Documentation

Authors: Hanwen Jin, Simão Meneses João, Johannes Lischner

November 7, 2022

This document contains the documentation of our hot carrier generation code, as well as an explanation of the methods involved and physical interpretation of the objects being used.

Contents

Part I
Overview

1 Introduction

Metallic nanoparticles have a very special interaction with light. When light shines on a metal, conduction electrons oscillate back and forth driven by the electric field. This non-equilibrium configuration excites electrons into more energetic unoccupied states creating a very energetic electron-hole pair. These carriers can be very energetic (hot carriers) and can be harnessed for many applications. Plasmonic photocatalysis is one such example.

2 Fermi Golden Rule

Fermi's Golden Rule provides the rate of transitions to states with energy E due to a perturbation.

$$N_e\left(E,\omega\right)=\frac{2}{V}\sum_{if}\Gamma_{if}\left(\omega\right)\delta\left(E-E_f\right)$$

where

$$\Gamma_{if}=\frac{2\pi}{\hbar}\left|\left\langle f\left|\Phi_{\text{tot}}\left(\omega\right)\right|i\right\rangle\right|^2\delta\left(E_f-E_i-\hbar\omega\right)f\left(E_i\right)\left(1-f\left(E_f\right)\right).$$

$N_e\left(E,\omega\right)dE$ is interpreted as the number of transitions per unit time with energy between E and $E+dE$ due to a perturbation of frequency ω.

Comment on units

In this document, these are the units being used

- Energy: eV
- Voltage: Volts
- Length: nanometers
- Time: second

So the physical constants have the following values:

- Planck's constant: $\hbar=6.582\times10^{-16}\text{eV s}=2.42\times10^{-17}\text{Ha s}$

For reference, the electric field E_{ref} used in experiments is such that the illumination intensity is $I=1\text{mW}/\mu\text{m}^2$, so $E_{\text{ref}}=8.7\times10^5\text{Volt/m}=8.7\times10^{-4}\text{Volt/nm}$. For comparison, the average solar intensity on Earth's surface is 1360W/m^2, which is about 10^6 times weaker!

Electrostatic potential

The electric potential in the quasistatic approximation is found using COMSOL by defining a geometry and providing the dielectric constant at a certain frequency ω. COMSOL then provides the electrostatic potential at each point in space as a result. See the next section for more details.

Calculation of the Fermi Golden Rule

To understand what the code is doing, it is useful to understand the mathematical operations that are required in order to compute $N_e\left(E,\omega\right)$. Since diagonalization is very hard to do for large matrices, we need to express N_e in a different way in order to use a different set of methods. Let's begin by introducing two integrals over the energy in order to capture all the eigen-energies with Dirac deltas:

$$N_e\left(E,\omega\right)=\frac{2}{V}\sum_{if}\int_{-\infty}^{\infty}d\varepsilon\int_{-\infty}^{\infty}d\varepsilon'\delta\left(\varepsilon-E_i\right)\delta\left(\varepsilon'-E_f\right)\Gamma_{if}\left(\omega\right)\delta\left(E-E_f\right)$$

Now we just need to replace the eigen-energies:

$$N_e(E,\omega) = \frac{4\pi}{\hbar V}\int_{-\infty}^{\infty} d\varepsilon \int_{-\infty}^{\infty} d\varepsilon' f(\varepsilon)(1-f(\varepsilon'))\,\delta(E-\varepsilon')\,\delta(\varepsilon'-\varepsilon-\hbar\omega)\sum_{if}\left|\langle f|\Phi_{\text{tot}}(\omega)|i\rangle\right|^2\delta(\varepsilon-E_i)\,\delta(\varepsilon'-E_f)$$

which motivates us to define the new object

$$\Phi_\omega(\varepsilon,\varepsilon') = \sum_{if}\left|\langle f|\Phi_{\text{tot}}(\omega)|i\rangle\right|^2\delta(\varepsilon-E_i)\,\delta(\varepsilon'-E_f).$$

This object is a reinterpretation of the potential matrix element in energy space. Instead of telling us the matrix element of the initial state i with final state f, it tells us the matrix element of initial state of energy ε with final state of energy ε'. It can be cast into a basis-independent form suitable for Chebyshev expansions:

$$\Phi_\omega(\varepsilon,\varepsilon') = \text{Tr}\left[\Phi_{\text{tot}}(\omega)\,\delta(\varepsilon-H)\,\Phi_{\text{tot}}^\dagger(\omega)\,\delta(\varepsilon'-H)\right].$$

Plugging this back into the expression for the hot-carrier generation rate, we get

$$N_e(E,\omega) = \frac{4\pi}{\hbar V}\int_{-\infty}^{\infty} d\varepsilon \int_{-\infty}^{\infty} d\varepsilon' f(\varepsilon)(1-f(\varepsilon'))\,\delta(E-\varepsilon')\,\delta(\varepsilon'-\varepsilon-\hbar\omega)\,\Phi_\omega(\varepsilon,\varepsilon').$$

$\Phi_\omega(\varepsilon,\varepsilon')$ is simply a function of two energies, so $N_e(E,\omega)$ can be obtained through a straightfoward numerical double integral. This form also makes it easier to interpret what is happening to $\Phi_\omega(\varepsilon,\varepsilon')$ during the integration process:

1. the product of Fermi functions $f(\varepsilon)(1-f(\varepsilon'))$ ensures that only initial states below the Fermi energy and final states above the Fermi energy contribute to this integral. A finite temperature relaxes this restriction slightly.

2. The second Dirac delta $\delta(\varepsilon'-\varepsilon-\hbar\omega)$ is a consequence of the interaction with the electric field of frequency ω. The electric field only causes transitions which change the energy by $\hbar\omega$. In practice, there are many mechanisms going on inside the nanoparticle, which slightly violate this restriction, so the Dirac delta is instead approximated by a gaussian of a certain width γ, denoted by δ_γ

3. The first Dirac delta $\delta(E-\varepsilon')$ means that only final states which have energy E are to be considered. Like before, this Dirac delta can also be approximated by a gaussian of a certain (different) width λ, representing the limited resolution of the measuring apparatus. If this is ignored, we have infinite resolution, and the definition of the Dirac delta can be used to simplify the integral in ε'. This is what is done in the code (but see next section - a finite number of Chebyshev polynomials effectively translates into a finite resolution anyway).

With point 3 in mind, the hot carrier generation rate can be simplified to

$$N_e(E,\omega) = \frac{4\pi}{\hbar V}(1-f(E))\int_{-\infty}^{\infty} d\varepsilon f(\varepsilon)\,\delta_\gamma(E-\varepsilon-\hbar\omega)\,\Phi_\omega(\varepsilon,E). \tag{1}$$

The final ingredient to understand the inner workings of the code is the Chebyshev expansion the Dirac delta operators inside of $\Phi_\omega(\varepsilon,\varepsilon')$. These operators can be expanded in a series of Chebyshev polynomials

$$\delta(\varepsilon-H) = \sum_{n=0}^{M-1}\Delta_n(\varepsilon)\,T_n(H)$$

where

$$\Delta_n(\varepsilon) = \frac{2}{\delta_{n,0}+1}\frac{T_n(\varepsilon)}{\pi\sqrt{1-\varepsilon^2}}g_J^{n,M}$$

is the coefficient of the expansion regularized by the Jackson weight g_J. Thus, $\Phi_\omega(\varepsilon,\varepsilon')$ is expressed as

$$\Phi_\omega(\varepsilon,\varepsilon') = \text{Tr}\left[\Phi_{\text{tot}}(\omega)\left(\sum_{n=0}^{M-1}\Delta_n(\varepsilon)\,T_n(H)\right)\Phi_{\text{tot}}^\dagger(\omega)\left(\sum_{m=0}^{M-1}\Delta_m(\varepsilon')\,T_m(H)\right)\right]$$

and can be calculated in two steps.

Step 1: Calculation of the Chebyshev moments

Gathering all the operators together, we get the Chebyshev moments

$$\mu_{nm} = \mathbf{Tr}\left[\Phi_{\mathbf{tot}}(\omega)\, T_n(H)\, \Phi^{\dagger}_{\mathbf{tot}}(\omega)\, T_m(H)\right]$$

which can be calculated very efficiently using the Chebyshev polynomials' recursion properties.

Step 2: Resummation of the expansion

What's left defines the optical matrix element $\Phi_\omega(\varepsilon, \varepsilon')$ in terms of the Chebyshev moments

$$\Phi_\omega(\varepsilon, \varepsilon') = \sum_{n=0}^{M-1}\sum_{m=0}^{M-1} \Delta_n(\varepsilon)\, \Delta_m(\varepsilon')\, \mu_{nm}.$$

Furthermore, considering $\Delta_n(\varepsilon)$ as a matrix $\Delta_{n\varepsilon}$, this reduces to a triple matrix product

$$\Phi_\omega(\varepsilon, \varepsilon') = \left[\Delta^T \mu \Delta\right]_{\varepsilon\varepsilon'}$$

Once these two steps have been completed, the hot-carrier generation rate can be found by doing the final step of integrating $\Phi_\omega(\varepsilon, \varepsilon')$ according to eq. 1. These three steps (1 - moments, 2 - resummation, 3 - integration) each represent three important parts of the workflow.

3 Workflow

4 Interpretation of the results

Part II
Technical information

5 Calculating the electric potential: Comsol

To find the electric potential inside the nanoparticle, we use Comsol to generate a simulation box representing the boundary conditions. We assume that the cube has side length L and is centered at the origin. In order to generate a uniform electric field inside the nanoparticle, we impose boundary conditions $V(z) = E_0 z$ on the faces of the cube. If z is in units of nm and $E_0 = 1$ Volt/nm, then V is in units of Volt. Finally, to convert from an electric potential to an electric potential energy $\Phi_0 = |e| V_0 = 1\text{eV}$.

With these boundary conditions, we obtain an electric potential $V(\mathbf{r})$ which depends nontrivially on the position because of the dielectric properties of the nanoparticle. Because of the boundary conditions imposed, away from the nanoparticle, we expect $V(|\mathbf{r}| \to \infty) = E_0 z$. If the simulation box is sufficiently large, we expect the result not to depend on its size.

5.1 Rescaling the electric field

By design, the electric field inside the simulation box is always $E_0 = 1$ V/nm. To analyze other electric fields, we make use of the linarity of Maxwell's equations: if the electric field is twice as strong, then the induced electrostatic potential (in the quasistatic approximation) is twice as strong as well. Therefore, we can state that for a generic electric field strength E, the corresponding potential V_E is

$$V_E(\mathbf{r}) = \frac{E}{E_0} V_{E_0}(\mathbf{r})$$

so we can always use the default value of E_0 so long as we rescale it back to E in the end. With this in mind, the electric potential energy term inside the expression for N_e (eq. 1) can always be calculated with the default electric field in Comsol. Let Φ^0 be the energy-resolved optical matrix with this potential. Then

$$N_e(\varepsilon, \omega) = \frac{4\pi}{\hbar V} \left(\frac{E}{E_0}\right)^2 (1 - f(\varepsilon)) \int_{-\infty}^{\infty} d\varepsilon f(\varepsilon) \delta_\gamma (E - \varepsilon - \hbar\omega) \Phi^0_\omega(\varepsilon, \varepsilon).$$

5.2 Rescaling the nanoparticle

Suppose that we want to find the electric potential inside several identically-shaped nanoparticles, but with different sizes. For example, cubic nanoparticles of different side lengths. We can of course use larger simulation boxes for larger particles in order to mitigate boundary effects, but there's a trick that can be used: rescale the nanoparticles into having identical sizes and rescale the value of the electric potential by the same amount.

For example, suppose we want to find the electric potential inside a nanocube of length $\ell = 3$nm and another of length 20nm. Let's further assume that we have already found the solution $V_{\ell=3}(\mathbf{r})$ for all positions inside the smaller nanocube. Then, $V_{\ell=3}(\mathbf{r}) = \frac{20}{3} V_{\ell=20}\left(\frac{3}{20}\mathbf{r}\right)$. More generally, any two potential profiles are related through

$$a V_{\ell=a}\left(\frac{\mathbf{r}}{a}\right) = b V_{\ell=b}\left(\frac{\mathbf{r}}{b}\right).$$

This is what is used in our implementation.

5.3 Using Comsol

Details on using Comsol

6 Chebyshev method

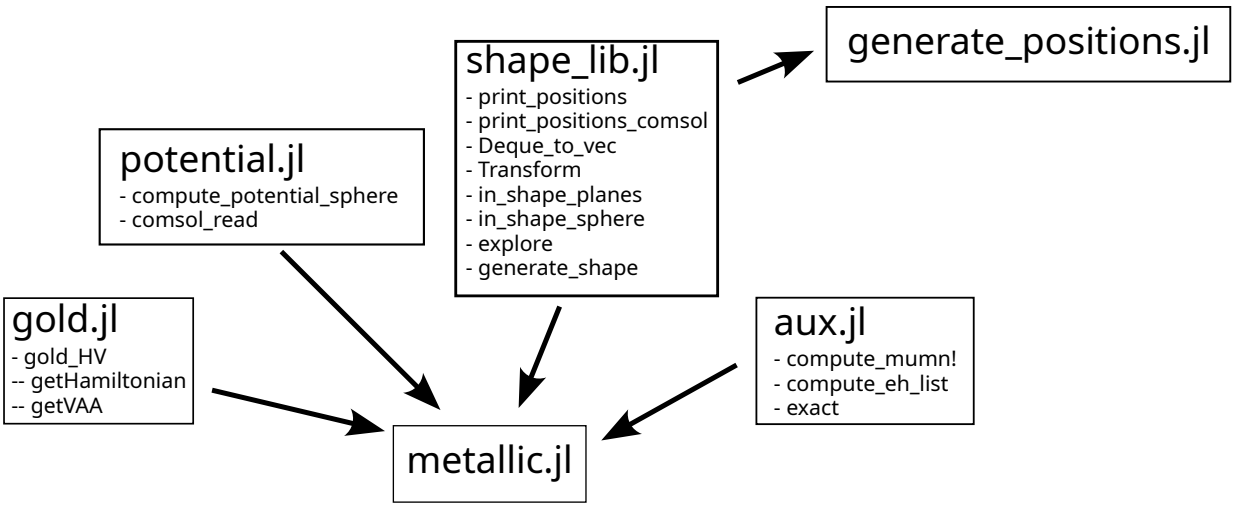

Figure 1: Diagram showing how the code is organized.

Part III
Code

7 Code organization

This program consists of a main code written in Julia (.jl) and several auxiliary programs in Python.

`shapelib.jl`

Contains several functions to help with the generation of the nanoparticles' shapes and atomic positions. `shapelib.jl` .

`print_positions_comsol(R, filename, rescale)` - prints out the atomic positions (`R`) to a text file (`filename`) which can be read by `COMSOL`. `COMSOL` will then evaluate the electric potential at these positions. This file already has the `COMSOL` header. If `rescale` is set to true, then all the atomic positions will be rescaled to fit within a cube of length 1.5nm. See the section about `COMSOL` for the reason behind this.

isgenerate_positions.jl -

shape_lib.jl