercell implementation) is available publically

	<code>kcw.x</code>	<code>kcp.x</code>
available flavours	KI only	KI, KIPZ, pKIPZ, PZ
screening parameters...	DFPT	finite differences
k -point sampling	explicit	implicit (via supercell)

`kcw.x` (DFPT implementation) is distributed in Quantum ESPRESSO v7.1 onwards

`kcp.x` (supercell implementation) is available publically

	<code>kcw.x</code>	<code>kcp.x</code>
available flavours	KI only	KI, KIPZ, pKIPZ, PZ
screening parameters...	DFPT	finite differences
k -point sampling	explicit	implicit (via supercell)
orbital minimization	uses MLWFs as a proxy (approximate)	explicit

`kcw.x` (DFPT implementation) is distributed in Quantum ESPRESSO v7.1 onwards

`kcp.x` (supercell implementation) is available publically

	<code>kcw.x</code>	<code>kcp.x</code>
available flavours	KI only	KI, KIPZ, pKIPZ, PZ
screening parameters...	DFPT	finite differences
k -point sampling	explicit	implicit (via supercell)
orbital minimization	uses MLWFs as a proxy (approximate)	explicit
scaling	$\mathcal{O}(N_q N_k N_{\text{orb}}^3)$	$\mathcal{O}(N_k^3 N_{\text{orb}}^3) = \mathcal{O}(N_k) \times$ primitive cell approach

How do I run these calculations?

Nevertheless, complicated workflows mean that...

How do I run these calculations?

Nevertheless, complicated workflows mean that...

- lots of different codes that need to handshake

How do I run these calculations?

Nevertheless, complicated workflows mean that...

- lots of different codes that need to handshake
- lots of scope for human error

How do I run these calculations?

Nevertheless, complicated workflows mean that...

- lots of different codes that need to handshake
- lots of scope for human error
- reproducibility becomes difficult

How do I run these calculations?

Nevertheless, complicated workflows mean that...

- lots of different codes that need to handshake
- lots of scope for human error
- reproducibility becomes difficult
- expert knowledge required

Nevertheless, complicated workflows mean that...

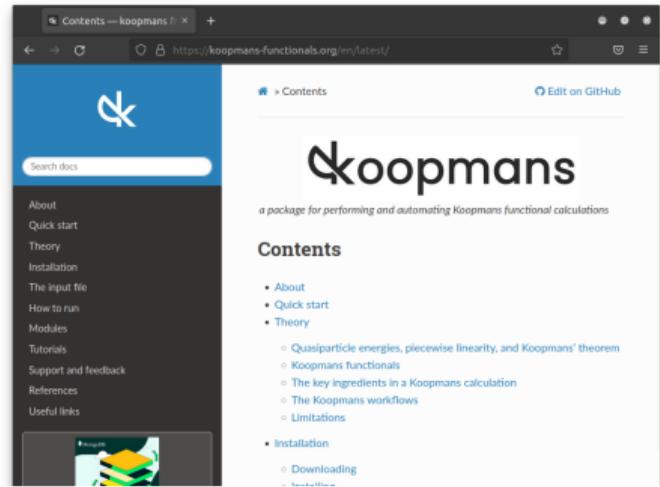
- lots of different codes that need to handshake
- lots of scope for human error
- reproducibility becomes difficult
- expert knowledge required

Our solution...

Koopmans

- v1.0 released earlier this year¹
- implementations of Koopmans functionals
- automated workflows
 - start-to-finish Koopmans calculations
 - Wannierisation
 - dielectric tensor
 - ...
- built on top of ASE²
- under the hood, calls Quantum ESPRESSO
- does not require expert knowledge

[koopmans-functionals.org](https://koopmans-functionals.org/en/latest/)



¹ E. Linscott et al. *Koopmans: An Open-Source Package for Accurately and Efficiently Predicting Spectral Properties with Koopmans Functionals*. 2023

² A. H. Larsen et al. *J. Phys. Condens. Matter* 29.27 (2017), 273002

koopmans: the input file

```
{  
    "workflow": {  
        "task": "singlepoint",  
        "functional": "ki",  
        "method": "dscf",  
        "init_orbitals": "mlwfs",  
        "alpha_guess": 0.1  
    },  
    "atoms": {  
        "atomic_positions": {  
            "units": "crystal",  
            "positions": [[{"Si": 0.00, 0.00, 0.00},  
                          {"Si": 0.25, 0.25, 0.25}]]  
        },  
        "cell_parameters": {  
            "periodic": true,  
            "ibrav": 2,  
            "celldm(1)": 10.262  
        }  
    },  
}
```

```
"k_points": {  
    "grid": [8, 8, 8],  
    "path": "LGXKG"  
},  
"calculator_parameters": {  
    "ecutwfc": 60.0,  
    "w90": {  
        "projections": [  
            [{"fsite": [0.125, 0.125, 0.125],  
             "ang_mtm": "sp3"}],  
            [{"fsite": [0.125, 0.125, 0.125],  
             "ang_mtm": "sp3"}]  
        ],  
        "dis_froz_max": 11.5,  
        "dis_win_max": 17.0  
    }  
}
```

koopmans is scriptable

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

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

# Define the projections for the Wannierization (same for filled and empty manifold)
si_proj = [{'fsite': [0.25, 0.25, 0.25], 'ang_mtm': 'sp3'}]
si_projs = ProjectionBlocks.from_list([si_proj, si_proj], atoms=atoms)

# Create the workflow
workflow = SinglepointWorkflow(atoms = atoms,
                                projections = si_projs,
                                ecutwfc = 40.0,
                                kpoints = Kpoints(grid=[8, 8, 8], path='LGXKG', cell=atoms.cell),
                                calculator_parameters = {'pw': {'nbnd': 10},
                                                        'w90': {'dis_froz_max': 10.6, 'dis_win_max': 16.9}})

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

Workflow

Workflow

atoms an ASE Atoms object

Workflow

`atoms` an ASE Atoms object
`calculations` a list of ASE calculators

Workflow

- `atoms` an ASE Atoms object
- `calculations` a list of ASE calculators
- `kpoints` a custom class containing k -point information

Workflow

`atoms` an ASE Atoms object

`calculations` a list of ASE calculators

`kpoints` a custom class containing k -point information

`pseudopotentials` a dictionary of pseudopotentials

Workflow

```
atoms  an ASE Atoms object
calculations  a list of ASE calculators
kpoints  a custom class containing k-point information
pseudopotentials  a dictionary of pseudopotentials
...

```

Workflow

`atoms` an ASE Atoms object

`calculations` a list of ASE calculators

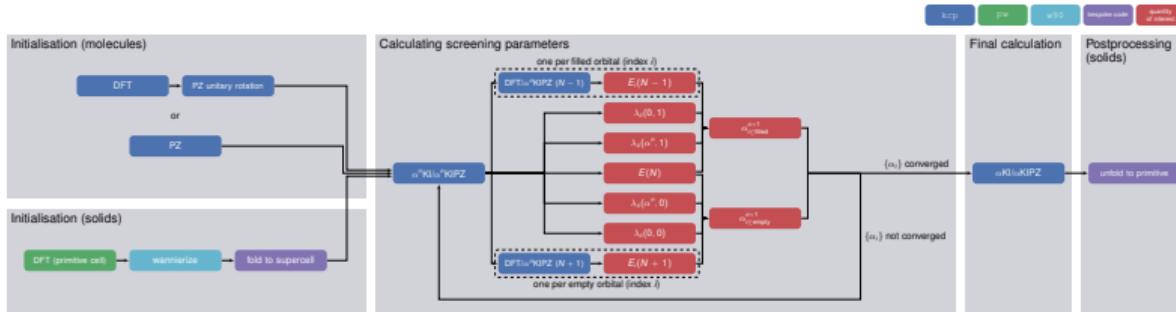
`kpoints` a custom class containing k -point information

`pseudopotentials` a dictionary of pseudopotentials

...

We will see examples in the hands-on!

Take home messages



- Koopmans functionals are more complicated than a simple semi-local DFT calculation, because of...
 - orbital-density-dependence
 - screening parameters
- Koopmans functionals are implemented in Quantum ESPRESSO
- the complexity of the workflows are handled by the koopmans package

koopmans: An Open-Source Package for Accurately and Efficiently Predicting Spectral Properties with Koopmans Functionals

Edward B. Linscott,*[△] Nicola Colonna,[△] Riccardo De Gennaro, Ngoc Linh Nguyen, Giovanni Borghi, Andrea Ferretti, Ismaila Dabo, and Nicola Marzari*



Cite This: <https://doi.org/10.1021/acs.jctc.3c00652>



Read Online

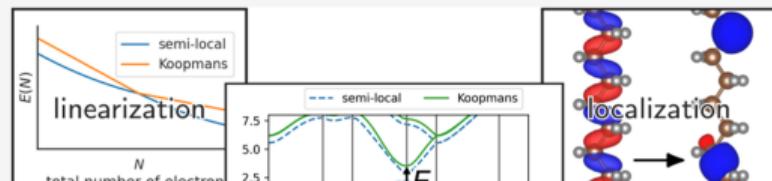
ACCESS |

Metrics & More

Article Recommendations

Supporting Information

ABSTRACT: Over the past decade we have developed Koopmans functionals, a computationally efficient approach for predicting spectral properties with an orbital-density-dependent functional framework. These functionals impose a generalized piecewise linearity condition to the entire electronic manifold, ensuring that



Acknowledgements



Nicola Marzari



Nicola Colonna



Riccardo De Gennaro



Yannick Schubert



Follow [@ed_linscott](https://twitter.com/ed_linscott) for updates | Slides available at github/elinscott

SPARE SLIDES

Recap from earlier

Key idea: construct a functional such that the *variational* orbital energies

$$\varepsilon_i^{\text{Koopmans}} = \langle \varphi_i | H | \varphi_i \rangle = \partial E_{\text{Koopmans}} / \partial f_i$$

are...

- independent of the corresponding occupancies f_i
- equal to the corresponding total energy difference $E_i(N - 1) - E(N)$

zero band gap \rightarrow occupancy matrix for variational orbitals is off-diagonal

Resonance with other efforts

- Wannier transition-state method of Anisimov and Kozhevnikov V. I. Anisimov et al. *Phys. Rev. B* 72.7 (2005), 075125
- Optimally tuned hybrid functionals of Kronik, Pasquarello, and others (refer back to Leeor's talk on Wednesday) L. Kronik et al. *J. Chem. Theory Comput.* 8.5 (2012), 1515; D. Wing et al. *Proc. Natl. Acad. Sci.* 118.34 (2021), e2104556118
- Ensemble DFT of Kronik and co-workers E. Kraisler et al. *Phys. Rev. Lett.* 110.12 (2013), 126403
- Koopmans-Wannier of Wang and co-workers J. Ma et al. *Sci. Rep.* 6.1 (2016), 24924
- Dielectric-dependent hybrid functionals of Galli and co-workers J. H. Skone et al. *Phys. Rev. B* 93.23 (2016), 235106
- LOSC functionals of Yang and co-workers C. Li et al. *Natl. Sci. Rev.* 5 (2018), 203