`OptaDOS`: Photoemission module
Version 2

Victor Chang, Bruno Camino, Nicholas M. Harrison

Department of Materials
Imperial College London
South Kensington
London, SW7 2AZ
UK,

Department of Chemistry
Imperial College London
White City Campus
London, W12 0BZ
UK,

and

The Institute for Molecular Science and Engineering
Imperial College London
London
UK

February, 2020

Chapter 1

Introduction

This photoemission module for OptaDOS is code design to generate angle-resolved photoemission spectrums from first principle simulations. The code was initially developed by Bruno Camino as part of his PhD thesis at Imperial College London and further developed by Victor Chang, also as part of his PhD thesis, both under the supervision of Professor Nicholas Harrison.

Chapter 2

Theoretical Background

2.1 Photoemission model

The formalism adopted in the photoemission model adopted here requires that the excitation spectrum can be described within a quasi-particle model with excitations represented as transitions between energy levels in a band structure of single particle states. The one-particle transition probability for the absorption of a photon of frequency ω is defined by Fermi's golden rule,

$$\Gamma_{i\to j} = \frac{2\pi}{\hbar} M(i,j,\mathbf{k},s)\delta(E(j,\mathbf{k},s) - E(i,\mathbf{k},s) - \hbar\omega) \tag{2.1}$$

where E is the electronic state eigenvalue, i and j are the indices for the initial and final state respectively, s is the spin index, $\mathbf{k}$ is the wavevector of the electron wavefunction, $M(i,j,\mathbf{k},s)$ is the photoemission matrix element and the Dirac delta function $\delta(E(j,\mathbf{k},s)-E(i,\mathbf{k},s)-\hbar\omega)$ ensure the energy conservation in the photoemission process. The Dirac delta function can be evaluated using the fixed-width Gaussian broadening, adaptive Gaussian broadening or linear extrapolative scheme implemented in the OptaDOS code.

The photoemission matrix element $M(i,j,\mathbf{k},\mathbf{s})$ in Eq. 2.1 is

$$M(i,j,\mathbf{k},s) = \left|\langle \psi_{\mathbf{k},j,s} \left| H' \right| \psi_{\mathbf{k},i,s}\rangle\right|^2 \tag{2.2}$$

The Fermi's golden rule describes the transition probability from an initial eigenstate ψ_i to a final state ψ_j as a result of a weak perturbation H' cause by the light. The perturbation due to the light H' can be evaluated by calculating the effect of the electromagnetic field on the electron (Eq. 2.3).

$$H' = \frac{e}{2m_0}(p\cdot A + A\cdot p) + \frac{e^2A^2}{2m_0} \tag{2.3}$$

where $p = -i\hbar\nabla$ is the momentum operator and here A is the polarization vector of the incoming electromagnetic wave. The A^2 term can be neglected under the assumption of a weak field. The Hamiltonian can be simplify due to the Coulomb gauge, $\nabla \cdot A = 0$. The perturbation due to the light field is rewritten as

$$H' = \frac{e}{2m_0} p \cdot A \tag{2.4}$$

The optical matrix from which the optical properties of a solid can be obtained.

2.1.1 Photoemission final state

Within the one-electron photoemission approach, two distinct photoemission models have been developed and widely used in previous work in order to describe the photoemission process. The current implementation generate two distinct optical matrices (Eq. 2.2) where the final state ψ_j changes according to the photoemission model.

- `Three-step photoemission model` : The three step model divides the photoemission process into three steps. In the first step, the ground state electrons are excited into a final state in the crystal or Bloch state

  $$\psi_j(\mathbf{r}) = \sum C_{j,\mathbf{G}} e^{i(\mathbf{k}+\mathbf{G})\mathbf{r}} \tag{2.5}$$

  Where $C_{j,\mathbf{G}}$ are the planewave coefficients and $\mathbf{G}$ are the reciprocal lattice vectors. In the second step, the excited Bloch electrons will have a probability to travel to the surface of the material. Once the electrons reach the surface of the material, in the third step, the electrons will escape into the vacuum.

- `One-step photoemission model` : The one step model final state in this work is described as

  $$\psi_j(\mathbf{r}) = e^{i\mathbf{k}\mathbf{r}} e^{-\frac{z sin(1-\theta(i,j,\mathbf{k}))}{\lambda(\omega)}} \tag{2.6}$$

  where $\lambda(\omega)$ is a frequency dependent scattering length and $\theta(i, j, \mathbf{k})$ consider the angular decay of the wavefunction inside of the material.

2.1.2 Atomic projection

It is possible to project the energy bands of a periodic solid onto localised atom centred orbitals and therefore to define atomic contributions to the excitation. This is particularly convenient as it allows, for instance, the contributions from excitation in each sub-surface layer to be computed. The contribution of each atomic layer that composed the slab can be performed by projecting the delocalised Bloch functions onto localised representations such as atomic orbitals (AO) or Wannier functions. CASTEP calculates this projection $W(i, j, \mathbf{k}, \mu, \mathbf{s})$ as

$$W(i, j, \mathbf{k}, \mu, \mathbf{s}) = \sum_{\nu} \mathbf{T}^*_{\mu\mathbf{ij},\mathbf{s}}(\mathbf{k}) \mathbf{T}_{\mu\mathbf{ij},\mathbf{s}}(\mathbf{k}) \mathbf{S}_{\nu\mu}(\mathbf{k})^{-\mathbf{1}} \tag{2.7}$$

where $T_{\mu ij,s}(\mathbf{k}) = \langle \Psi_{ij,s}(\mathbf{k}) \mid \phi_\mu(\mathbf{k}) \rangle$ are the overlap matrices between linear combination of atomic orbitals (LCAO) basis $\phi_\mu(\mathbf{k})$ and planewave state $\Psi_{ij,s}(\mathbf{k})$, and $S_{\nu\mu}(\mathbf{k}) = \langle \phi_\nu(\mathbf{k}) \mid \phi_\mu(\mathbf{k}) \rangle$ are the overlap matrices of the LCAO basis.

2.1.3 Temperature

The original formulation of the Kohn-Sham DFT describes a system of interacting particles in an external potential at zero temperature. Later, Mermin extended this formalism for a finite temperature $T > 0$. In the current implementation, the finite temperature formalism of Mermin is adopted by introducing a Fermi-Dirac distribution in which the state occupancy is

$$n(i, \mathbf{k}, T, s) = \frac{1}{e^{(E_F - E_i(\mathbf{k},s))/k_B T} + 1} \tag{2.8}$$

where E_F is the Fermi energy and k_B is the Boltzmann constant.

2.1.4 Light decay

The intensity of the light at each atom μ of the surface expressed as

$$I(\omega, \mu) = I(\omega, \mu - 1)e^{-\alpha(\omega,\mu)t(\mu)} - R(\omega, \mu) \tag{2.9}$$

where $\alpha(\omega, \mu)$ is the absorption coefficient at each atomic site, $t(\mu)$ is the length of the light path and $R(\omega, \mu)$ is the reflective coefficient of the surface. OptaDOS calculates the absorption coefficient $\alpha(\omega, \mu)$ and the reflective coefficient $R(\omega, \mu)$ from the dielectric function within the random phase approximation;

$$\epsilon_2(\omega) = \frac{\pi e^2}{\epsilon_0 \Omega} \int_{BZ} \frac{d\mathbf{k}}{(2\pi^3)} M(i, j) \delta[E(j, \mathbf{k}) - \mathbf{E}(\mathbf{i}, \mathbf{k}) - \hbar\omega] \tag{2.10}$$

where e is the electron charge, ϵ_0 is the permittivity of vacuum and Ω is the volume of the unit cell. The real part of the dielectric function ϵ_1 is calculated using the Kramers-Kronig relation. The absorption coefficient is

$$\alpha(\omega, \mu) = \frac{2k(\omega, \mu)\omega}{c} \tag{2.11}$$

and the reflective index is

$$R(\omega, \mu) = \frac{[n(\omega, \mu) - 1]^2 + k(\omega, \mu)^2}{[n(\omega, \mu) + 1]^2 + k(\omega, \mu)^2} \tag{2.12}$$

where n and k are calculated from the dielectric function as

$$n(\omega,\mu)^2 = \frac{1}{2}[\epsilon_1(\omega,\mu)^2 + \epsilon_2(\omega,\mu)^2]^{\frac{1}{2}} + \frac{1}{2}\epsilon_1(\omega,\mu) \tag{2.13}$$

and

$$k(\omega,\mu)^2 = \frac{1}{2}[\epsilon_1(\omega,\mu)^2 + \epsilon_2(\omega,\mu)^2]^{\frac{1}{2}} - \frac{1}{2}\epsilon_1(\omega,\mu) \tag{2.14}$$

2.1.5 Electron transport

Determining absorption in each sub-surface layer facilitates the use of an explicit model of the electron transport from the surface slab model simulation. In the current implementation a simple model of the inelastic scattering rate is used for the electron transport in the three step model and the escape probability of a free electron as it travels through a material in the one step model. The inelastic mean free path theory explains that electrons can only travel a certain average distance without suffering an inelastic scattering event. The assumption that once an electron suffers an inelastic scattering event the electron the electron will be reabsorbed is adopted in this work, resulting in an escape function for the $i \to j$ excitation as a function of the escape depth;

$$esc(\omega,i,j,\mathbf{k},\mu,s) = e^{-\frac{d(\mu)sin(1-\theta(i,j,\mathbf{k},s))}{\lambda(\omega)}} \tag{2.15}$$

The angle $\theta(i,j,\mathbf{k})$ indicates that electrons that travels with a higher angle with respect to the surface normal travels a longer distance inside of the material. This angle is calculated as

$$\theta(i,j,\mathbf{k},s) = acos\frac{E_{\|}(i,\mathbf{k},s)}{E_K(j,\mathbf{k},s)} \tag{2.16}$$

where $E_{\|}(i,\mathbf{k})$ is the transverse energy and $E_K(j,\mathbf{k})$ is the final state kinetic energy.

2.1.6 Parallel momentum conservation

The momentum of photons can be considered as negligible compared to the momentum of the surface electrons and therefore, the electron momentum is conserved upon excitation. In addition, since electrons can only travel very short distances inside of a material, photoemission is considered as a surface phenomenon. At the surface of a crystal, the periodic transverse symmetry perpendicular to the surface is broken and so the pseudo momentum perpendicular to the surface is not a conserved quantity. The parallel momentum conservation is described by

$$\Theta'(\mathbf{k}_\perp)\,\{\,1\ ,\mathbf{k}_\perp(i,j,s) \geq 0\delta((E_v - E(j,\mathbf{k}),s),\mathbf{k}_\perp(i,j,s) < 0 \tag{2.17}$$

where E_v is the vacuum level and

$$k_{\perp}(i,j,\mathbf{k},s) = \frac{2m}{\hbar^2}(E(j,\mathbf{k},s) - E_v) - \mathbf{k}_{\|}(i,\mathbf{k},\mathbf{s})^{\mathbf{2}} \tag{2.18}$$

Here $\mathbf{k}_{\|}(i,\mathbf{k},s)$ is the electron parallel momentum inside the crystal.

Θ' is a modified version of the Heavyside step function Θ in which 0 is replaced by a Dirac delta function $\delta(E_v - E_f)$ evaluated with a Gaussian function is included to account for finite temperature broadening of electrons that do not strictly fulfill the parallel momentum conservation.

$$\delta(E_v - E(j,\mathbf{k},s)) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{1}{2}\frac{E_v - E(j,\mathbf{k},s)}{\sigma^2}\right) \tag{2.19}$$

where the width σ can be used to approach external parameters such as the electron temperature.

2.1.7 Projected, bulk and total quantum efficiency and mean transverse energy

By considering Fermi's Golden rule (Eq. 2.1), the projection onto atomic orbitals (Eq. 2.7), temperature effects on the electronic population (Eq. 2.8), light decay (Eq. 2.9), electron scattering effects (Eq. 2.15) and the momentum conservation rules (Eq. 2.17), the QE in this model is computed as;

$$QE(\omega,\mathbf{k},\mu,T,s) = \sum_{i,j,s} \Gamma_{i\to j} W(i,j,\mathbf{k},\mu,s) n(i,j,T,s) I(\omega,\mu) esc(\omega,i,\mathbf{k},\mu,s) \Theta(\mathbf{k}_{\perp}) A^{-1} \tag{2.20}$$

where A is the surface area of the simulation cell. The atomic weighting of the contributions of the atomic sites allows the projected quantum efficiency to be defined as;

$$QE(\omega,\mu,T) = \int_{BZ} QE(\omega,\mathbf{k},\mu,T)\frac{d\mathbf{k}}{8\pi^3} \tag{2.21}$$

The accuracy of the computed surface photoemission can be limited by the computational resources. First principles simulation of surfaces is typically performed in using the slab model. In this model, a 2D periodic slab of material of finite thickness in the third dimension (a periodic "slab") separated from the repetitive images by a vacuum region is used to model a semi-infinite surface. The properties of the surface simulated using the slab model usually converges quickly with respect to the slab thickness and so only a small finite number of atomic layers is required. Sun and Ceder developed an efficient scheme for the creation and convergence of surface slabs that was able to converge the surface energy with slabs as thin as seven atomic layers. However, photoemitted electrons can be emitted from layers deeper inside the material, in the orders of nanometers (tens to hundreds of atomic layers).

Since the electronic properties of the slab in a suitable construction of a surface converges with respect to the slab thickness for a small finite number of atomic layers, the centre of the slab approximates as periodic bulk states. It is therefore possible to compute the photoemission efficiently from layers many nanometers from the surface by replicating bulk layers to simulate the photoemission from a slab model tof arbirtrary thickness. In the current implementation, the contribution of the deeper atomic layers to the photoemission process is approximated by the repetition of the atomic layer at the centre of the slab. The QE contribution of the bulk states is expressed as

$$QE_{bulk}(\omega, T) = \sum_{\mu=N}^{\tau} QE(\omega, \mu, T) \tag{2.22}$$

where N is the index of the atomic layer at the centre of the slab and

$$\tau = \frac{10\lambda(\omega)}{d(\mu)} \tag{2.23}$$

defines a cutoff in which the electrons excited from the deepest layer that has a probability of reaching the surface higher than 10^{-5} are considered to contribute to the bulk emission. Here $\lambda(\omega)$ is the inelastic mean free path (IMFP) constant and $d(\mu)$ is the thickness of the atomic layer.

The total quantum efficiency $QE_{tot}(\omega, T)$ is the sum of the explicitly computed N surface layers and the bulk contribution

$$QE_{tot}(\omega, T) = \sum_{\mu}^{N} QE(\omega, \mu, T) + QE_{bulk}(\omega, T) \tag{2.24}$$

The mean transverse energy

$$MTE(\omega, T) = \int_{BZ} \frac{\sum_{\mu}^{N} QE(\omega, \mathbf{k}, \mu, T) \frac{\hbar^2}{m_e} \mathbf{k}_{\|}{}^{\mathbf{2}}}{QE_{tot}(\omega, T)} \frac{d\mathbf{k}}{8\pi^3} \tag{2.25}$$

is the sum of the QE contribution at each atomic site point times the transverse energy at a particular $\mathbf{k}$ point normalized to the total quantum efficiency integrated over the Brillouin zone (BZ).

2.1.8 Field emission

In applications that involves photoemission such as the generation of free electrons for particle accelerators, an external electric field that enhances the photoemission properties of the photocathode is commonly used. The effect of this external electric field is included using an approximation based on the Fowler-Nordheim (FN) theory of field emission. The external electric field is assumed to be both uniform and to only modify the vacuum potential barrier, the electron distribution is in thermodynamic equilibrium and obeys Fermi-Dirac statistics and the emission surface is flat and planar with a constant uniform local work function. The vacuum potential in the FN theory in this current implementation is described using the Schottky-Nordheim barrier B_{SN},

$$B_{SN} = \phi - eFz - \frac{e^2}{16\pi\epsilon_0 z} \tag{2.26}$$

where e is the electron charge, ϵ_0 is the vacuum permittivity, ϕ is the potential barrier, F is the external electric field and z is the distance from the surface. The surface-vacuum potential barrier B_{SN} can be plotted as a function of the distance from the surface for different external electric fields.

Electrons tunnel through this barrier with a probability;

$$D \approx \frac{exp^{-G}}{1 + exp^{-G}} \tag{2.27}$$

where G is known as the strength of the barrier,

$$G = g_e \int B_{SN}^{1/2} dz \tag{2.28}$$

In the current implementation the integral in Eq. 2.28 is computed by numerical integration.

Chapter 3

Getting Started

3.1 Installation

OptaDOS is usually obtained in a gzipped tarball, `optados-X.X.tar.gz`. Extract this (`tar -xzf optados-X.X.tar.gz`) in the desired directory. Inside the `optados/` directory are a number of sub directories, `documents/`, `examples/` and `tools/`. The code may be compiled using the `Makefile` in the `optados/` directory. The `SYSTEM`, `BUILD`, `COMMS_ARCH` and `PREFIX` flags must be set, either in the `makefile.system`, or from the command line (for example `make BUILD=fast`).

3.1.1 `SYSTEM`

Choose which compiler to use to make OptaDOS. The valid values are:

- `g95` (default)
- `gfortran`
- `ifort`
- `nag`
- `pathscale`
- `pgf90`
- `sun`

3.1.2 `BUILD`

Choose the level of optimisations required when making OptaDOS. The valid values are:

- `fast` (default) All optimisations
- `debug` No optimisations, full debug information

3.1.3 COMMS_ARCH

Whether to compile for serial or parallel execution. The valid values are:

- `serial` (default)
- `mpi` (not tested)

3.1.4 PREFIX

Choose where to place the OptaDOS binary. The default is the OptaDOS directory.

3.2 Usage

```
optados.x86_64 [seedname]
```

- `seedname`: If a seedname string is given the code will read its input from a file `seedname.odi`. The default value is castep.

Chapter 4

Structure of the Module

- **calc_layers**: this subroutine identifies which atoms belong to each layer and which is the maximum one to include in the calculations (half slab). It contains:
  - a sorting algorithm
  - identification of the maximum layer
  - identification of the maximum atom
  - identification of how many atoms belong to each layer
- **schottky_effect**: this subrotuine calculates the effective WF, which is the WF of the material - the lowering due to the external electric field, which is read from the input: elec_field. The default is zero field.
- **identify_states**: this subroutine calculates the maximum band in the conduction band below the vacuum energy. It also calculates the number of transitions to the conduction band below and above the vacuum energy, this can be printed to check.
- **calc_electron_esc**: this subroutine calculates the escape length and the probability for the emission of the electron from a certain layer. At the moment, only the constant value is available.
- **make_pdos_weights_atoms**: this subroutine calculates the PDOS on atoms, by summing the orbital contributions of a certain atom. It is taken from the *pdos_merge* subroutine of pdos.F90, but only the sum on atoms is included. It contains:
  - the sum of the orbital contribution of the same atom
  - the sum of all atomic contribution at fixed **k** and band indices.
- **make_optical_weight**s: the task of this subroutine is calculating the optical matrix weights. This is done by weigthing the optical matrix by the components of the direction of the light, which are specified by optics_geom and optics_qdir parameters, see Section 5.6 (and 5.6.2) for details. For the photoemission calculation, the optics_geom can be
  - **polarized**: the three indices specify the plane of oscillation of the electric field of the incoming radiation;

  - **unpolarized**: the three indices define the plane perpendicular to the direction of the light.

  Tensor calculation is not implemented for photoemission calculations.

- **make_photo_weights**: this subroutine splits the optical_matrix_weights into atomic contributions by multiplying it with the pdos_weights_atoms (normalised with respect to the sum of the weights at that **k** point and band.
- **jdos_utils_calculate(matrix_weights, weighted_jdos)**: The photo.f90 module sends the matrix_weights calculated in make_photo_weights and receives the weighted_jdos. The last index of the optical_matrix_weights is *atom* for the photo.f90 module.
- **intraband_photo**: calculate the intraband contribution to the dielectric function.
- **calc_epsilon_2_photo**: this subroutine calculates the imaginary part of the dielectric function (epsilon_2) per atom, by mulptiplying the weighted_jdos (which depends on the atom index) with the epsilon constants. It also calulates the total epsilon_2 by summing all the atomic contributions.
- **calc_epsilon_1_photo**: this subroutine calculates the real part of the dielectric function (epsilon_1) through the Kramers-Kronig relations. First a refract_tmp is calculated, then a normalisation factor, which is the ratio between the epsilon_1 (non atom-by-atom) and finally, the sum of the atom-by-atom contribution is calculated.
- **calc_refract_photo**: this subroutine calculates the refraction coefficient.
- **calc_absorp_photo**: this subroutine calculates the absorption coefficient from the imaginary part of the refractive index.
- **calc_reflect_photo**:this subroutine calculates the reflection coefficient from the refractive index.
- **calc_absorp_layer**: this subroutine calculates the portion of light absorbed by each layer. If there is only one layer, the thickness is fixed to one.
- **write_optics**: Write the layer by layer optical properties in the OptaDOS output file.
- **calc_angle**: Calculate the photoemission angles theta/phi and the transverse energy.
- **QE calculation**: Calculates the layer-by-layer QE. Two options are available:
  - **calc_three_step_model**: Calculates the QE assuming a final Bloch state in the crystal.
  - **calc_one_step_model**: Calculates the QE assuming a final free electron state.
- **weighted_mean_te**: Weights the contribution to the transverse energy of each individual electron according to their QE.
- **Photoemission graphs**: Two options are available.
  - **transverse_energy_spread**: Plot the QE againts the transverse energy. A Gaussian broadening is applied.

- **binding_energy_spread**: Plot the QE againts the binding energy. A Gaussian broadening is applied. Additionally, the angles θ and ϕ can be set for better comparison.

Chapter 5

Parameters

5.1 `seedname.odi` File

The OptaDOS input file `seedname.odi` has a flexible free-form structure.

The ordering of the keywords is not significant. Case is ignored (so `smearing_width` is the same as `Smearing_Width`). Characters after !, or # are treated as comments. Most keywords have a default value that is used unless the keyword is given in `seedname.odi`. Keywords may be set in any of the following ways

```
smearing_width = 0.4

smearing_width :  0.4

smearing_width 0.4
```

A logical keyword can be set to `.true.` using any of the following strings: `T`, `true`, `.true.`.

5.2 General Parameters

5.2.1 `character(len=20) :: task`

Tells the code what to compute.

The specific option to run this module is:

- `photoemission`

5.2.2 `character(len=50) :: broadening`

Specified the scheme used to broaden a discrete sampling of the Brillouin Zone to a continuous spectral function.

The valid options for this parameter are:

- `adaptive` (default)
- `fixed`
- `linear` (recommended)

5.2.3 `logical :: legacy_file_format`

- `TRUE` Read CASTEP input compatible with versions < 6.0.
- `FALSE` (Default) Read CASTEP input compatible for use with CASTEP versions 6.0+ and generated with the castep spectral task.

5.2.4 `real(kind=dp) :: adaptive_smearing`

Set the relative smearing in the adaptive scheme.

Default value is 0.4

5.2.5 `real(kind=dp) :: fixed_smearing`

Smearing width for fixed broadening.

If `spectral_scheme` = `fixed` default value is 0.3eV.

5.2.6 `real(kind=dp) :: linear_smearing`

Smear the linear broadening with a Gaussian of this width.

Default value is 0.0.

5.2.7 `character(len=20) :: efermi`

Choose which Fermi energy to use.

The valid options for this parameter are:

- `optados` (default) OptaDOS recalculates the Fermi energy by performing a DOS calculation.
- `file` Take the value from the output of the ab-initio calculation.
- `insulator` Assume that the material is an insulator and counts filled bands to find the Fermi energy.
- `<real number>` User supplied value.

The default value is `optados`.

5.2.8 `logical :: finite_bin_correction`

Force each Gaussian to be larger than a single energy bin. (Useful for adaptive smearing and semi-core states when `numerical_intdos=TRUE`).

Default value `TRUE`.

5.2.9 `logical :: numerical_intdos`

Calculate the integrated dos by numerical integration instead of semi-analytically. (Useful for comparison with `LinDOS`.)

Default value `FALSE`.

5.2.10 `logical :: hybrid_linear`

Switch from linear broadening scheme to adaptive broadening when band gradient less than `hybrid_linear_grad_tol`. This allows for a good description of very flat bands such as defect and semi-core states. May also be used in conjunction with `finite_bin_correction` further improving the DOS and band energy

Default value `FALSE`.

5.2.11 `real(kind=dp) :: hybrid_linear_grad_tol`

Tolerance for switching from linear to adaptive broadening when using hybrid_linear option.

The default value is 0.01eV/Å.

5.3 Optics Parameters

5.3.1 `character(len=20) :: optics_geom`

Specifies the geometry for the optics calculation. Possible options:

- `polycrystalline` (Isotropic average)
- `polarized`
- `unpolarized`
- `tensor` (Full dielectric tensor)

The default is polycrystalline.

5.3.2 `real(kind=dp) :: optics_qdir(3)`

Direction of polarisation. Must be specified if `optics_geom = polarized` or `optics_geom = unpolarized`.

There is no default value

5.3.3 `logical :: optics_intraband`

If true, the intraband contribution to the dielectric function will be calculated. (Important for metals.)

The default is FALSE.

5.3.4 `real(kind=dp) :: optics_drude_broadening`

Value of broadening included in the Drude term expressed in s^{-1}.

The default value is 1E-14.

5.3.5 `real(kind=dp) :: optics_lossfn_broadening`

FWHM of Gaussian used to broaden the loss function.

The default value is 0 (*i.e.* no broadening is used).

5.4 Photoemission Parameters

5.4.1 `logical :: photo_model`

Set the final state for the photoemitted electrons.

- `3step` Read the matrix elements assuming a conduction band final state in the solid.
- `1step` ead the matrix elements assuming a free electron final state.

5.4.2 `logical :: momentum`

Set the parallel momentum of the photoemitted electrons.

- `crystal` Takes the crystal momentum k as value to compute the electron parallel momentum.
- `kp` Takes the electron velocity and effective mass of an electron at a particular k point and band from an external file and computes the electron momentum.

5.4.3 `logical :: photo_output`

- `te` :: Plot QE vs Transverse energy spread – It is useful for photocathodes for FELs, since the emittance can be extracted from this graph.
- `be` :: Plot QE vs Binding energy – It is useful for ARPES. Additionally, you have to specify the angles phi and theta by defining 4 additional parameters (lower limit and upper limit)
  - `phi_upper` :: Set the upper limit for the phi angle in the binding energy output.
  - `phi_lower` :: Set the lower limit for the phi angle in the binding energy output.
  - `theta_upper` :: Set the upper limit for the theta angle in the binding energy output.
  - `theta_lower` :: Set the lower limit for the theta angle in the binding energy output.

5.4.4 `real(kind=dp) :: s_area`

Set the area of the simulation cell surface.

5.4.5 `real(kind=dp) :: slab_volume`

Set the volume of the simulation slab.

5.4.6 `real(kind=dp) :: elec_field`

Set the external electric field in V/m.

Chapter 6

Examples

This is an example of using photoemission module for calculating the photoemission properties of Cu(111) surface using a 16 atomic layers slab.*Input Files*:

- – `examples/Cu_photo/Cu.cell` - The CASTEP cell file containing information about the simulation cell.
  – `examples/Cu_photo/Cu.param` - The CASTEP param file containing information about the parameters for the SCF and spectral calculations.
  – `examples/Cu_photo/Cu.odi` - The OptaDOS input file, containing the parameters necessary to run OptaDOS.

1. Perform a CASTEP calculation using the `Cu.cell` and `Cu.param` input files. Slab calculations are time consuming calculations, a high performing computer should be used.

   ```
   $ castep Cu
   ```

   More help can be found in the tutorials on the CASTEP website `www.castep.org`.

2. Perform an OptaDOS calculation.

   ```
   $ optados.<SYSTEM>.<BUILD>.<COMMS_ARCH>.x86_64 Cu
   ```

   This generates several files:

   - `Cu.odo` – OptaDOS general output file.
   - `Cu_be.dat` – The gaussian broadened angle-resolved beinding energy spectra raw output data.

3. Open the `Cu.odo` file in a text editor (*e.g.* vi or emacs).

   Output file (truncated):

```
 +============================================================================+
+                             Photoemission Calculation                       +
+============================================================================+
|                                                                            |
+-------------------------- Setting Fermi Energy  -------------------------+
```

```
+----------------------------- Fermi Energy Analysis ------------------------+
| From Linear broadening                                                     |
|   Spin Component : 1 occupation between  175.95555 and  176.00289   <- Occ |
|         Fermi energy (Linear broadening) :   0.7883 eV             <- EfL |
+----------------------------------------------------------------------------+
|                      Fermi energy from DOS :   0.7883 eV             <- EfD |
+----------------------------------------------------------------------------+

+------------------------------- Atomic Order  ------------------------------+
| Atom |  Atom Order  |   Layer   |         Atomic Position (Angs)           |
|  cu         1            1                  13.326149909131798             |
|  cu         2            2                  11.593149920948759             |
|  cu         3            3                  9.8006999331710976             |
|  cu         4            4                  8.0101999453801387             |
|  cu         5            5                  6.2310999575114456             |
|  cu         6            6                  4.4502499696546858             |
|  cu         7            7                  2.6706499817894018             |
|  cu         8            8                 0.89019999392991433             |
|  cu         9            9                -0.89019999392991433             |
|  cu        10           10                 -2.6706499817894018             |
|  cu        11           11                 -4.4502499696546858             |
|  cu        12           12                 -6.2310999575114456             |
|  cu        13           13                 -8.0101999453801387             |
|  cu        14           14                 -9.8006999331710976             |
|  cu        15           15                 -11.593149920948759             |
|  cu        16           16                 -13.326149909131798             |
+----------------------------------------------------------------------------+
|  Max number of atoms:           8    Max  number of layers:           8    |
+----------------------------------------------------------------------------+
+---------------------------- Photoemission ---------------------------------+
+----------------------------------------------------------------------------+
| Work Function         4.5560 eV          Photon Energy        4.7000 eV   |
| Effective Work Function    4.4360   eV        Electric field    0.0010V/A  |
| Final state : Bloch state                                                  |
+----------------------------------------------------------------------------+
| Atom |  Atom Order  |   Layer   |            Quantum Efficiency            |
|  cu         1            1                  6.4323462534403726E-006        |
|  cu         2            2                  0.0000000000000000             |
|  cu         3            3                  4.6827844145536271E-007        |
|  cu         4            4                  1.0551651594541037E-006        |
|  cu         5            5                  1.9117009775645270E-006        |
|  cu         6            6                  1.6248016213239814E-006        |
|  cu         7            7                  1.0332973207922402E-006        |
|  cu         8            8                  4.1019373536807091E-007        |
|  Bulk                                       2.1753811814371912E-006        |
|  Total quantum efficiency (electrons/photon):   1.5111164690835850E-005    |
|  Weighted mean transverse energy (eV):   2.3076853605907301E-002           |
|  Total field emission (electrons/A^2):   1.7679531256450310E-075           |
+----------------------------------------------------------------------------+
End of execution
|                                                                            |
  + Time to calculate Photoemission                            93.607 (sec) +
+============================================================================+
```

Layer-by-layer decomposition of the Quantum efficiency, the total quantum efficiency and the weighted mean transverse.

Plotting the gaussian broadened angle-resolved binding energy spectra,

```
xmgrace -nxy Cu_be.dat
```

Appendix A

Interface with other codes

Currently OptaDOS is interfaced with castep and onetep. This appendix consists of Fortran95 code which defines the input files for OptaDOS so as to aid its interfacing with other electronic structure codes. Presented below are the type declaration statements and write statements that may be used in other electronic structure codes or converters to generate the .bands, .ome_bin, .pos_bin and .elnes_bin files used as inputs to OptaDOS. These code snippets are also available in the `tools/` directory of the OptaDOS distribution.

A.1 `.ome_bin` file

The `.ome_bin` file is required for adaptive and linear broadening, and for optics calculations. Note that a `.dome` file is just the diagonal elements of the `.ome`. The `.ome_bin` file is an unformatted file.

```
integer,parameter:: dp=selected_real_kind(15,300) ! Define double precision
real(dp):: file_version=1.0_dp     ! File version
character(len=80):: file_header    ! File header comment
integer:: num_kpoints              ! Number of k-points
integer:: num_spins                ! Number of spins
integer:: max_eigenv               ! Number of bands included in matrix elements
integer:: num_eigenvalues(1:num_spins)   ! Number of eigenvalues per spin channel
complex(dp):: optical_mat(max_eigenv,max_eigenv,1:3,1:num_kpoints,num_spins)! OMEs

write(ome_unit) file_version
write(ome_unit) file_header

do ik=1,num_kpoints
 do is=1,num_spins
  write(ome_unit) (((optical_mat(ib,jb,i,ik,is),ib=1,num_eigenvalues(is)), &
       &jb=1,num_eigenvalues(is)),i=1,3)
 end do
end do
```

A.2 .fome_bin file

The .fome_bin file is required for the one step model. Note that a .dome file is just the diagonal elements of the .fome. The .fome_bin file is an unformatted file.

```
integer,parameter:: dp=selected_real_kind(15,300) ! Define double precision
real(dp):: file_version=1.0_dp     ! File version
character(len=80):: file_header    ! File header comment
integer:: num_kpoints              ! Number of k-points
integer:: num_spins                ! Number of spins
integer:: max_eigenv               ! Number of bands included in matrix elements
integer:: num_eigenvalues(1:num_spins)   ! Number of eigenvalues per spin channel
complex(dp):: foptical_mat(max_eigenv,max_eigenv,1:3,1:num_kpoints,num_spins)! OMEs

write(fome_unit) file_version
write(fome_unit) file_header

do ik=1,num_kpoints
 do is=1,num_spins
  write(ome_unit) (((foptical_mat(ib,jb,i,ik,is),ib=1,num_eigenvalues(is)+1), &
       &jb=1,num_eigenvalues(is)+1),i=1,3)
 end do
end do
```

A.3 .pdos_bin file

The .pdos_bin file is required for PDOS calculations. The .pdos_bin file is an unformatted file.

```
integer,parameter:: dp=selected_real_kind(15,300) ! Define double precision
integer:: num_kpoints            ! Number of k-points
integer:: num_spins              ! Number of spins
integer:: num_popn_orb           ! Number of LCAO projectors
integer:: max_eigenv             ! Number of bands included in matrix elements
real(dp):: file_version=1.0_dp   ! File version
integer:: species(1:num_popn_orb)! Atomic species associated with each projector
integer:: ion(1:num_popn_orb)    ! Ion associated with each projector
integer:: am_channel(1:num_popn_orb)     ! Angular momentum channel
integer:: num_eigenvalues(1:num_spins)   ! Number of eigenvalues per spin channel
real(dp):: kpoint_positions(1:num_kpoints,1:3) ! k_x, k_y, k_z in fractions of BZ
real(dp):: pdos_weights(1:num_popn_orb,max_eigenv,num_kpoints,num_spins)!Matrix elements
character(len=80):: file_header ! File header comment

write(pdos_file) file_header
write(pdos_file) file_version
write(pdos_file) num_kpoints
write(pdos_file) num_spins
write(pdos_file) num_popn_orb
write(pdos_file) max_eigenv
write(pdos_file) species(1_:num_popn_orb)
write(pdos_file) ion(1:num_popn_orb)
write(pdos_file) am_channel(1:num_popn_orb)

do nk=1,num_kpoints
 write(pdos_file) nk, kpoint_positions(nk,:)
 do ns = 1,num_spins
  write(pdos_file) ns
  write(pdos_file) num_eigenvalues(ns)
  do nb = 1,num_eigenvalues(ns)
   write(pdos_file) real(pdos_weights(num_popn_orb,nb,nk,ns))
  end do
 end do
end do
```

A.4 .dome_bin file

The .dome_bin file contains the first derivative of a band with respect to k. Also known as electron velocity.

The .dome_bin file is an unformatted file.

```
integer,parameter:: dp=selected_real_kind(15,300) ! Define double precision
real(dp):: file_version=1.0_dp     ! File version
character(len=80):: file_header    ! File header comment
integer:: num_kpoints              ! Number of k-points
integer:: num_spins                ! Number of spins
integer:: max_eigenv               ! Number of bands included in matrix elements
integer:: num_eigenvalues(1:num_spins)   ! Number of eigenvalues per spin channel
real(dp):: band_gradient(max_eigenv,1:3,1:num_kpoints,num_spins)!

write(gradient_unit) file_version
write(gradient_unit) file_header

do ik=1,num_kpoints
 do is=1,num_spins
  write(gradient_unit) ((band_gradient(ib,i,ik,is),ib=1,num_eigenvalues(is)),i=1,3)
 end do
end do
```

A.5 `.ddome_bin` file

The `.ddome_bin` file contains the second derivative of a band with respect to k. Also known as effective mass.

The `.ddome_bin` file is an unformatted file.

```
integer,parameter:: dp=selected_real_kind(15,300) ! Define double precision
real(dp):: file_version=1.0_dp     ! File version
character(len=80):: file_header    ! File header comment
integer:: num_kpoints              ! Number of k-points
integer:: num_spins                ! Number of spins
integer:: max_eigenv               ! Number of bands included in matrix elements
integer:: num_eigenvalues(1:num_spins)   ! Number of eigenvalues per spin channel
real(dp):: band_curvature(max_eigenv,1:3,1:3,1:num_kpoints,num_spins)! OMEs

write(curvature_unit) file_version
write(curvature_unit) file_header

do ik=1,num_kpoints
 do is=1,num_spins
  do ib = 1,nbands
   do i = 1,3
    do j= 1,3
       write(curvature_unit) optical_mat(ib,i,j,ik,is)
     end do
    end do
   end do
 end do
end do
```