DETERMINATION OF THE DIELECTRIC RESPONSE FUNCTION USING EELS

by

M.F.S. Zwanenburg

in partial fulfillment of the requirements for the degree of

Bachelor of Science
in Applied Physics

at the Delft University of Technology,
to be defended publicly on Thursday July 18, 2019 at 3:00 P.M.

Supervisor:	Dr. Sonia Conesa Boj
	MSc. Louis Maduro
Thesis committee:	Dr. Toeno van der Sar

TU Delft Delft University of Technology

ABSTRACT

Electron energy-loss spectroscopy is used to derive the complex dielectric response function of a specimen. A script has been developed to obtain the dielectric response function from the energy-loss spectrum, and has been tested on numerous data sets, with different specimen thickness.

The script proved to be a fast and straight-forward tool to obtain the dielectric response function from the energy-loss spectrum, and to calculate the optical properties of the specimen. However, when the specimen thickness decreased and surface plasmon contributions dominated the energy-loss spectrum, the script failed to provide an accurate and reliable description of the dielectric response function.

The script only takes into account bulk and surface plasmon contributions to the energy-loss spectrum, and assumes normal incidence for the incident electrons. When other effects, such as band transitions, add a significant contribution to the energy-loss spectrum, the result of the script will contain an error as these contributions are not accounted for properly.

The script is initialised with the assumption that the entire energy-loss spectrum is bulk contribution. When the specimen is very thin, and the energy-loss spectrum is dominated by surface losses, this initialisation is not appropriate and the obtained dielectric response function is incorrect.

Contents

1

INTRODUCTION

When an everyday object, for example a piece of wood, is cut in half, its properties will not change very significantly: it will not all of a sudden become transparent, or become a conductor. This is definitely not the case when working with materials on a nanoscale. For example, 2 monolayers of MoS_2 may exhibit noticeable different properties than 4 monolayers of MoS_2 [1] [2]. The study of the properties of these materials have become a large field of research during the past decade [3]. One of the studies on these kinds of materials, is the investigation of their electronic and optical properties, which provide information about the interaction the specimen undergoes with light of different frequencies. In this thesis, a script is produced to obtain the dielectric response function from the energy-loss spectrum of nanomaterials. From the dielectric response function, the optical properties of the specimen can be determined.

In scanning transmission electron microscopy (STEM), electrons are transmitted from an electron gun at energies typically between 100 and 300 keV. When the specimen is less than 1 μm thick, these electrons will pass completely through the specimen [4]. In electron energy-loss spectroscopy (EELS) in a STEM, the energy-losses of the incident electrons are studied after they have passed through the specimen. One of the prominent peaks in the energy-loss spectrum, is the plasmon peak. This peak appears in the low-loss region of the spectrum, typically up to an energy-loss of 50-100 eV. The plasmon peak is caused by a collective oscillation of the valence/conduction electrons in the specimen. Studying these so called plasmon excitations allow for the derivation of the dielectric response function of the material, which provides us with information about the electronic and optical behaviour of the material.

The optical properties of the specimen can be expressed in terms of the absorption coefficient, transmission coefficient and reflection coefficient. These coefficients decide if incident waves of a certain energy will be absorbed, transmitted or reflected by the specimen. It is of great interest to study the effect of sample size, type of material and orientation on these coefficients. All three coefficients are directly linked to one specific parameter: the dielectric response function. The dielectric response function provides information about the response of a material to an electric field. This dielectric response can be determined by studying the plasmon excitations in the energy-loss spectrum. EELS can therefore provide information about the optical behaviour of the specimen.

In this thesis, an attempt is made to obtain the dielectric response function from the energy-loss spectrum of different specimen. The obtained dielectric response function can then be used to derive the optical properties of the specimen. In the theory, the basics to the excitation of plasmons is discussed. In the experimental method, the functionality of the script is explained, as well as the studied data. In the results, the dielectric functions and optical properties of the studied data have been exploited.

2

THEORY

Electron energy-loss spectroscopy (EELS) is used to study the atomic structure of a material. In EELS, electrons pass through a thin specimen, at energies ranging between 100 and 300 keV. Since electrons are charged particles, they may interact through Coulomb forces with charged particles inside the specimen. The electrons in the valence/conduction band of the sample are relatively weakly bound to an atom, and can therefore be displaced relatively easy by an incident electron. The collective excitation of the valence electrons is called a plasmon oscillation. Studying these plasmon excitations allow for the derivation of the dielectric response function of the material, which provides us with information about the electronic and optical behaviour of the material.

2.1. ELECTRON ENERGY-LOSS SPECTROSCOPY IN THE SCANNING TRANSMISSION ELECTRON MICROSCOPE

In transmission electron microscopy (TEM), electrons are excited at energies ranging from several to hundreds of electronvolts and are transmitted through a specimen to study its atomic structure. Due to the wave-nature of electrons, a much better spatial resolution can be achieved than in X-Ray techniques. In scanning transmission electron microscopy (STEM), the incident electrons are focused using electromagnetic lenses into a small probe before they pass through a specimen. This probe can have a diameter as small as 0.1 nm, which is about as large as an atom. Spatial resolution is one of the main strengths of STEM, when compared to other microscope techniques. X-Ray microscopy, for example, can only reach a spatial resolution of several tenths of nanometers [4]. The energy resolution of STEM is mainly determined by the type of electron gun that is used. When using a field-emission tip, a resolution of 0.5-1 eV can be achieved. However, when a monochromator is used, the energy resolution can be improved to 100 meV.

In EELS, electrons are transmitted at energies typically between 100 and 300 keV [5]. The electrons will pass completely through the sample provided its thickness is below 1 μm. After the electrons have passed through the specimen, they are collected by a spectrometer. The spectrometer collects only the electrons that have scattered under very small angles, typically several milliradians, and uses electric and magnetic fields to sort the electrons based on their energy, which gives the energy-loss spectrum. The energy-loss spectrum provides the possibility to study different excitations the incident electron might cause within the specimen. A typical setup for EELS in a STEM is displayed in figure 2.1.

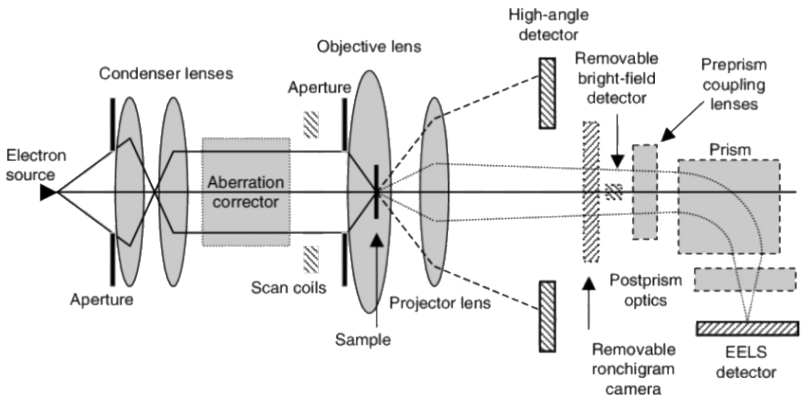

Figure 2.1: A typical setup for EELS in a STEM.

2.2. THE ENERGY-LOSS SPECTRUM

There are several possibilities for an incident electron to interact with the specimen. These interactions can be divided into three subgroups. First, the electrons that haven't lost any energy are considered. These electrons may have scattered elastically with a nucleus of the specimen, or may have not scattered at all. These electrons appear as a sharp peak around 0 eV in the energy-loss spectrum. This peak is commonly known as the zero-loss peak (ZLP). Typically, the ZLP is several electronvolts wide, which makes it difficult to observe energy losses below 5 eV. These electrons, which may for example have excited phonon modes, are covered by the ZLP and thus not visible in the energy-loss spectrum [6].

The second group contains the electrons up to an energy-loss of 100 eV, and is known as the low-loss region. Most of these electrons have excited oscillations of the valence/conduction electrons of the specimen. These oscillations are known as plasmon excitations, and cause a peak at integer multiples of the plasmon energy, which is typically between 15 and 25 eV. The third group contains electrons which have interacted with the stronger bound electrons of the specimen, and have lost a significantly larger amount of energy. This region is therefore known as the core-loss region, and extends from 100 eV up to several keV. A typical low-loss spectrum and ZLP are displayed in figure 2.2, and an example of a core-loss region is displayed in figure 2.3. For the purpose of this thesis, only the low-loss region and ZLP of the energy-loss spectrum are of interest. Interactions in the core-loss region will therefore not be addressed.

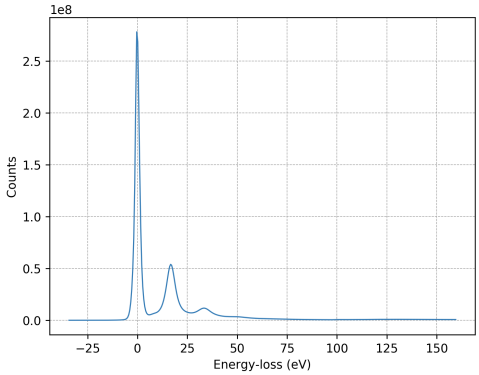

Figure 2.2: Example of the low-loss region of a typical energy-loss spectrum. This example is taken from a Si wafer.

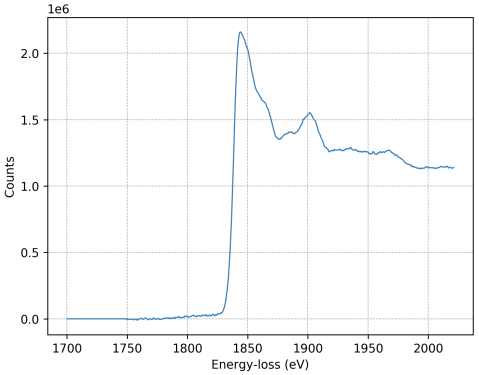

Figure 2.3: Example of the core-loss region of a typical energy-loss spectrum. This example is taken from a Si wafer.

2.3. BULK PLASMON EXCITATIONS

The valence electrons of an atom are not very strongly bound to the nucleus of the atom. Therefore, when a fast moving electron passes through a solid, the nearby valence electrons are displaced by coulomb forces, leaving a hole of positive potential behind the electron [5]. Provided the electron speed exceeds the Fermi

velocity, the incident electron will excite charge oscillations in the region behind the electron. This effect is known as a plasmon wake. In figure 2.4, such a plasmon wake is visualised. These charge oscillations are excited inside the specimen, and are hence referred to as volume or bulk plasmons.

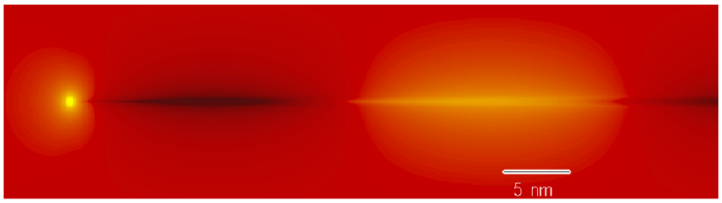

Figure 2.4: A visualisation of a plasmon wake of a 100 keV electron travelling through aluminium. The dark region represents positive charge, and the light region represent negative charge. The bright dot on the left is the incident electron, which propagates to the left. [5]

The frequency of the charge oscillation is characteristic for a material, and is given by:

$$2\pi f_p = \omega_p = \sqrt{\frac{ne^2}{\epsilon_0 m}} \tag{2.1}$$

Where e is the elementary charge, ϵ_0 the permittivity in vacuum m_0 is the mass of the electron, and n is the electron density, given by:

$$n = \frac{z\rho}{uA} \tag{2.2}$$

Where z is the number of valence electrons per atom, u the atomic mass unit, A the atomic weight and ρ the density of the solid. The energy-loss associated with the excitation of one of these oscillations is called the plasmon energy: $E_p = \hbar\omega_p$. A typical value of E_p lies somewhere between 15 and 25 eV. In figure 2.2, the plasmon energy can be observed to be around 19 eV. The incident electron may excite an oscillation at any integer multiple of the plasmon frequency, which explains the peak at 38 eV in the low-loss spectrum.

2.3.1. DIELECTRIC FORMULATION

The dielectric response function ϵ of a material provides information on the response of a material to an electric field. The dielectric response function is a complex function, for which we will use the notation:

$$\epsilon = \epsilon_1 + i\epsilon_2 \tag{2.3}$$

When an electron passes through a material, the displacement D is directly linked to the dielectric response function ϵ [7]:

$$D(\mathbf{q}, E) = \epsilon(\mathbf{q}, E)\epsilon_0 E(\mathbf{q}, E) \tag{2.4}$$

Here, ϵ depends on the wavevector $\mathbf{q}$ and the energy-loss E, and ϵ_0 is the permittivity in vacuum. The scattering distribution of the electron caused by the excitation of bulk plasmons is given by a double differential cross section σ, which gives the energy and angle dependence of the scattering. The derivation of the double differential cross section is not very easy and straight-forward, so only the result is presented [8]:

$$\frac{d^2\sigma}{d\Omega dE} \approx \frac{\mathrm{Im}\left[\frac{-1}{\epsilon(q,E)}\right]}{\pi^2 a_0 m_0 v^2 n_a}\left(\frac{1}{\theta^2 + \theta_E^2}\right) \tag{2.5}$$

Here, $\theta_E = E/(\gamma m_0 v^2)$ is a characteristic scattering angle, and n_a is the number of atoms per unit volume. From equation 2.5, we observe that $\mathrm{Im}[-1/\epsilon]$ provides a complete description of the response of the medium, and is therefore named the energy-loss function (ELF). When equation 2.5 is integrated over the unit solid angle up to the collection angle of the spectrometer β, and using $d\Omega = \sin\theta d\theta d\phi \approx \theta d\theta d\phi$, we obtain:

$$\frac{d\sigma(\beta)}{dE} \approx \left(\frac{1}{\pi a_0 m_0 v^2 n_a}\right)\mathrm{Im}\left[\frac{-1}{\epsilon(E)}\right]\ln\left(1 + \frac{\beta^2}{\theta_E^2}\right) \tag{2.6}$$

When the zero-loss peak is removed from the low-loss region of the energy loss spectrum, we may extract the energy-loss function from the low-loss region of the spectrum using equation 2.6.

2.3.2. RADIATION LOSSES

When the electron exceeds the speed of light of the medium it passes through, it emits Čerenkov radiation. This condition is satisfied when [4]:

$$\epsilon_1(E) > c^2/v^2 \tag{2.7}$$

The differential probability of bulk scattering including radiation is given by [9]:

$$\frac{d^2P_{\text{b,ret}}}{d\Omega dE} = \frac{1}{\pi^2 a_0 m_0 v^2}\text{Im}\left[\frac{t\mu^2}{\epsilon^*\phi^2}\right] \tag{2.8}$$

In which ϵ^* marks the complex conjugate of ϵ, and

$$\mu^2 = 1-\epsilon^*(v/c)^2 \tag{2.9}$$

$$\phi^2 = \theta^2+\theta_E^2-\epsilon^*\theta_E^2(v/c)^2 \tag{2.10}$$

After converting the differential probability from equation 2.8 to a differential cross section and setting $v/c \to 0$, equation 2.5 for bulk losses is obtained.

2.4. SURFACE PLASMON EXCITATIONS

Apart from the excitation of charge oscillations inside the specimen, the incident electron may excite a charge oscillation on either of the surfaces of the specimen. The typical energy-loss that comes with a surface plasmon excitation is characteristic for a material and for a sample, and is referred to as the surface plasmon energy E_s. E_p and E_s typically exhibit the relation $E_s = E_p/\sqrt{2}$ [10]. Due to the continuity of the electric field on the surface of the specimen, we require [4]:

$$\epsilon+\eta = 0 \tag{2.11}$$

In which ϵ and η are the permittivities on either side of the boundary. The differential probability of surface scattering at a single interface is given by [11]:

$$\frac{d^2P_s}{d\Omega dE} = \frac{k_0^2|q_s|}{\pi^2 a_0 m_0 v^2 q^4\cos\theta_i}\text{Im}\left[\frac{-4}{\epsilon+\eta}+\frac{1}{\epsilon}+\frac{1}{\eta}\right] \tag{2.12}$$

In which $q^2 = k_0^2(\theta^2+\theta_E^2)$ and $q_s = k_0\theta\cos\theta_i + k_0\theta_E\sin\theta_i$, with θ_i the angle of the incident electron to the normal of the surface. For normal incidence, the differential probability can be integrated up to the collection angle β of the spectrometer:

$$\frac{dP_s}{dE} = \frac{1}{\pi a_0 k_0 m_0 v^2}\left[\frac{\tan^{-1}(\beta/\theta_E)}{\theta_E}-\frac{\beta}{(\beta^2+\theta_E^2)}\right]\text{Im}\left[\frac{-4}{\epsilon+\eta}+\frac{1}{\epsilon}+\frac{1}{\eta}\right] \tag{2.13}$$

2.4.1. BEGRENZUNGS EFFECT

When the incident electron is close to an interface, it rather excites surface plasmons than bulk plasmons. This effect is known as the Begrenzungs Effect gives rise to a negative value of the surface-loss [4]. This negative value is represented by the $\text{Im}(1/\epsilon)$ and $\text{Im}(1/\eta)$ terms in equation 2.12. This reduction in bulk plasmon excitations can range up to 40% for thin specimens [12].

2.5. THIN SURFACE PLASMONS

When the specimen is thick, the surface excitations are independent of each other. However, when a very thin specimen is considered, the surface excitations are no longer independent of one another. The surface excitations will know be excited simultaneously, and will appear in a symmetric or anti-symmetric mode. The surface plasmon energy is therefore split into two modes, given by [8]:

$$\omega_s = \omega_p\sqrt{\frac{1\pm\exp(-q_s t)}{1+\epsilon}} \tag{2.14}$$

Assuming normal incidence, the differential probability of surface excitation of a specimen with thickness t is given by [12]:

$$\frac{d^2P_{\text{s,thin}}}{d\Omega dE} = \frac{1}{\pi^2 a_0 k_0 m_0 v^2} \frac{\theta}{(\theta^2+\theta_E^2)^2} \text{Im}\left[\frac{(\eta-\epsilon)^2}{\eta^2\epsilon} R_c\right] \tag{2.15}$$

With R_c given by:

$$R_c = \frac{\eta \sin^2(tE/2\hbar v)}{\epsilon+\eta\tanh(q_s t/2)} + \frac{\eta\cos^2(tE/2\hbar v)}{\epsilon+\eta\coth(q_s t/2)} \tag{2.16}$$

Integrating equation 2.15 to a scattering angle β can only be done by integrating over the unit solid angle Ω numerically, since q_s also depends on θ.

2.6. SINGLE SCATTERING DISTRIBUTION

Electrons can scatter multiple times in the specimen. Since the cross section and scattering probabilities described in sections 2.3, 2.4 and 2.5 only describe the single scattering of the incident electron, it is necessary to obtain the single scattering distribution (SSD) $J^1(E)$ from the measured energy loss spectrum $J(E)$. Neglecting energy broadening by the spectrometer and the effect of coupled surface plasmons, the SSD is given by [4] [13]:

$$J^1(E) = I_0 n_a t \frac{d\sigma}{dE} + I_0 \frac{dP_s}{dE} \tag{2.17}$$

If ineleastic scattering events are assumed to be independent, the total scattering distribution can be described by Poisson statistics. The probability of an electron undergoing n scattering events is given by:

$$P_n = I_n/I_t = (1/n!)(t/\lambda)^n \exp(-t/\lambda) \tag{2.18}$$

With λ the mean free path, and I_t the total intensity of the energy-loss spectrum. This probability provides a method to extract the mean free path from the spectrum, provided the thickness of the specimen is known. Filling in $n=0$ in equation 2.18 gives:

$$t/\lambda = \ln(I_t/I_0) \tag{2.19}$$

Where I_0 is the intensity of the ZLP. In theory, I_t would be the intensity of the incident electrons. However, due to elastic scattering, not all incident electrons are captured by the spectrometer, which makes obtaining λ from the energy-loss spectrum not very accurate. It is therefore necessary to obtain t and λ prior to the calculation of the SSD. Then, the intensity of the single scattering distribution can be obtained through:

$$I_1 = I_0 t/\lambda \tag{2.20}$$

For the derivation of λ, the Malis formula is used [14]:

$$\lambda = 106\text{nm}\frac{FE_0}{E_m \ln(2\beta E_0/E_m)} \tag{2.21}$$

With $F = (1+E_0/1022)/(1+E_0/511)^2$, $E_m = 7.6Z^{0.36}$, and Z the atomic number, and with β in mrad and E_0 in keV.

2.7. THE DIELECTRIC FUNCTION

Without much introduction, the dielectric function was introduced as the key factor to the excitation of bulk and surface plasmons. The dielectric function is a complex function, for which the notation $\epsilon = \epsilon_1 + i\epsilon_2$ is used. For the description of the dielectric function, the Lorentz model is used, which states ϵ can be expressed as [15] [16] [17]:

$$\epsilon(E) = 1 + \frac{E_p^2}{E_{\text{eff}}^2 - E^2 - i\Gamma E} \tag{2.22}$$

Here, E_p is the plasmon energy, E_{eff} represents an effective inter-band transition energy [18], and Γ denotes the damping of the mode. From ϵ, the refractive index n, absorption coefficient α and reflection coefficient R can be determined [15]:

$$n = \sqrt{\frac{\sqrt{\epsilon_1^2 + \epsilon_2^2} + \epsilon_1}{2}} \tag{2.23}$$

$$\alpha = \sqrt{\frac{\sqrt{\epsilon_1^2 + \epsilon_2^2} - \epsilon_1}{2}} \tag{2.24}$$

$$R = \frac{(n-1)^2 + k^2}{(n+1)^2 + k^2} \tag{2.25}$$

The Lorentz model for the dielectric function as in equation 2.22 has been plotted in figure 2.5, for $E_p = 24$ eV, $E_{\text{eff}} = 6$ eV and $\Gamma = 3$ eV. In figure 2.6 the bulk and surface losses according to equations 2.6 and 2.13 have been plotted for the dielectric function in figure 2.5 and a thickness $t = 30$ nm.

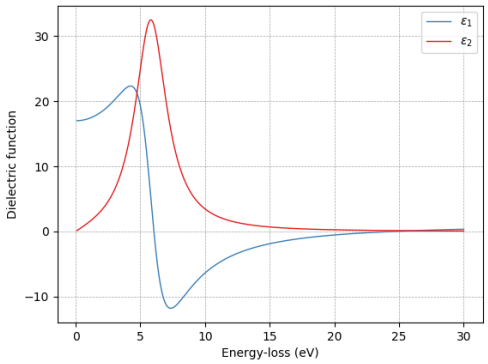

Figure 2.5: Lorentz model for the dielectric function using $E_p = 24$ eV, $E_{\text{eff}} = 6$ eV and $\Gamma = 3$ eV.

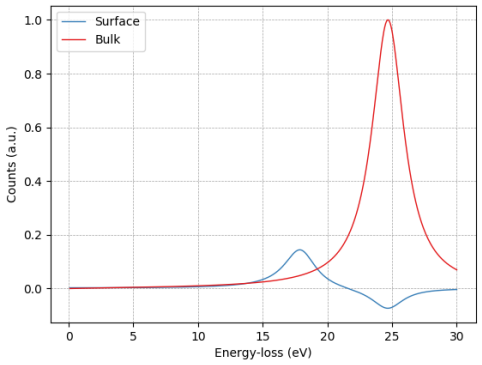

Figure 2.6: Bulk and surface losses according to the dielectric function in figure 2.5.

The surface contribution is independent of the specimen thickness and the bulk contribution scales linearly with t, which implies the bulk contribution will dominate the energy-loss spectrum at large t, as one might expect. The negative counts in the surface contribution represent the Begrenzungs effect.

From equation 2.22, several observations can be made from the plot of the dielectric function which immediately provide information about the constants E_p, E_{eff} and Γ. The function ϵ_1 and ϵ_2 should intersect twice. In the limit $\Gamma \to 0$ they intersect at $E = E_{\text{eff}}$ and $E = \sqrt{E_p^2 + E_{\text{eff}}^2}$. For $\Gamma > 0$, the first intersection shifts to a lower energy, and the second intersection shifts to a higher energy. Studying the limit $E \to 0$ shows that $\epsilon_1(E=0) = 1 + E_p^2/E_{\text{eff}}^2$ and $\epsilon_2(E=0) = 0$. The peak of ϵ_2 appears at $E = E_{\text{eff}}$ for $\Gamma = 0$, and shifts down for increasing Γ. The peaks of ϵ_1 also appear at $E = E_{\text{eff}}$ for $\Gamma = 0$. Increasing Γ shifts the upper peak to a lower energy, and the lower peak to a higher energy. For high damping, both peaks in ϵ_1 shift so far that they disappear from the energy-loss spectrum.

In figure 2.7 the bulk and surface contribution to the dielectric function in figure 2.5 have been plotted, with the only difference that the thickness has been reduced to 1 nm. As expected, the surface contribution increases relative to the bulk contribution. In figure 2.8, the bulk contribution has been modified to take retardation effects into account, according to equation 2.8. The surface contribution has been modified to equation 2.15 the surface contribution for thin specimens. The retardation effects are visible in the form of Čerenkov radiation in the bulk contribution close to $E = 0$. Switching to thin surface contribution changes the surface contribution quite radically: it reduces the counts and spreads the contribution towards $E = 0$. Moreover, the two different modes, as described in chapter 2.5 become apparent.

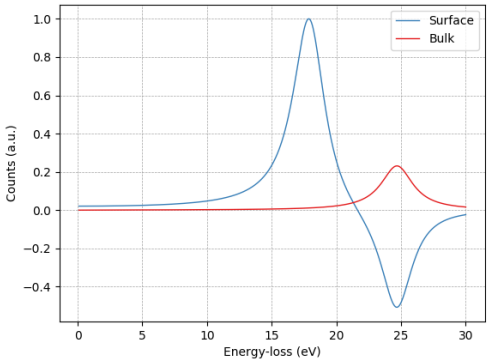

Figure 2.7: Bulk and surface losses according to the dielectric function in figure 2.5, with specimen thickness 1 nm.

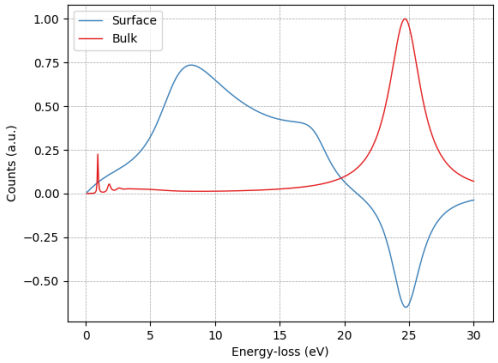

Figure 2.8: Bulk and surface losses with the same requirements as in figure 2.7, with the bulk contribution including retardation losses, and the surface contribution modified to the thin surface contribution.

2.8. KRAMERS KRONIG ANALYSIS

When the energy-loss function $\mathrm{Im}[-1/\epsilon]$ is determined from the energy-loss spectrum, a Kramers Kronig Analysis (KKA) can be applied to determine the dielectric response function of the material. The KKA calculates the real part of a function from its imaginary part [19]:

$$\mathrm{Re}\left[\frac{1}{\epsilon(E)}\right] = 1 - \frac{2}{\pi} P \int_0^\infty \mathrm{Im}\left[\frac{-1}{\epsilon(E')}\right] \frac{E' \, dE'}{E'^2 - E^2} \tag{2.26}$$

In which P marks the Cauchy principal value of the integral, due to the discontinuity at $E' = E$. To get an exact solution, we use Fourier transforms to prevent the need of integrating over the singularity. Apart from preventing the discontinuity at $E' = E$ in equation 2.26, the FFT method is much faster than computing a direct integral. The steps necessary to obtain $\mathrm{Re}[1/\epsilon]$ are:

1. Extrapolate the energy-loss spectrum, so the data decays smoothly to zero, and the amount of data points is a power of two. Adding enough data points will avert the wrap-around effect of the FFT.
2. Take the FFT of the energy-loss spectrum.
3. Multiply the first half of the FFT with $2/\pi$, and the second half with $-2/\pi$.
4. Swap the real and imaginary part of the FFT, and set the imaginary part to 0.
5. Take the inverse FFT, multiply with 2π and add 1 to obtain $\mathrm{Re}[1/\epsilon]$.

ϵ can then be obtained by using straight-forward algebra:

$$\epsilon = \epsilon_1 + i\epsilon_2 = \frac{\mathrm{Re}\,[1/\epsilon] + i\,\mathrm{Im}\,[-1/\epsilon]}{(\mathrm{Re}\,[1/\epsilon])^2 + (\mathrm{Im}\,[-1/\epsilon])^2} \tag{2.27}$$

3

EXPERIMENTAL METHOD

The main purpose of this thesis is to develop a python script which obtains the dielectric response function from an energy-loss spectrum. As discussed in the theory, the energy-loss spectrum mainly consists of bulk plasmon contributions and surface plasmon contributions. When the bulk contribution is known, the energy-loss function can be computed using equation 2.5, from which the dielectric response can be computed using the Kramers-Kronig analysis. To obtain the bulk contribution from an energy-loss spectrum which also contains surface contribution, we first assume the entire spectrum to be bulk contribution. Then, the dielectric function is determined following the steps stated. An approximation of the surface contribution is then made by computing the surface loss using equation 2.12. The estimated surface loss is subtracted from the energy-loss spectrum. The spectrum is then again assumed to consist only of bulk contribution, and the process is repeated untill it converges. The complete script has been added in the appendix.

3.1. THE SCRIPT

Several straight-forward functions have been written to retrieve data from csv files, to plot the data and compute the dielectric function. Furthermore, a module named *c* has been created which contains several important physical constants. For example, *c.c* returns the speed of light and $c.a_0$ returns the Bohr radius. All constants are in SI units. A structure named *experiment* has been defined which contains all the required information of the EELS experiment. For example, *experiment.phi_out* returns the collection angle of the spectrometer, and *experiment.beam_energy* returns the energy of the incident electrons. All variables are in SI units, except for the beam energy, which is in keV.

Several key functions are discussed. After retrieving the data, it needs to be extrapolated with many data points as discussed in chapter 2.8, and it needs to be extrapolated to $E = 0$. Both extrapolations should be made with the same energy resolution as the original data. The extrapolation to $E = 0$ must be done first, so in the extrapolation at the end of the data set can ensure the amount of data points is a power of two, as required by the FFT. It is important to note that the data should not be extrapolated to exactly $E = 0$, but to a value close to $E = 0$ to prevent singularities in the calculations. Several methods can be considered when extrapolating the data. In both extensions, the data should go rapidly, but smoothly to zero. The easiest way to do this is to use a function in the form of ae^{bx} to extrapolate the data to zero. A value for b can either be fitted from the data, or chosen to be a certain fixed value. The value of a can then be chosen such that the extrapolated data does not show a very large discontinuity.

After the extrapolations, the data should be converted to the SSD. The intensity of the SSD can be calculated using equation 2.20. Computing the SSD is a straight-forward calculation after the mean free path has been calculated using equation 2.21.

```
def ssd(EELS,experiment,I0,t):
  Z = 10
  Em = 7.6*Z**0.36
  E0 = experiment.beam_energy
  F = (1+E0/1022)/np.square(1+E0/511)
  L = 106e-9*F*E0/(Em*np.log(2*beta*1e3*E0/Em))

  xdata = EELS[:,0]*c.e
  for i in range(1,np.size(EELS,1)):
```

```
    In = integrate.simps(EELS[:,i],xdata)
    I1 = I0[i-1]*t[i-1]/L
    EELS[:,i] = EELS[:,i]*I1/In

  return EELS
```

Listing 3.1: SSD function

The input *EELS* is a (NxM+1) numpy matrix, with in the first column the energy-losses in eV, and in the other M columns M different datasets. I_0 and t should be numpy arrays of size M, containing the intensities of the zero-loss peak of the different spectra, and the thicknesses of the specimen. Note that when multiple spectra are feeded to the script at the same time, they should all have the same amount of data points.

After the SSD has been obtained, the ELF can be calculated by dividing the SSD with the constant factor and angular dependence function in equation 2.6, taking into account the factor tI_0n_a added by the derivation of the SSD in equation 2.17.

```
def ELF(EELS,experiment,I0,t):
  ELF = np.copy(EELS)
  theta_e = EELS[:,0]*c.e/(experiment.gamma*experiment.m*experiment.v**2)
  angular_dependence = np.log(1+np.square(np.divide(experiment.phi_out,theta_e)))

  factor = np.multiply(I0,t)/(np.pi*c.a0*c.me*experiment.v**2)

  ELF[:,1:] = np.divide(ELF[:,1:],np.ones((EELS.shape[0],1),dtype=float)*np.asmatrix(
    factor))
  ELF[:,1:] = np.divide(ELF[:,1:],np.transpose(np.asmatrix(angular_dependence))*np.ones
    ((1,EELS.shape[1]-1),dtype=float))

  return ELF
```

Listing 3.2: ELF function

Again, the input *EELS* is a (NxM+1) numpy matrix. In lines 8 and 9, the determined factor and angular dependence are converted to matrices, so the entire *EELS* spectrum can be corrected of these factors in one calculation. The ELF can be feeded to the KKA to determine ϵ. A script has been written for the direct integral method, and for the FFT method. A class has been made named *kka* which contains these two functions.

```
class kka:

  @staticmethod
  def integrate_eels(EELS):
    N = np.size(EELS,0)
    r = EELS[1,0]-EELS[0,0]

    B = np.zeros((N,2),dtype=float)

    for i in range(0,N):
      EELS_integrate = np.delete(EELS,i,0)
      x = EELS_integrate[:,0]
      y = EELS_integrate[:,1]
      y = np.multiply(y,np.divide(x,np.square(x) - np.square(EELS[i,0])))

      Bi = 1 - (2/np.pi)*integrate.simps(y,x)
      B[i,:] = np.array([EELS[i,0],Bi])

    epsilon = np.divide(np.complex(0,1)*EELS[:,1] + B[:,1],np.square(EELS[:,1]) + np.
    square(B[:,1]))

    return epsilon

  @staticmethod
  def fft(EELS):
    nn = np.size(EELS,0)

    if(not nn%2 == 0):
      struct.ColorPrint.print_fail("Error in kka.fft: Amount of data points N must be
    even.")

    h = int(nn/2)
    q = fftpack.fft(EELS[:,1:],n=nn,axis=0)
```

```
32
33      q[0:h,:] = (2/np.pi) * np.imag(q[0:h,:])
34      q[h:,:] = (-2/np.pi) * np.imag(q[0:h,:])
35      q=np.real(q)
36
37      p = fftpack.ifft(q,n=nn,axis=0)*2*np.pi
38      p = 1+np.real(p)
39
40      epsilon = np.divide(np.complex(0,1)*EELS[:,1:] + p,np.square(EELS[:,1:]) + np.
        square(p))
41
42      return epsilon
```

Listing 3.3: KKA class

Since the variable name *integrate* was already taken by the module *scipy.integrate*, the script of the direct integration has been named *integrate_eels*. The *integrate_eels* function takes an (Nx2) matrix as input, meaning it can only handle one data set at the time. The function is not very fast, since it needs to compute a numerical integral for each data point. Therefore, feeding the extrapolated data to this function is quite ambiguous and will result in a memory error, and moreover not necessary since a discrete integration does not have a wrap-around effect like the FFT. The numerical integration is performed using Simpson's rule [20]. To avert the discontinuity, the data point at $E^{'} = E$ is simply removed from the data set, which happens in line 11. The *FFT* function takes a (NxM+1) matrix as input, and determines ϵ according to the steps in chapter 2.8.

After ϵ has been determined, the surface loss may be estimated using equation 2.13. The inputs are a (NxM+1) matrix *EELS*, a (NxM) matrix *epsilon*, a (NxM) matrix *eta*, the experiment and an array I_0 with M elements. The function assumes normal incidence, and returns the surface correction in an (NxM) matrix.

```
1  def surface_correction(EELS,epsilon,eta,experiment,I0):
2    T = 0.5*c.me*experiment.v**2
3    factor = np.asmatrix(I0/(np.pi*c.a0*experiment.k*T))
4    theta_e = EELS[:,0]*c.e/(experiment.gamma*experiment.m*np.square(experiment.v))
5
6    S1 = np.transpose(np.asmatrix(np.divide(np.arctan(np.divide(experiment.phi_out,
       theta_e)),theta_e) - np.divide(experiment.phi_out,(experiment.phi_out**2 + np.
       square(theta_e)))))
7    S2 = np.imag(np.divide(-4,epsilon+eta)+1/epsilon+1/eta)
8
9    S1 = S1*np.ones((1,np.size(EELS,1)-1),dtype=int)
10   factor = np.ones((np.size(EELS,0),1),dtype=int)*factor
11
12   SS = np.multiply(S1,S2)
13   SS = np.multiply(factor,SS)
14
15   return SS
```

Listing 3.4: Surface function

When working with energy-loss spectra of thin specimens, it is necessary to separate the surface contribution from the energy-loss spectrum before the dielectric function can be computed. First, the assumption is made that the energy-loss spectrum only consists of bulk contribution. Then, the dielectric response function can be obtained by using the functions stated above. The surface contribution can be approximated by plugging the determined dielectric response into the function $surface\_correction$. The approximated surface contribution is subtracted from the energy-loss spectrum, after which the entire process repeats itself. In the end, the dielectric function ϵ is obtained so that:

$$SSD(E) = S_b(\epsilon(E)) + S_s(\epsilon(E)) \tag{3.1}$$

In which S_b and S_s are the bulk and surface contribution to the SSD. The process is visualised in figure 3.1, in which the gray outline represents the iteration. The iteration is considered converged when the variable *Modified SSD* does not change between iterations.

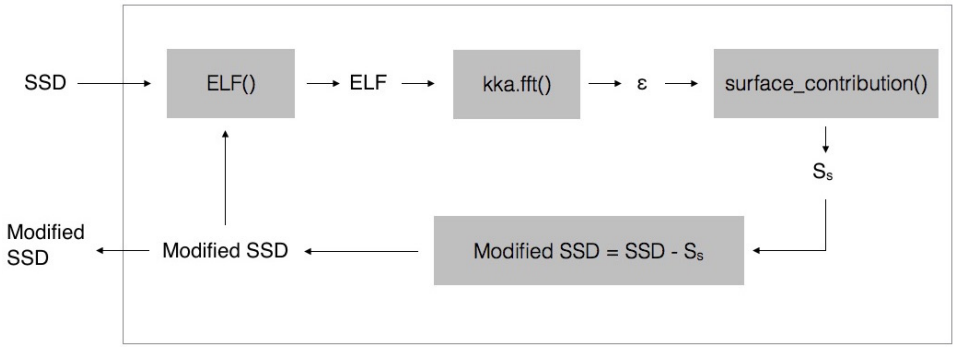

Figure 3.1: A visualisation of the iteration that removes the surface contribution from the SSD.

For thick specimen ($t > 50$nm), it is sufficient to determine the dielectric response function as described above. However, for thin specimen, a correction needs to be made for retardation effects. This is done by a script very similar to that in figure 3.1, with the only difference that the *surface_contribution* function is replaced by a function that determines the retardation contribution as in 2.8.

3.2. DATA

The script is first tested on a MoS_2 energy-loss function $\mathrm{Im}[-1/\epsilon]$, which is obtained from the program WIEN2k. To verify the script, the obtained dielectric function is compared to the result obtained from the program WIEN2k. When referring to *bulk* samples, the assumption is made that the sample is sufficiently thick to neglect surface contributions in the energy-loss spectrum. A comparison will be made between the integration and FFT methods for the Kramers-Kronig analysis.

After the script has been tested on the bulk MoS_2, three energy-loss spectra of a MoS_2 nanowall are analyzed. A nanowall can be visualised as a thin slab, as in figure 3.2. The thickness of the nanowall is 89 nm, which means surface contributions will become apparent in the energy-loss spectrum. The script will be applied to this spectrum to check its ability to remove the surface contribution from the spectrum. The nanowall is assumed to be surrounded by vacuum, meaning $\eta = 0$.

The lattice structure of MoS_2 consists of hexagonal layers put on top of each other, as shown in figure 3.3. In the bulk spectrum of MoS_2, the specimen is oriented in such a way that the incident electrons pass perpendicular through the layers. However, in the nanowall, the crystal is rotated under 90 degrees with respect to the incident electrons, meaning the electrons pass parallel to the layers through the crystal. This difference in orientation might give rise to differences in the energy-loss spectra of the bulk MoS_2 and the nanowall [21].

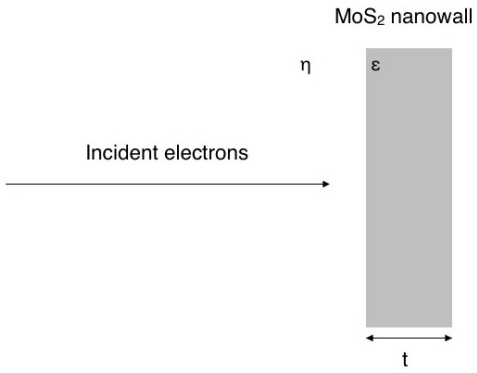

Figure 3.2: A visualisation of the MoS_2 nanowall. ϵ marks the dielectric function of the MoS_2, and η marks the dielectric function of the surroundings. The specimen is assumed to be in vacuum, meaning $\eta = 0$.

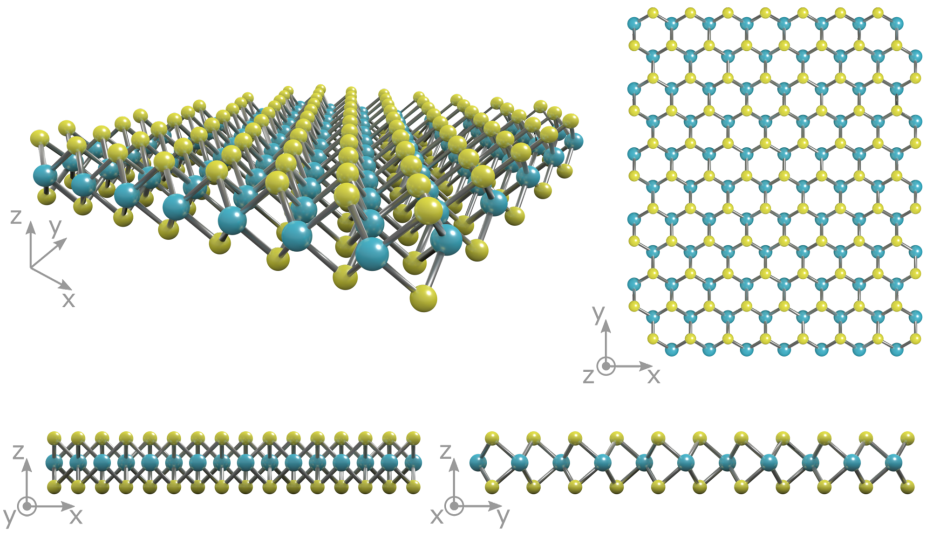

Figure 3.3: A visualisation of a single layer of MoS_2

To test whether the script can separate the surface contribution from the bulk contribution in thin samples, the energy-loss spectra from an InAs nanowire is used. This nanowire is covered by a layer of aluminum, and a layer of aluminum-oxide. Three energy-loss spectra are obtained from the nanowire:

1. The first spectrum is taken through the edge of the nanowire, so only through the Al_2O_3. However, since this layer is very thin, excitations in the aluminum may occur in the energy-loss spectrum.
2. The second spectrum is taken through the Al and the Al_2O_3. The spectrum should mainly contain excitations from the Al, since the layer of Al the electrons pass through is much larger than the layer of Al_2O_3.
3. The third spectrum is taken more or less through the center of the nanowire. Excitations in the Al_2O_3 should barely be visible, since this layer is very thin compared to the Al and InAs.

A visualisation of the acquisition of these spectra is shown in figure 3.4.

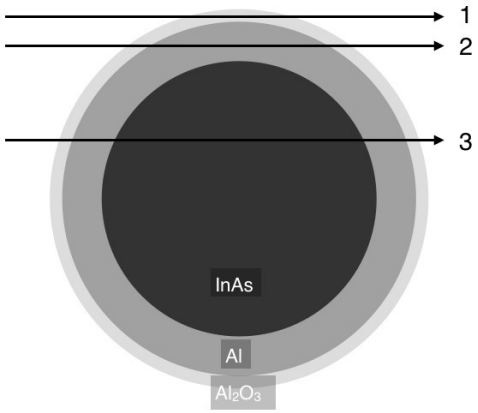

Figure 3.4: A visualisation of the three different spectra acquired from the nanowire.

Before the energy-loss spectra can be analyzed, it is important to find the literary values of the bulk and surface plasmon peaks. This will create a better understanding of the energy-loss spectra, and will allow for a prediction on how the separation of the bulk and surface contributions should look like.

MoS_2 has a bulk plasmon energy of 23 eV [22]. Apart from the bulk plasmon energy, MoS_2 energy-loss spectra tend to have a peak around 8 eV, which is due to the excitation of π-electron [22]. Al_2O_3 has a bulk plasmon energy between 23 and 26 eV, depending on its exact structure [23] [24]. The bulk plasmon energy of

Aluminum is around 15 eV [25], and typically has a double plasmon excitation around 30 eV [26]. The value of the surface plasmon energy reported in the literature is 10 eV [27]. The bulk plasmon energy of InAs is around 14 eV [28].

4

Results and Discussion

4.1. WIEN2k simulated MoS_2 ELF

The dielectric response function has been determined from the energy-loss function of bulk MoS_2. The energy-loss function has been plotted in figure 4.1. The dielectric response function has been determined using the FFT method from the Kramers-Kronig analysis, as described in chapter 2.8. The results are compared to the dielectric response function calculated by HypersPY. A comparison between the results is shown in figure 4.2. The two function overlap for the most part, except for some small deviations close to $E = 0$. A satisfactory explanation for this deviation has not been found.

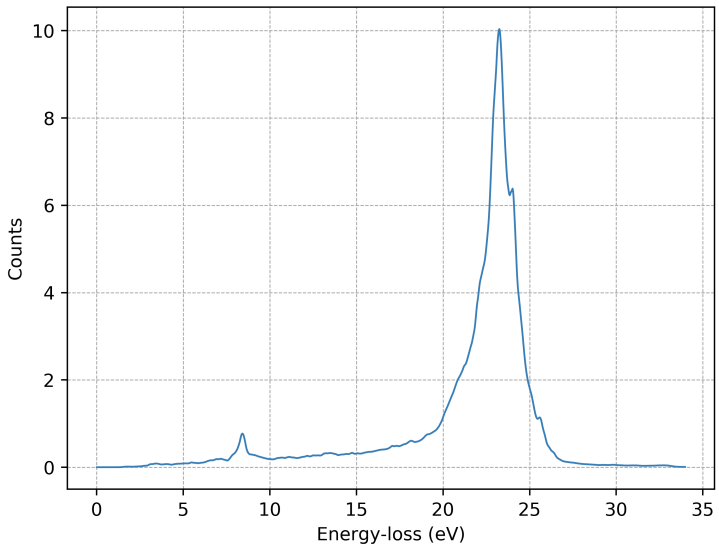

Figure 4.1: The energy-loss function of bulk MoS_2.

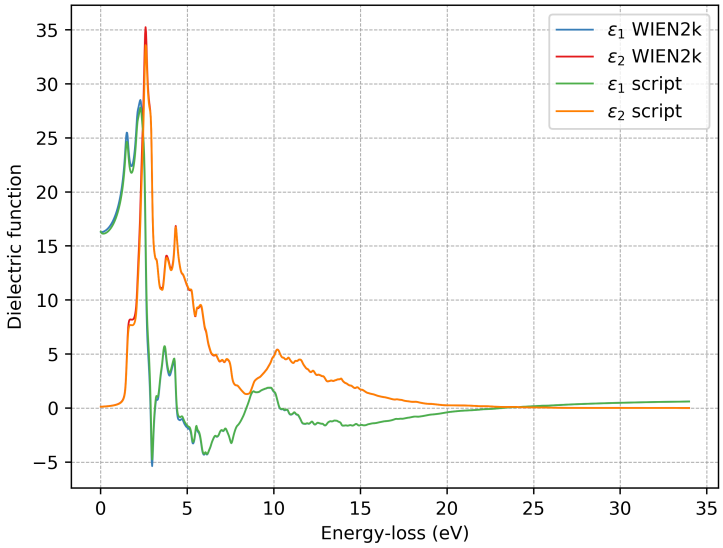

Figure 4.2: The real and imaginary part of the dielectric response function computed by HypersPY versus the dielectric response function computed using the FFT method described in 2.8.

A straight-forward way to check if the FFT analysis provides the correct result, is to compare it to the dielectric response function obtained from the Kramers-Kronig analysis when the direct integral is used. The singularity is removed by simply removing the point from the data for which $E^{'} = E$ in equation 2.26. The result is shown in figure 4.3. The two computed dielectric response functions match very well. This also suggests that the deviation in figure 4.2 is not caused by the Kramers-Kronig analysis, but due to some correction that has been made on the data.

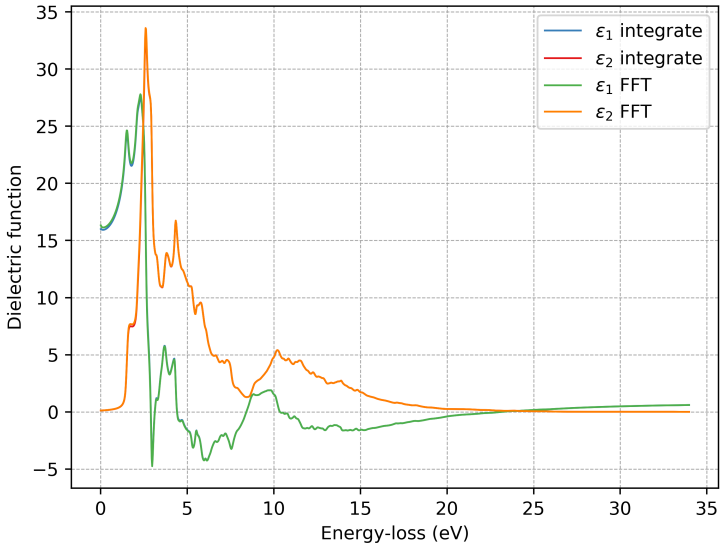

Figure 4.3: Comparison of the Kramers-Kronig analysis when using a direct integration (blue and red) and the FFT method (orange and green). The plots overlap for the majority of the graph.

The refractive index n and absorption coefficient α for the bulk MoS_2 have been determined, and are plotted in figures 4.4 and 4.5 respectively.

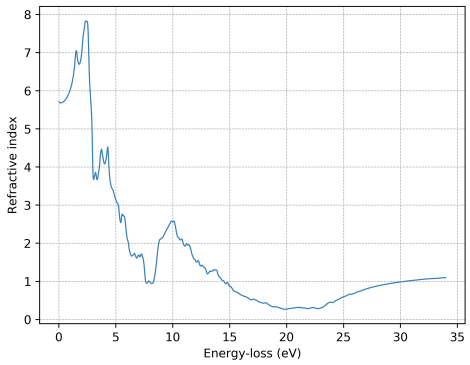

Figure 4.4: Refractive index of the WIEN2k simulated MoS_2.

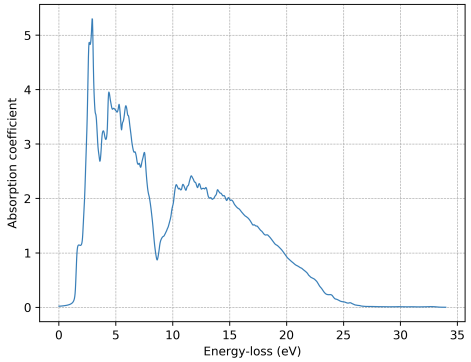

Figure 4.5: Absorption coefficient of the WIEN2k simulated MoS_2.

4.2. BULK MoS_2

The experimental data from a bulk MoS_2 specimen is shown in figure 4.6. The thickness of this specimen is 85 nm. The SSD is shown in figure 4.7. The data is extended by fitting a exponential to through the last 20 data points of the SSD.

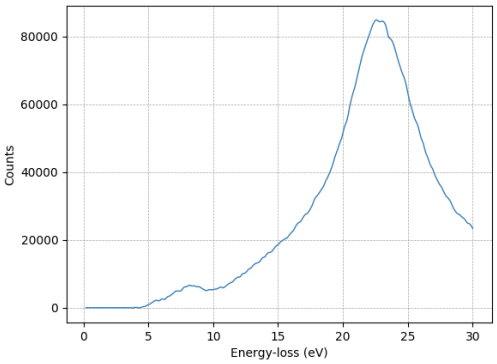

Figure 4.6: Energy-loss spectrum of the bulk MoS_2.

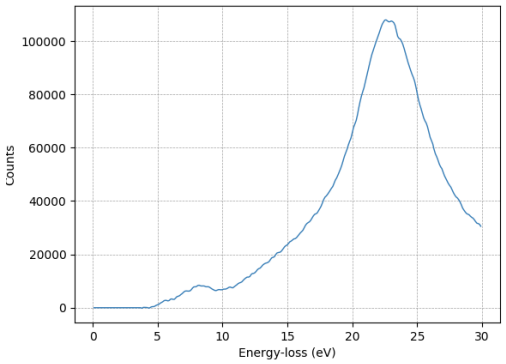

Figure 4.7: SSD of the bulk MoS_2.

The extracted bulk and surface contributions are plotted in figure 4.8 and 4.9. As expected for a bulk specimen, the determined surface contribution is very small compared to the bulk contribution. The surface contributions consists of two peaks, whereas only one peak should be expected. The explanation for this, is that since the material is very thick and surface plasmons only account for a very small contribution to the energy-loss spectrum, some of the bulk contribution on the rising edge of the bulk plasmon peak might trigger a second peak in the surface contribution.

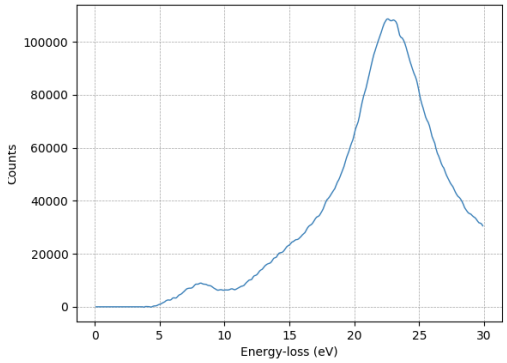

Figure 4.8: Bulk contribution to the SSD of the bulk MoS_2.

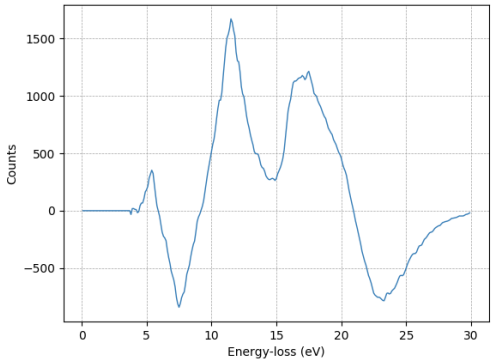

Figure 4.9: Surface contribution to the SSD of the bulk MoS_2.

The dielectric function is plotted in figure 4.10. The refractive index, absorption coefficient and reflection coefficient have been plotted in figures 4.11, 4.12 and 4.13 respectively.

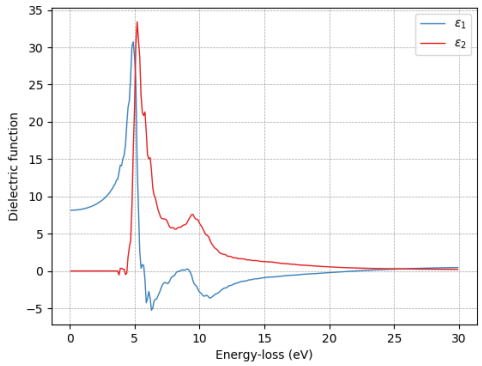

Figure 4.10: Dielectric response function of the bulk MoS_2.

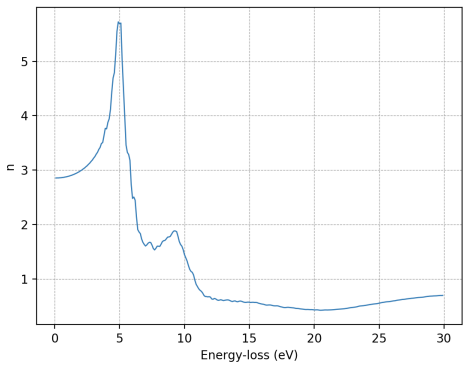

Figure 4.11: Refractive index of the bulk MoS_2.

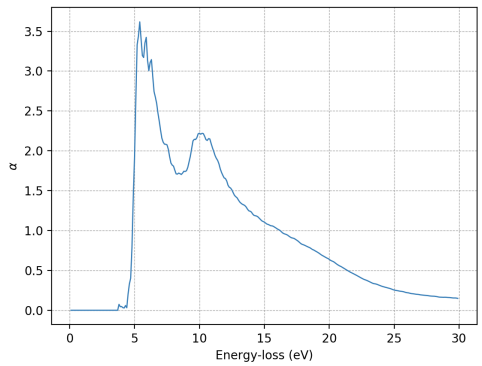

Figure 4.12: Absorption coefficient of the bulk MoS_2.

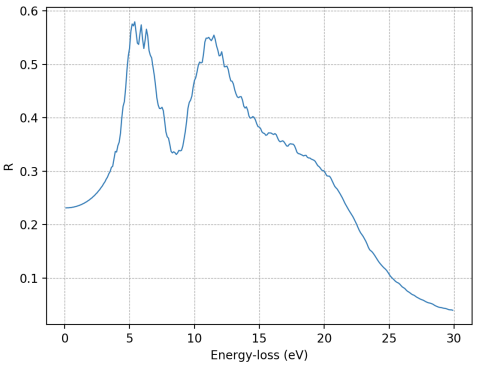

Figure 4.13: Reflection coefficient of the bulk MoS_2.

The measured value of E_p is 22.6 eV. The intersections of ϵ_1 and ϵ_2 are found at 5.1 and 25.5 eV. The value of $\epsilon_1(E=0) = 8.1$ eV. The determine value of E_{eff} is then calculated as 8.5 eV.

4.3. MoS_2 NANOWALL

Three energy-loss spectra have been studied from an approximately 89 nm thick MoS_2 nanowall. The spectra have been acquired from the same nanowall, which has a uniform thickness. The prediction is therefore that all three spectra should look very much alike. The SSD of the three spectra have been plotted in figure 4.14. The data sets have been acquired from 5 eV to 30 eV. The data decays exponentially to zero after 30 eV.

Even though the nanowall has more or less the same thickness as the bulk MoS_2, some differences can be spotted in their energy-loss spectrum. The nanowall seems to have an extra contribution between 10 and 15 eV compared to the bulk MoS_2. These differences are probably due to the change in rotation of the lattice with respect to the incident electrons as discussed in chapter 3.2. In the bulk MoS_2 case, the electrons propagate perpendicular to the layers, whereas in the nanowall, the electrons propagate parallel to the layers.

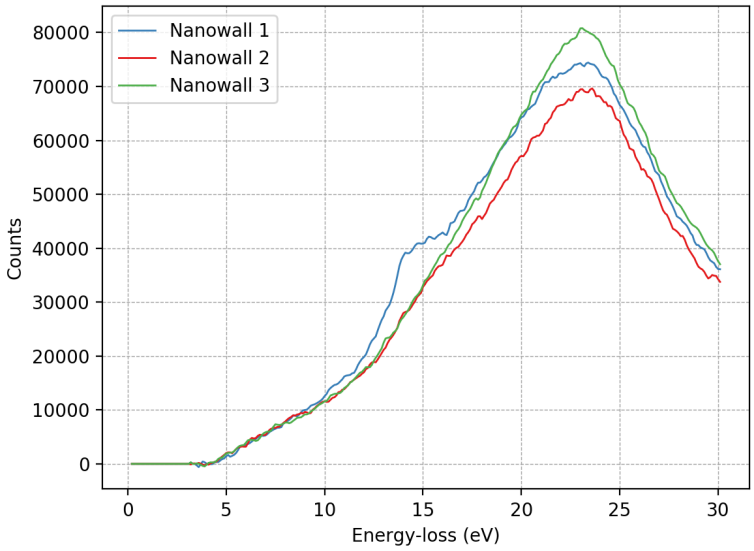

Figure 4.14: SSD distribution of the three nanowalls.

The surface contribution to the energy-loss spectra can be observed in the bobble on the rising edge of the bulk plasmon peak, around 15 eV. The bulk and surface contributions to the three spectra have been separated, and plotted in figures 4.15 and 4.16. The real and imaginary parts of the dielectric function have been plotted in figures 4.17 and 4.18.

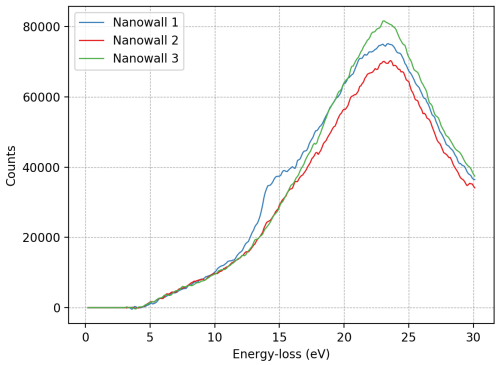

Figure 4.15: Bulk contributions to the energy-loss spectra of the nanowall.

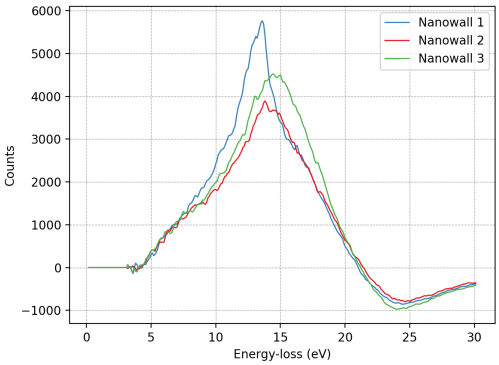

Figure 4.16: Surface contributions to the energy-loss spectra of the nanowall.

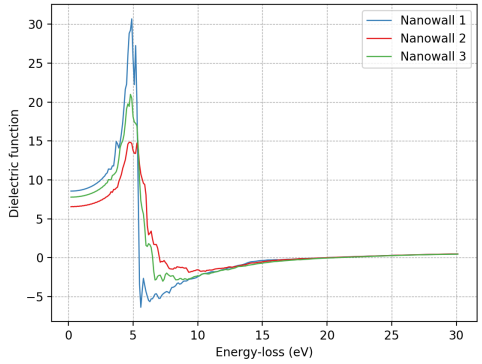

Figure 4.17: Real part of the dielectric functions of the spectra of the nanowall.

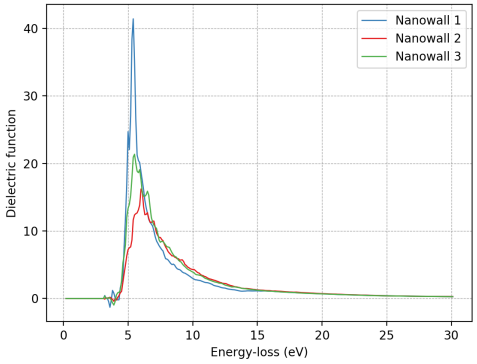

Figure 4.18: Imaginary part of the dielectric functions of the spectra of the nanowall.

All three bulk contributions in figure 4.15 seem to contain additional peaks between 5 and 15 eV. The spread-out peak towards $E = 5$ eV may be due to the excitation of π-electrons. Since MoS_2 consists of monolayers, suggestions have been made that there is an additional peak at $E = 15eV$, which might be due to surface modes between the individual layers of the MoS_2 [22].

Even though the dielectric functions from the bulk MoS_2 and the nanowall match pretty well to a certain extend, they do not match the theoretical values as imaged by the data from WIEN2k. Other literary values of the dielectric function of bulk MoS_2 also showed that $\epsilon_1(E = 0) \approx 16$, whereas the determined dielectric functions all have values between 5 and 10 eV for $\epsilon_0(E)$ [29].

One of the factors that can have an influence on the result is the mean free path. The mean free path is determined using equation 2.21, but can deviate from this value with tenths of nanometers [30]. However, the separation of the bulk and surface contributions did not seem very sensitive to change in the mean free path. However, when the mean free path is chosen too low, the surface contribution is overestimated by the script, and the script does not converge. When the mean free path is chosen too high, the surface contribution is underestimated, and the script converges rapdily without identifying much surface contribution. The dielectric function proved to be very sensitive to change in the thickness and mean free path. The real and imaginary part of the dielectric function seemed to be multiplied by some factor as these parameters changed, while maintaining their shape.

The measured bulk plasmon energy E_p and surface plasmon energy E_s in the three spectra are displayed in table 4.1. As discussed in chapter 2.4, the ratio E_p^2/E_s^2 is typically close to 2. The determined values of the bulk plasmon energy match the 23 eV predicted by the literature reasonably well [22].

Spectrum	E_p **(eV)**	E_s **(eV)**	E_p^2/E_s^2
1	23.4	13.6	3.0
2	23.6	13.8	2.9
3	23.1	14.4	2.6

Table 4.1: Values of E_p and E_s as determined from figures 4.15 and 4.16.

Several of the properties discussed in chapter 2.7 have been determined for the dielectric functions, and are plotted in table 4.2. The values of E_{eff} have been determined by using that $\epsilon_1(E = 0) = 1 + E_p^2/E_{\text{eff}}^2$, as shown in section 2.7.

Spectrum	**Intersects (eV)**	**Max** ϵ_2 **(eV)**	$\epsilon_1(E = 0)$	E_{eff} **(eV)**
1	5.2, 26.5	5.4	8.6	9.6
2	5.5, 26.5	6.0	10.3	6.5
3	5.3, 26.5	5.5	9.7	7.8

Table 4.2: Intersections and maxima of the dielectric function of the three spectra of the nanowalls.

The refractive index n and absorption coefficient α have been determined for the three spectra. All three spectra exhibit the same shape, and only differ a small amount in intensity.

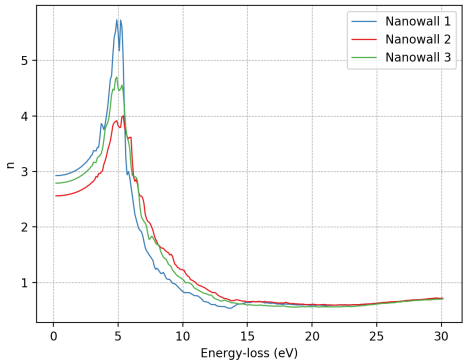

Figure 4.19: Refractive index n determined from the dielectric responses of the nanowall.

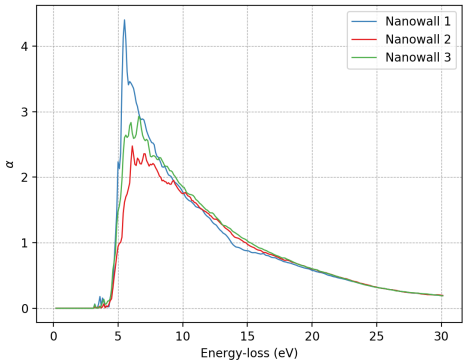

Figure 4.20: Absorption coefficient α determined from the dielectric responses of the nanowall.

As discussed in chapter 2.3.2, retardation effects come into play when $\epsilon_1(E) > c^2/v^2$. The velocity of the incident electrons in this experiment was $c/v = 1.29$, which means retardation effects become significant when $\epsilon_1 > 1.67$. This is definitely the case in the region close to $E = 0$ in figure 4.17. Unfortunately, this data set has only been acquired for energies above 5 eV, so no comparison can be made. More over, the fine structure near $E = 0$ eV is mostly covered by the zero-loss peak, and therefore difficult to measure.

The surface contribution has also been calculated taking into account the thin surface contribution. However, the nanowall turned out to be sufficiently thick to discard this contribution, as the result was exactly the same as in figure 4.16.

Typically, these energy-loss spectra require about 10-20 iterations before they fully converge. Since the Fourier method for the Kramers Kronig analysis is a very fast algorithm compared to the direct integration, the entire script runs within seconds, and the main contribution to the runtime is the plotting and saving of the figures.

4.4. NANOWIRE

4.4.1. Al_2O_3

In figure 4.21, the energy-loss spectrum of the Al_2O_3 section of the nanowire is shown. In figure 4.22, the SSD is plotted according to this spectrum. The energy-loss spectrum has been acquired between 5 and 40 eV. In figures 4.21 and 4.22, the data has already been extended to $E = 0.1$ eV.

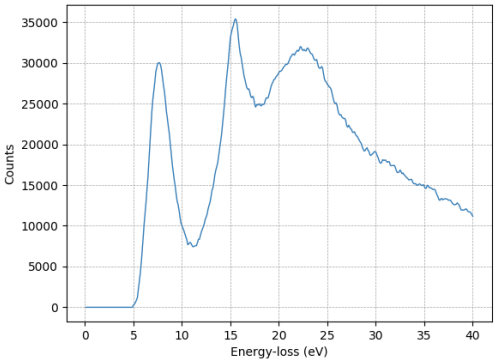

Figure 4.21: Energy-loss spectrum of the Al_2O_3 section of the nanowire.

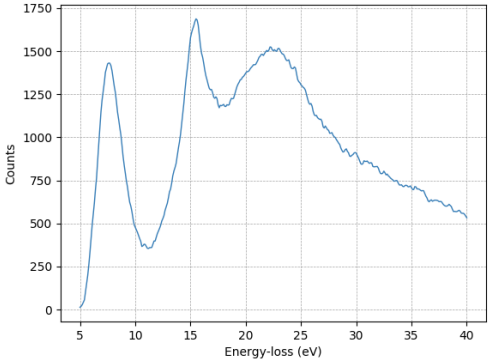

Figure 4.22: SSD corresponding to the energy-loss spectrum of the Al_2O_3 section of the nanowire.

Before an attempt can be made to obtain the dielectric response function from this SSD, it is important to

first make a prediction of what the bulk and surface contributions to this SSD are. Since the layer of Al_2O_3 is very thin, this SSD might contain contributions from aluminum bulk and surface plasmon excitations. When examining the different peaks in the energy-loss spectrum of Al_2O_3, the surface plasmon of Aluminum can be observed at 7.5 eV, which is slightly lower than reported in the literature [27]. Then, a peak is observed at 15 eV, which is due to the Aluminum bulk plasmon [25]. The last peak is observed at 23 eV, which is the bulk plasmon peak of Al_2O_3.

The separated bulk and surface contributions are plotted in figures 4.23 and 4.24

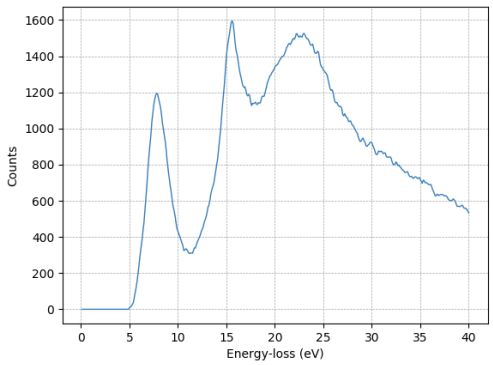

Figure 4.23: Bulk contribution to the energy-loss spectrum of the Al_2O_3 section of the nanowire.

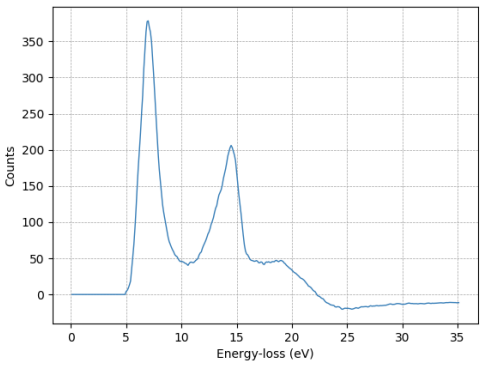

Figure 4.24: Bulk contribution to the energy-loss spectrum of the Al_2O_3 section of the nanowire.

Only a small fraction of the suspected surface contribution is separated from the bulk contribution. The energy-loss spectrum might contain too many different contributions to effectively separate bulk and surface contributions. Apart from bulk and surface contributions, further literary research suggested more band transitions contributing to the energy-loss spectrum in the 10-16 eV range [31]. Moreover, this specific cross-section of the nanowire is only tenths of nanometers thick. The energy-loss spectrum therefore consists for the most part of surface contributions. The initialisation of the script, which assumes the entire spectrum is bulk contribution, is too bad of an initialisation to effectively separate the contributions. The last factor that might cause the script to not separate the surface contribution entirely, is the assumption of normal incidence. Since the nanowire is typically cylindrically shaped, this assumption might not be very appropriate, especially near the edge of the nanowire.

4.4.2. ALUMINUM

The energy-loss spectrum and SSD of the Aluminum spectrum are displayed in figures 4.25 and 4.26.

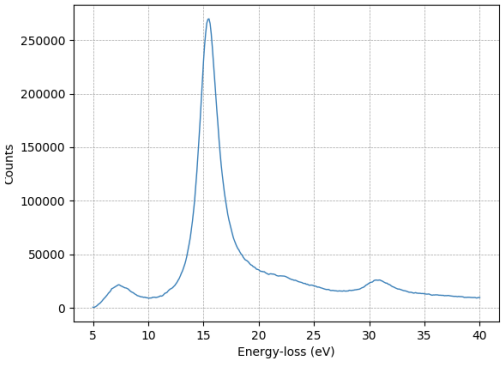

Figure 4.25: Energy-loss spectrum of the Al section of the nanowire.

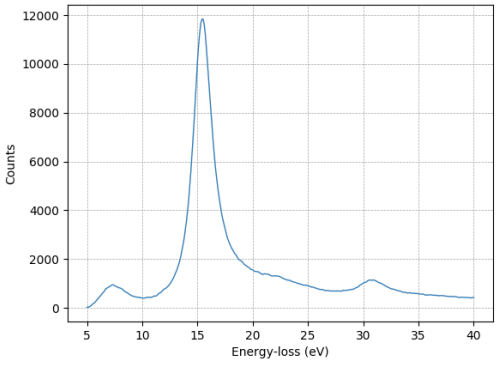

Figure 4.26: SSD corresponding to the energy-loss spectrum of the Al section of the nanowire.

The peak at 7.5 eV is again the surface plasmon from Aluminum, and the peaks at 16 and 32 eV are the

single and double bulk plasmon excitations of Aluminum. Just as in the spectrum of Al_2O_3, the surface contribution is not separated completely from the spectrum, as can be concluded from figure2 4.27 and 4.28.

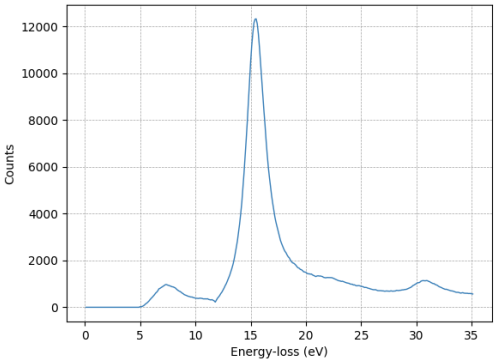

Figure 4.27: Bulk contribution to the energy-loss spectrum of the Al section of the nanowire.

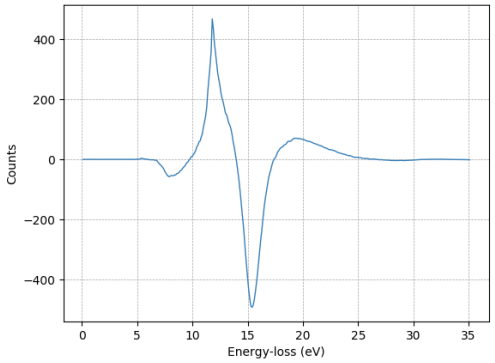

Figure 4.28: Surface contribution to the energy-loss spectrum of the Al section of the nanowire.

The script finds a surface contribution on the rising edge of the bulk plasmon peak of the aluminum, but, as can be concluded from the negative counts, sees the surface contribution at 7.5 eV as bulk contribution. This again might have to do with the initialisation of the script, in which the entire energy-loss is assumed to be bulk contribution.

The dielectric response function and refractive index have been determined for the aluminum, and are plotted in figures 4.29 and 4.30.

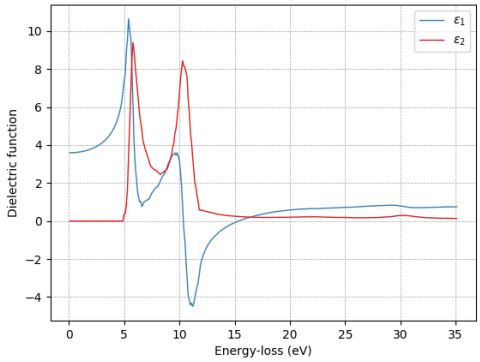

Figure 4.29: Dielectric function determined from the energy-loss spectrum of the Al section of the nanowire.

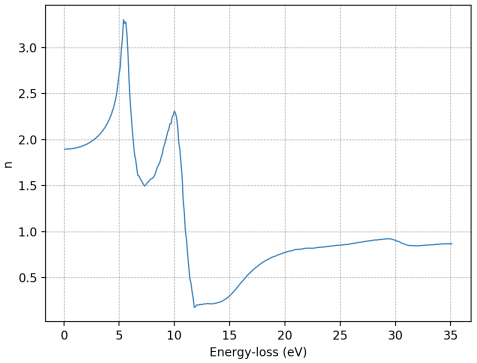

Figure 4.30: Refractive index determined from the dielectric function.

The dielectric function looks like the dielectric function derived from the Lorentz model in section 2.7, with an extra peak in ϵ_1 and ϵ_2 around 5 eV. This is most likely due to the ineffective removal of the surface contribution to the energy-loss spectrum. A qualitative analysis of the dielectric function, as done with the nanowall, is therefore not accurate.

4.4.3. INAS

The energy-loss spectrum and the SSD of InAs is plotted in figures 4.31 and 4.32.

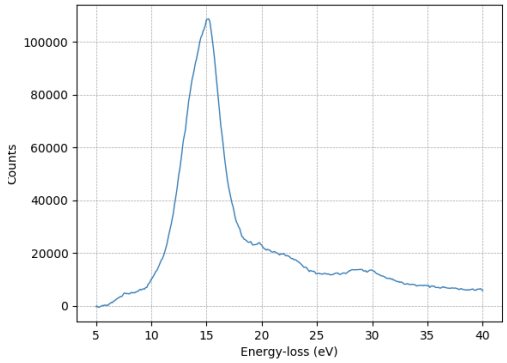

Figure 4.31: Energy-loss spectrum of the InAs section of the nanowire.

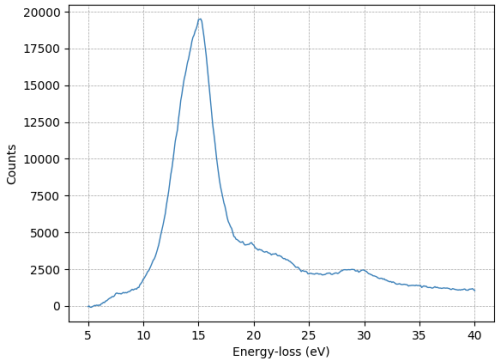

Figure 4.32: SSD corresponding to the energy-loss spectrum of the InAs section of the nanowire.

The bulk plasmon of InAs is clearly visible around 15 eV, which is slightly higher than the literary suggests [28]. Around 5 eV, a small contribution from the Aluminum surface plasmon can be identified. The bulk and surface contributions are plotted in figures 4.33 and 4.34.

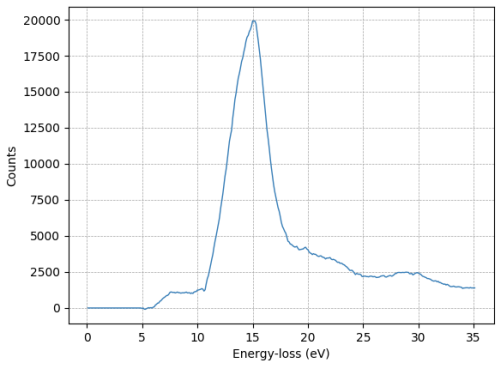

Figure 4.33: Bulk contribution to the energy-loss spectrum of InAs.

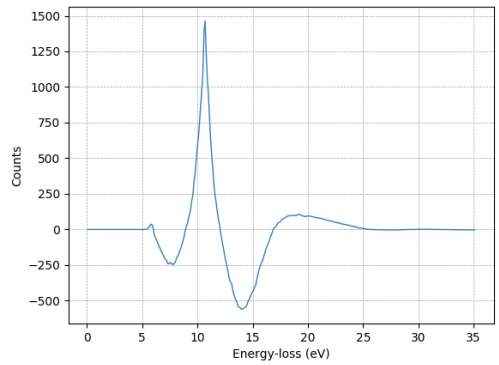

Figure 4.34: Surface contribution to the energy-loss spectrum of InAs.

The largest peak in the surface contribution is the surface contribution of the InAs. Again, the script seems to identify the surface peak of Aluminum at 7.5 eV as a bulk peak, as it produces negative counts in the surface contribution. At 14 eV, the Begrenzungs effect of the bulk InAs is visible. The dielectric function, refractive index, absorption coefficient and reflection coefficient have been plotted in figures 4.35, 4.36, 4.37 and 4.38 respectively.

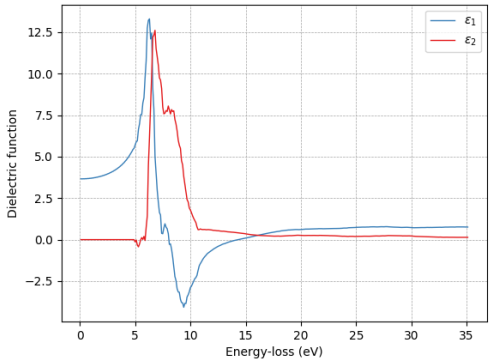

Figure 4.35: Dielectric function calculated from the energy-loss spectrum of InAs.

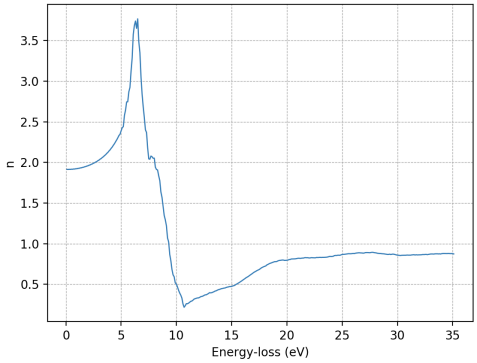

Figure 4.36: Refractive index calculated from the dielectric function.

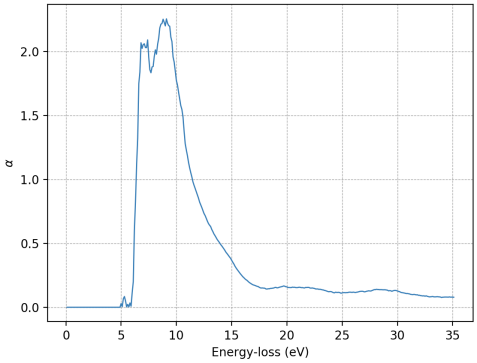

Figure 4.37: Absorption coefficient calculated from the dielectric function.

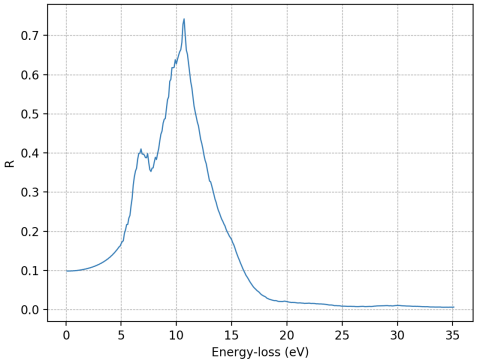

Figure 4.38: Reflection coefficient calculated from the dielectric function.

The measured plasmon energy in figure 4.33 is $E_p = 15.2$ eV. The measured surface energy is $E_s = 10.7$ eV. The square ratio between the bulk plasmon energy and surface plasmon energy is $E_p^2/E_s^2 = 2$, which matches the prediction made in section 2.4. The intersections of ϵ_1 and ϵ_2 occur at 6.5 and 16.1 eV, and $\epsilon_1(E = 0) = 3.67$. The calculated value for E_{eff} is then 9.3 eV.

The script did not manage to remove the surface contributions from the energy-loss spectra of the nanowire very effectively. This is probably due to the initialisation of the script, and the fact that other contributions are present in the energy-loss spectra. It is therefore recommended to deal with other (non-plasmonic) contributions to the energy-loss spectrum in an appropriate way. These contributions could, for example, be estimated and removed from the spectrum before the dielectric function is calculated. Moreover, a different initialisation for the script needs to be found and implemented for a more reliable calculation of the dielectric function for surface dominated energy-loss spectra.

For a better determination of the dielectric response function and optical properties of the specimen, it is recommended to use a different method to obtain the SSD. The SSD is in the script determined by a simple calculation of the intensity of the SSD. However, the choice of the mean free path has a very large influence on the outcome of the script. For example, a Fourier log deconvolution of the energy-loss spectrum could be used to the determine the SSD, from which a more accurate estimate of the mean free path can be made [32].

5

CONCLUSION

The developed script provides a fast and straight-forward way to determine the dielectric response function of bulk samples in EELS. The script determines the single-scattering distribution (SSD) through Poisson statistics and an estimate of the mean free path. The script then uses a Fourier method for the Kramers Kronig analysis to compute the dielectric function, which can be used to determine the optical properties of the sample.

When surface losses are present in the energy-loss spectrum of a sample, the script assumes the entire energy-loss spectrum to be bulk contribution, for which the dielectric response function is determined. The surface loss is then estimated and subtracted from the spectrum. This process repeats itself after it converges, typically after about 10-20 iterations.

The energy-loss spectrum of very thin specimens is typically dominated by surface plasmon contributions. The script fails in identifying the surface contributions in such spectra, which is most likely due to the initialisation of the script, which assumes the entire spectrum to be bulk contribution. When the script converges for thin specimen, it most likely labels the true surface contribution as bulk contribution.

Bulk and surface plasmon excitations are not the only contributions that might occur in the low-loss region of an energy-loss spectrum. For example, band transitions might occur in the energy-loss spectrum. Such contributions must be dealt with appropriately before the script can be used.

When no other contributions to the energy-loss spectrum are present, and the spectrum is not dominated by surface losses, the script provides a reliable and fast way of determining the dielectric response function and optical properties of the sample. The script has proven to be very sensitive to the input value of the mean free path. Even though the mean free path can be estimated fairly easily, experience with running the script for different spectra has proven that this might not always provide a good approximation, and that the mean free path might deviate slightly from the approximated value.

The script may be improved by dealing with all contributions to the energy-loss spectrum in an appropriate manner. Moreover, a better estimate of the mean free path should be made, since the outcome of the script is very sensitive to this parameter. For energy-loss spectra dominated by surface contributions, a better initialisation of the script needs to be implemented in order for the script to converge.

BIBLIOGRAPHY

[1] A. Dashora, U. Ahuja, and K. Venugopalan, *Electronic and optical properties of mos2 (0 0 0 1) thin films: Feasibility for solar cells,* Computational Materials Science **69**, 216 (2013).

[2] D. Davelou, G. Kopidakisn, G. Kioseoglou, and I. Remediakis, *Mos2 nanostructures: Semiconductors with metallic edges,* Solid State Communications **192**, 42 (2014).

[3] M. Ye, D. Winslow, D. Zhang, R. Pandey, and Y. Khin Yap, *Recent advancement on the optical properties of two-dimensional molybdenum disulfide (mos2) thin films,* Photonics , 288 (2015).

[4] R. Egerton, *Electron energy-loss spectroscopy in the electron microscope (3rd ed.)* (Springer, New York, 2011).

[5] R. Egerton, *Electron energy-loss spectroscopy in the TEM,* Reports on Progress in Physics (2008), doi:10.1088/0034-4885/72/1/016502.

[6] R. Egerton, *New techniques in electron energy-loss spectroscopy and energy-filtered imaging,* Micron (2003), doi:10.1016/s0968-4328(03)00023-4.

[7] H. Kuzmany, *Solid-State Spectroscopy*, 2nd ed. (Springer).

[8] R. H. Ritchie, *Plasma losses by fast electrons in thin films,* Phys. Rev. **106**, 874 (1957).

[9] R. Erni and N. Browning, *The impact of surface and retardation losses on valence electron energy-loss spectroscopy,* Ultramicroscopy **108**, 84 (2008).

[10] M. Rocca, *Low-energy eels investigation of surface electronic excitations on metals,* Surface Science Reports **22**, 1 (1995).

[11] H. Raether, *Excitation of plasmons and interband transitions by electrons,* Springer Tracts in Modern Physics **88** (1980).

[12] H. Raether, *Surface plasma oscillations as a tool for surface examinations,* Surface Science **8**, 233 (1967).

[13] D. Johnson, *Pre-spectrometer optics in a ctem/stem,* Ultramicroscopy **5**, 163 (1980).

[14] T. Malis, S. C. Cheng, and R. F. Egerton, *Eels log-ratio technique for specimen-thickness measurement in the tem,* Microscopy Research & Technique **8**, 193 (1988).

[15] X. Yan, L. Zhu, Y. Zhou, Y. E, L. Wang, and X. Xu, *Dielectric property of mos2 crystal in terahertz and visible regions,* Applied Optics **54**, 6732.

[16] V. Keast, *An introduction to the calculation of valence eels: Quantum mechanical methods for bulk solids,* Micron **44**, 93 (2013).

[17] F. Wooten, *Optical Properties of Solids* (Academic Press, INC., New York, 1972).

[18] A. Eljarrat Ascunce, *Quantitative methods for electron energy loss spectroscopy,* (2015).

[19] P. Burzzoni, R. Carranza, J. Collet Lacoste, and E. Crespo, *Kramers-kronig transforms calculation with a fast convolution algorithm,* Electrochimica Acta **48**, 341 (2002).

[20] C. Vuik, F. Vermolen, M. van Gijzen, and M. Vuik, *Numerical Methods for Ordinary Differential Equations,* 2nd ed. (Delft Academic Press, 2016).

[21] M. Tinoco, L. Maduro, and S. Conesa-Boj, *Metallic edge states in zig-zag vertically-oriented mos2 nanowalls,* (2019), arXiv:1906:03437.

[22] M. Disko, M. Treacy, S. Rice, R. Chianelli, J. Gland, T. Halbert, and A. Ruppert, *Spatially resolved electron energy-loss spectroscopy of mos2 platelets,* Ultramicroscopy **23**, 313 (1987).

[23] W. Tews and R. Gründler, *Electron-energy-loss spectroscopy of different al2o3 modifications,* Physica Status Solidi (b) **109** (1982), doi:10.1002/pssb.2221090128.

[24] R. B. Pettit, J. Silcox, and V. R., *Measurement of surface-plasmon dispersion in oxidized aluminum films,* Physical Review Letters **11** (1975), doi:10.1103/PhysRevB.11.3116.

[25] M. McClain, A. Schlather, E. Ringe, *et al.*, *Aluminum nanocrystals,* Nanoletters (2015), doi:10.1021/acs.nanolett.5b00614.

[26] P. Schattschneider, F. Fodermayr, and D.-S. Su, *Coherent double-plasmon excitation in aluminum,* Physical Review Letters **59**, 724 (1987).

[27] A. Bagchi and C. Duke, *Determination of the surface-plasmon dispersion relation in aluminum by inelastic electron diffraction,* Physcial Review B **5** (1972), doi:10.1103/PhysRevB.5.2784.

[28] M. Kundmann, *Study of semiconductor valence plasmon line shapes via electron energy-loss spectroscopy in the transmission electron microscope,* (1988), doi:10.2172/6340092.

[29] L. Cao *et al.*, *Exciton-dominated dielectric function of atomically thin mos2 films,* Scientific Reports (2015), doi:10.1038/srep16996.

[30] H. Zhang, R. Egerton, and M. Malac, *Eels investigation of the formulas for inelastic mean free path,* Microsc. Microanal. **17** (2011), doi:10.1017/S1431927611008208.

[31] E. Arakawa and M. Williams, *Optical properties of aluminum oxide in the vacuum ultraviolet,* Journal of Physics and Chemistry of Solids **29**, 735 (1968).

[32] F. Wang, R. Egerton, and M. Malac, *Fourier-ratio deconvolution techniques for electron energy-loss spectroscopy (eels),* Ultramicroscopy **109**, 1245 (2009).

A

DATA.PY

The *data.py* file consists of two important things. First, it contains a class *retrieve*, which contains the functions which retrieve all the data from the csv files. Secondly, the file contains the functions necessary to extrapolate the data, and to calculate the SSD.

```
import numpy as np
import csv
import constants as c
import functions as func
import structure as struct
from scipy import optimize
from scipy import integrate
from scipy import fftpack
import os

class retrieve:

  @staticmethod
  def sample_EELS(experiment):
    silicone = struct.Materials('Silicone',4,28.0855,2.33e3,1e16)

    Ep = func.plasmon_energy(c.hbar,silicone.n(c.u),c.e,c.epsilon_0,c.me)

    E = np.linspace(0.1,80,1000)*c.e
    L = 100e-6

    theta_e = E/(experiment.gamma*experiment.m*experiment.v**2)

    factor = Ep**2/(np.square(E) +(silicone.Damping*c.hbar)**2)
    epsilon1 = 1-factor
    epsilon2 = np.divide(factor*silicone.Damping*c.hbar,E)
    epsilon2 = np.dot(epsilon2,np.complex(0,1))
    epsilon = epsilon1 + epsilon2

    Gamma2 = np.divide((c.hbar*silicone.Damping*E*Ep**2),((np.square(E) - Ep**2)**2 + (
    c.hbar*E*silicone.Damping)**2))
    Gamma3 = np.log(1+(experiment.phi_out/theta_e)**2)
    Sb = np.multiply(Gamma2,Gamma3)

    Sb = Sb*np.random.uniform(1,10)

    return np.transpose(np.array((E/c.e,Sb)))

  @staticmethod
  def mgo():
    data = []
    with open('DATA/MgO-lowloss.csv') as csvfile:
      readCSV = csv.reader(csvfile, delimiter=',')
      for row in readCSV:
        row = [float(x) for x in row]
        data.append(row)
```

```
46
47      data = np.asarray(data)
48
49      return data
50
51    @staticmethod
52    def MoS2():
53      file_object  = open("DATA/MoS2.eloss", "r")
54      data = file_object.read()
55      data = data.strip()
56      data = data.split("\n")
57      data = [x.split(" ") for x in data]
58      data = [list(filter(None, x)) for x in data]
59      data = np.asarray(data)
60      data = data.astype(float)
61
62      return data
63
64    @staticmethod
65    def Nanowire():
66      data = []
67      with open('DATA/Nanowire/Nanowire.csv') as csvfile:
68        readCSV = csv.reader(csvfile, delimiter=';')
69        for row in readCSV:
70          row = [x.replace(',','.') for x in row]
71          row = np.asarray(row)
72          row[row==''] = '0'
73          row = [float(x) for x in row]
74          data.append(row)
75
76      data = np.asarray(data)
77
78      return data
79
80    @staticmethod
81    def Nanowire_thickness():
82      t = []
83      with open('DATA/Nanowire/Nanowire-thickness.csv') as csvfile:
84        readCSV = csv.reader(csvfile, delimiter=';')
85        for row in readCSV:
86          row = [x.replace(',','.') for x in row]
87          row = [float(x) for x in row]
88          t.append(row[1])
89
90      return np.asarray(t)*1e-9 #scale to nm
91
92    @staticmethod
93    def Nanowire_ZLP_intensities():
94      I0 = []
95      count = len([name for name in os.listdir('DATA/Nanowire/withPCAwithZLP/')])
96      for i in range(1,count+1):
97        data = []
98        with open('DATA/Nanowire/withPCAwithZLP/sp'+str(i)+'withZLP.msa') as csvfile:
99          readCSV = csv.reader(csvfile, delimiter=',')
100         for row in readCSV:
101           if(row[0][0] != '#'):
102             row = [x.replace(',','.') for x in row]
103             row = [float(x) for x in row]
104             data.append(row)
105
106       data = np.asarray(data)
107       I0.append(integrate.simps(data[:,1],data[:,0]*c.e))
108
109       if(i==10):
110         plt.plot(data[:,0],data[:,1],color=struct.color('1'),linestyle='-',label="ZLP
    10",linewidth=1)
111         plt.xlabel("Energy (eV)")
112         plt.ylabel("Intensity")
113         plt.legend()
114         plt.grid(color=struct.color('grid'), linestyle='--', linewidth=0.5)
115         plt.savefig("IMAGES/zlp10.png",dpi=200)
```

```
        plt.close()

    return np.asarray(I0)

def Nanowire_ZLP(j):
    data = []
    with open('DATA/Nanowire/withPCAwithZLP/sp'+str(j)+'withZLP.msa') as csvfile:
        readCSV = csv.reader(csvfile, delimiter=',')
        for row in readCSV:
            if(row[0][0] != '#'):
                row = [x.replace(',','.') for x in row]
                row = [float(x) for x in row]
                data.append(row)

    data = np.asarray(data)

    return data

@staticmethod
def MoS2_epsilon():
    file_object  = open("DATA/MoS2_unit_relaxed_nlvdw_optic.epsilon", "r")
    data = file_object.read()
    data = data.strip()
    data = data.split("\n")
    data = [x.split(" ") for x in data]
    data = [list(filter(None, x)) for x in data]
    data = np.asarray(data)
    data = data.astype(float)

    return data

def Nanowalls():
    data = []
    with open('DATA/MoS2_nanowalls/MoS2_NW_ZZ_withoutZLP.csv') as csvfile:
        readCSV = csv.reader(csvfile, delimiter=';')
        for row in readCSV:
            row = row[0:-1]
            row = [x.replace(',','.') for x in row]
            row = np.asarray(row)
            row = [float(x) for x in row]
            data.append(row)

    data = np.asarray(data)

    return data

def Nanowalls_ZLP():
    data = []
    with open('DATA/MoS2_nanowalls/MoS2_NW_ZZ_withZLP.csv') as csvfile:
        readCSV = csv.reader(csvfile, delimiter=';')
        for row in readCSV:
            row = row[0:-1]
            row = [x.replace(',','.') for x in row]
            row = np.asarray(row)
            row = [float(x) for x in row]
            data.append(row)

    data = np.asarray(data)

    return data

def MoS2_bulk():
    data = []
    with open('DATA/MoS2_Bulk/Bulk_MoS2.csv') as csvfile:
        readCSV = csv.reader(csvfile, delimiter=';')
        for row in readCSV:
            if(row[0][0] != '#'):
                row = [x.replace(',','.') for x in row]
                row = [float(x) for x in row]
                data.append(row)

```

```
    data = np.asarray(data)

    return data

  def MoS2_bulk_zlp():
    data = []
    with open('DATA/MoS2_Bulk/Bulk_MoS2_zlp.csv') as csvfile:
      readCSV = csv.reader(csvfile, delimiter=';')
      for row in readCSV:
        if(row[0][0] != '#'):
          row = [x.replace(',','.') for x in row]
          row = [float(x) for x in row]
          data.append(row)

    data = np.asarray(data)

    return data

def backextend(EELS,d=0,M=5):
  size = EELS.shape
  N = size[0] #amount of datapoints
  if(d==0):
    d = np.power(2,6+np.ceil(np.log(N)/np.log(2))) #extend data with d amount of
    datapoints
  else:
    d = np.power(2,d)

  eres = EELS[1,0]-EELS[0,0] #energy resolution, assumed to be constant
  x = np.linspace(EELS[0,0],(d-1)*eres+EELS[0,0],d) #new xdata

  EELS_extended = np.zeros((int(d),size[1]),dtype=float) #create new matrix
  EELS_extended[:,0] = x #add new xdata to matrix
  EELS_extended[0:N,1:] = EELS[:,1:] #copy EELS data into new matrix

  xdata = EELS_extended[N-M-1:N-1,0]
  xdata = xdata - xdata[-1]

  for i in range(1,size[1]):
    popt, pcov = optimize.curve_fit(func.exponential,xdata,EELS[N-M-1:N-1,i])

    popt[0] = (EELS[-1,i]-(EELS[-2,i]-EELS[-1,i]))/np.exp(-popt[1]*eres)

    ydata_extend = func.exponential(EELS_extended[N:,0]-EELS_extended[N,0]+eres,popt
    [0],popt[1])
    EELS_extended[N:,i] = ydata_extend

  return EELS_extended

def backextend_fixed(EELS,d=0):
  size = EELS.shape
  N = size[0] #amount of datapoints
  if(d==0):
    d = np.power(2,4+np.ceil(np.log(N)/np.log(2))) #extend data with d amount of
    datapoints
  else:
    d = np.power(2,d)

  eres = EELS[1,0]-EELS[0,0] #energy resolution, assumed to be constant
  x = np.linspace(EELS[0,0],(d-1)*eres+EELS[0,0],d) #new xdata

  EELS_extended = np.zeros((int(d),size[1]),dtype=float) #create new matrix
  EELS_extended[:,0] = x #add new xdata to matrix
  EELS_extended[0:N,1:] = EELS[:,1:] #copy EELS data into new matrix

  M=10 #Amount of datapoints to fit
  xdata = EELS_extended[N-M-1:N-1,0]
  xdata = xdata - xdata[-1]

  for i in range(1,size[1]):
    #popt, pcov = optimize.curve_fit(func.exponential,xdata,EELS[N-M-1:N-1,i])
```

```
    ydata_extend = func.exponential(EELS_extended[N:,0]-EELS_extended[N,0],EELS[-1,i
    ],2)
    EELS_extended[N:,i] = ydata_extend

  return EELS_extended

def backextend_onlyzeros(EELS,d=0):
  size = EELS.shape
  N = size[0] #amount of datapoints
  if(d==0):
    d = np.power(2,6+np.ceil(np.log(N)/np.log(2))) #extend data with d amount of
    datapoints
  else:
    d = np.power(2,d)

  eres = EELS[1,0]-EELS[0,0] #energy resolution, assumed to be constant
  x = np.linspace(EELS[0,0],(d-1)*eres+EELS[0,0],d) #new xdata

  EELS_extended = np.zeros((int(d),size[1]),dtype=float) #create new matrix
  EELS_extended[:,0] = x #add new xdata to matrix
  EELS_extended[0:N,1:] = EELS[:,1:] #copy EELS data into new matrix

  M=10 #Amount of datapoints to fit
  xdata = EELS_extended[N-M-1:N-1,0]
  xdata = xdata - xdata[-1]

  for i in range(1,size[1]):
    popt, pcov = optimize.curve_fit(func.exponential,xdata,EELS[N-M-1:N-1,i])
    if(popt[1]<0.02):
      popt[1]=0.02

    #ydata_extend = func.exponential(EELS_extended[N:,0]-EELS_extended[N,0],popt[0],
    popt[1])
    ydata_extend = 0*EELS_extended[N:,0]
    EELS_extended[N:,i] = ydata_extend

  return EELS_extended

def frontextend(EELS):
  size = EELS.shape

  xmin = EELS[0,0]
  eres = EELS[1,0]-EELS[0,0]

  d = int(np.floor(xmin/eres)-1) #amount of datapoints to add

  if(d>0):
    EELS_extended = np.zeros((d+size[0],size[1]),dtype=float)
    EELS_extended[int(d):,:] = EELS

    xdata = np.linspace(EELS[0,0]-d*eres,EELS[0,0]-eres,d)
    EELS_extended[0:d,0] = xdata

    M = 10

    for i in range(1,size[1]):
      x_fit = np.append(EELS_extended[0:d,0],EELS[0:M,0])
      y_fit = np.append(EELS_extended[0:d,i],EELS[0:M,i]/EELS[0,i])
      x_fit = x_fit - EELS[0,0]
      popt, pcov = optimize.curve_fit(func.exponential_zero2,x_fit,np.absolute(y_fit))
      ydata_extend = EELS[0,i]*func.exponential_zero2(xdata-EELS[0,0],popt[0])
      EELS_extended[0:d,i] = ydata_extend

  else:
    EELS_extended = np.copy(EELS)

  return EELS_extended,int(d)

def ssd(EELS,experiment,I0,t,Z=1):
  beta = experiment.phi_out

```

```
322   Em = 7.6*Z**0.36
323   E0 = experiment.beam_energy
324
325   F = (1+E0/1022)/np.square(1+E0/511)
326   L = 106e-9*F*E0/(Em*np.log(2*beta*1e3*E0/Em))
327
328   xdata = EELS[:,0]*c.e
329   for i in range(1,np.size(EELS,1)):
330     In = integrate.simps(EELS[:,i],xdata)
331     I1 = I0[i-1]*t[i-1]/L
332     EELS[:,i] = EELS[:,i]*I1/In
333
334   return EELS
```

B

BULK.PY

The *bulk.py* file contains the class *KKA*, which contains the direct integration method and Fourier method for the Kramers Kronig analysis. Furthermore, it contains functions to calculate the bulk contribution, energy-loss function and retardation contribution.

```
import numpy as np
from scipy import fftpack
from scipy import integrate
import constants as c
import structure as struct
import functions as func

import matplotlib as mpl
mpl.use('Agg')
import matplotlib.pyplot as plt

class kka:

  @staticmethod
  def integrate_eels(EELS):
    '''
    This function uses the Kramers-Kronig Analysis to compute the dielectric function
    Make sure the energy loss data (typically the x-data) is in J in stead of eV
    Input EELS = Im[-1/epsilon] should be a (Nx2) numpy matrix of type float
    The calculation uses the parameter B = Real[1/epsilon]
    '''

    # Due to a discontinuity in the integral, we need to take a limit using the Chauchy
     principle part of the integral.
    # We do this by simple removing the point that causes the discontinuity
    N = np.size(EELS,0)
    r = EELS[1,0]-EELS[0,0]

    B = np.zeros((N,2),dtype=float)

    for i in range(0,N):
      EELS_integrate = np.delete(EELS,i,0)
      x = EELS_integrate[:,0]
      y = EELS_integrate[:,1]
      y = np.multiply(y,np.divide(x,np.square(x) - np.square(EELS[i,0])))

      Bi = 1 - (2/np.pi)*integrate.simps(y,x)
      B[i,:] = np.array([EELS[i,0],Bi])

      ,,,
      #You can uncomment this data to view an example of the integration, to view the
    singularity
      if(i==150):
        print("Plotted KKA at "+str(EELS[i,0]/c.e)+"eV")
        plt.plot(x/c.e,y,color=struct.color('1'),linestyle='-',label="kka",linewidth=1)
        plt.xlabel("Energy (eV)")
```

```
45          plt.ylabel("Intensity")
46          plt.legend()
47          plt.grid(color=struct.color('grid'), linestyle='--', linewidth=0.5)
48          plt.savefig("IMAGES/kka.png",dpi=500)
49          plt.close()'''
50
51      epsilon = np.divide(np.complex(0,1)*EELS[:,1] + B[:,1],np.square(EELS[:,1]) + np.
        square(B[:,1]))
52
53      return epsilon
54
55    @staticmethod
56    def fft(EELS):
57      '''
58      Input: (NxM) numpy matrix, with the first column the x-data (energies) in eV
59      Output: (NxM-1) numpy matrix with the dielectric functions
60      Use fourier transforms to retrieve the dielectric function
61      For this function to work properly, the data has to be extended using the data.
        extend function
62      '''
63      nn = np.size(EELS,0)
64
65      if(not nn%2 == 0):
66        #If the amount of datapoints isn't even, we have a problem
67        #Luckily, data.extend always returns an even amount of datapoints
68        struct.ColorPrint.print_fail("Error in bulk.kka.fft(EELS): Amount of datapoints N
         must be even.")
69
70
71      h = int(nn/2)
72      q = fftpack.fft(EELS[:,1:],n=nn,axis=0)
73
74      q[0:h,:] = (2/np.pi) * np.imag(q[0:h,:])
75      q[h:,:] = (-2/np.pi) * np.imag(q[0:h,:])
76      q=np.real(q)
77
78      p = fftpack.ifft(q,n=nn,axis=0)*2*np.pi
79      p = 1+np.real(p)
80
81      epsilon = np.divide(np.complex(0,1)*EELS[:,1:] + p,np.square(EELS[:,1:]) + np.
        square(p))
82
83      return epsilon
84
85  def ELF(EELS,experiment,I0,t):
86    '''
87    Input: (NxM) numpy matrix with EELS spectra, with in the first column the x-data (
        energies) in eV, and in the other M-1 column intensities/counts
88    Output: (NxM) energy loss
89    This function gets rid of the factor and angular dependence in the bulk plasmon
        contribution
90    '''
91    ELF = np.copy(EELS)
92    theta_e = EELS[:,0]*c.e/(experiment.gamma*experiment.m*experiment.v**2)
93    angular_dependence = np.log(1+np.square(np.divide(experiment.phi_out,theta_e)))
94
95    factor = np.multiply(I0,t)/(np.pi*c.a0*c.me*experiment.v**2)
96
97    ELF[:,1:] = np.divide(ELF[:,1:],np.ones((EELS.shape[0],1),dtype=float)*np.asmatrix(
        factor))
98    ELF[:,1:] = np.divide(ELF[:,1:],np.transpose(np.asmatrix(angular_dependence))*np.ones
        ((1,EELS.shape[1]-1),dtype=float))
99
100   return ELF
101
102 def contribution(E,epsilon,experiment,I0,t):
103   '''
104   Calculates the bulk contribution from the dielectric permittivity
105   '''
106   EELS = np.imag(-1/epsilon)
107   EELS_shape = EELS.shape
```

```
108
109   ELF = np.zeros((EELS_shape[0],EELS_shape[1]+1),dtype=float)
110   ELF[:,1:] = EELS
111   ELF[:,0] = E
112
113   beta = experiment.phi_out
114   theta_e = E*c.e/(experiment.gamma*experiment.m*experiment.v**2)
115   angular_dependence = np.log(1+np.square(np.divide(beta,theta_e)))
116
117   factor = np.multiply(I0,t)/(np.pi*c.a0*c.me*np.square(experiment.v))
118
119   #We now need to multiply each nth column of EELS[:,1:] by I0[n],
120   #And multiply each nth row of EELS[:,1:] by theta_e
121   o = np.ones((ELF.shape[0],1),dtype=int)
122   factor_o = np.asmatrix(factor)
123   factor_o = o*factor_o
124   ELF[:,1:] = np.multiply(ELF[:,1:],factor_o)
125
126   angular = np.transpose(np.asmatrix(angular_dependence))
127   o = np.ones((1,ELF.shape[1]-1),dtype=int)
128   angular_o = angular*o
129   ELF[:,1:] = np.multiply(ELF[:,1:],angular_o)
130
131   return ELF
132
133 def retardation(EELS,epsilon,experiment,I0,t):
134   contribution = np.zeros(epsilon.shape,dtype=float)
135
136   t = t[0]
137   I0 = I0[0]
138
139   factor = I0/(np.pi**2*c.a0*experiment.m*experiment.v**2)
140   factor = factor*2*np.pi
141
142   theta_e = np.transpose(np.asmatrix(EELS[:,0]*c.e/(experiment.gamma*experiment.m*
        experiment.v**2)))
143   theta_e = theta_e*np.ones((1,np.size(EELS,0)))
144
145   theta = np.linspace(0,experiment.phi_out,np.size(EELS,0))
146   theta = np.asmatrix(theta)
147   theta = np.ones((np.size(EELS,0),1))*theta
148
149   epsilon = epsilon*np.ones((1,np.size(EELS,0)))
150
151   lambda2 = np.square(theta)-np.multiply(np.conj(epsilon),np.square(theta_e))*np.square
        (experiment.v/c.c)
152   phi2 = lambda2 + np.square(theta_e)
153   mu2 = 1 - np.conj(epsilon)*np.square(experiment.v/c.c)
154
155   cer = factor*np.imag(np.divide(t*mu2,np.multiply(np.conj(epsilon),phi2)))
156   cer = np.multiply(cer,np.sin(theta))
157   contribution[:,0] = integrate.simps(cer,theta,axis=1)
158
159   return contribution
```

C

SURFACE.PY

The *surface.py* file contains two functions which determine the surface and thin surface contribution.

```
import numpy as np
import constants as c
import functions as func
from scipy import integrate

def correction(EELS,epsilon,eta,experiment,I0):
  '''
  Estimates the surface loss from epsilon
  '''
  T = 0.5*c.me*experiment.v**2
  factor = np.asmatrix(I0/(np.pi*c.a0*experiment.k*T))

  theta_e = EELS[:,0]*c.e/(experiment.gamma*experiment.m*np.square(experiment.v))

  S1 = np.transpose(np.asmatrix(np.divide(np.arctan(np.divide(experiment.phi_out,
    theta_e)),theta_e) - np.divide(experiment.phi_out,(experiment.phi_out**2 + np.
    square(theta_e)))))

  #S1 = np.asmatrix(np.divide(np.arctan(np.divide(experiment.phi_out,theta_e)),theta_e)
     - np.divide(experiment.phi_out,(experiment.phi_out**2 + np.square(theta_e))))
  S2 = np.imag(np.divide(-4,epsilon+eta)+1/epsilon+1/eta)

  S1 = S1*np.ones((1,np.size(EELS,1)-1),dtype=int) #Provides correct dimensions
  factor = np.ones((np.size(EELS,0),1),dtype=int)*factor

  SS = np.multiply(S1,S2)
  SS = np.multiply(factor,SS)
  return SS

def thin_correction(EELS,epsilon,experiment,I0,t,epsa=1):
  thin_correction = np.zeros(epsilon.shape,dtype=float)
  T = 0.5*c.me*np.square(experiment.v)
  theta_e = np.transpose(np.asmatrix(EELS[:,0]*c.e/(experiment.gamma*experiment.m*
    experiment.v**2)))
  theta_e = theta_e*np.ones((1,np.size(EELS,0)))

  factor = np.asarray(2*I0/(np.pi*c.a0*experiment.k*T))

  theta_i=0*np.pi

  E = np.transpose(np.asmatrix(EELS[:,0]))
  E = c.e*E*np.ones((1,np.size(EELS,0)))

  for i in range(1,np.size(EELS,1)):
    theta = np.linspace(experiment.phi_out/1e6,experiment.phi_out,np.size(EELS,0))
    theta = np.asmatrix(theta)
    theta = np.ones((np.size(EELS,0),1))*theta
    q = experiment.k*(theta*np.cos(theta_i)+theta_e*np.sin(theta_i))

```

```
46      epsm = np.asmatrix(epsilon[:,i-1])
47      epsm = np.transpose(np.ones((np.size(epsilon,0),1))*epsm)
48      epsa = np.asmatrix(epsa[:,i-1])
49      epsa = np.transpose(np.ones((np.size(epsilon,0),1))*epsa)
50
51      f1 = np.divide(theta,np.square(np.square(theta)+np.square(theta_e)))
52      f21 = np.divide(np.square(epsa-epsm),np.multiply(np.square(epsa),epsm))
53
54      Rc1 = np.divide(np.square(np.multiply(epsa,np.sin(t[i-1]*E/(2*c.hbar*experiment.v))
        )),epsm+np.multiply(epsa,np.tanh(q*t[i-1]/2)))
55      Rc2 = np.divide(np.square(np.multiply(epsa,np.cos(t[i-1]*E/(2*c.hbar*experiment.v))
        )),epsm+np.multiply(epsa,func.coth(q*t[i-1]/2)))
56
57      R = np.divide(np.square(np.sin(t[i-1]*E/(2*c.hbar*experiment.v))),epsm+np.multiply(
        epsa,np.tanh(q*t[i-1]/2)))+np.divide(np.square(np.cos(t[i-1]*E/(2*c.hbar*experiment
        .v))),epsm+np.multiply(epsa,func.coth(q*t[i-1]/2)))
58      R = 1/(epsm+np.multiply(epsa,np.tanh(q*t[i-1]/2)))+1/(epsm+np.multiply(epsa,func.
        coth(q*t[i-1]/2)))
59
60      f2 = np.multiply(f21,Rc1+Rc2)
61      f2 = np.imag(f2)
62
63      correction = factor[i-1]*np.multiply(f1,f2)
64      y = np.multiply(correction,np.sin(theta))
65
66      correction = integrate.simps(y,theta,axis=1)
67      thin_correction[:,i-1] = correction
68
69  return thin_correction
```

D

FUNCTIONS.PY

The *functions.py* file contains general functions, which, for example, plot entire data sets simultaneously.

```
import numpy as np
import matplotlib as mpl
mpl.use('Agg')
import matplotlib.pyplot as plt
import structure as struct
import bulk
import surface
import csv

def plasmon_energy(hbar,n,e,epsilon,m):
  '''
  Returns the plasmon energy of a material as a float
  '''
  return hbar*np.sqrt(n*e**2/(epsilon*m))

def plot_data(data,folder="IMAGES/DATA/",lab="Data"):
  '''
  Plots all the data in the IMAGES/DATA folder
  Assumes an (NxM) numpy matrix, with M-1 the amount of EELS spectra, in the first
    column the x-data (energy loss) in eV, and N the amount of datapoints
  '''
  size = data.shape
  for i in range(1,size[1]):
    plt.plot(data[:,0],data[:,i],color=struct.color('1'),linestyle='-',label=lab,
    linewidth=1)
    plt.xlabel("Energy-loss (eV)")
    plt.ylabel("Counts")
    plt.legend()
    plt.grid(color=struct.color('grid'), linestyle='--', linewidth=0.5)
    plt.savefig(folder+str(i)+".png",dpi=100)
    plt.close()

def plot_epsilon(xdata,ydata,folder="IMAGES/EPSILON/"):
  '''
  Plots real and imaginary parts of the dielectric functions in ydata
  Assumes xdata as a (1xN) numpy array (in eV), and ydata as a (NxM) matrix
  Images are stored in IMAGES/EPSILON/ as default
  '''
  for i in range(0,np.size(ydata,1)):
    plt.plot(xdata,np.real(ydata[:,i]),color=struct.color('1'),linestyle='-',label="$\
    epsilon_1$",linewidth=1)
    plt.plot(xdata,np.imag(ydata[:,i]),color=struct.color('2'),linestyle='-',label="$\
    epsilon_2$",linewidth=1)
    plt.xlabel("Energy-loss (eV)")
    plt.ylabel("Dielectric function")
    plt.legend()
    plt.grid(color=struct.color('grid'), linestyle='--', linewidth=0.5)
    plt.savefig(folder+str(i+1)+".png",dpi=100)
    plt.close()
```

```
46
47 def plot_bulk(xdata,ydata,folder="IMAGES/BULK/",lab=""):
48   '''
49   Plots real and imaginary parts of the dielectric functions in ydata
50   Assumes xdata as a (1xN) numpy array (in eV), and ydata as a (NxM) matrix
51   Images are stored in IMAGES/EPSILON/ as default
52   '''
53   for i in range(0,np.size(ydata,1)):
54     plt.plot(xdata,ydata[:,i],color=struct.color('1'),linestyle='-',label=lab,linewidth
     =1)
55     plt.xlabel("Energy-loss (eV)")
56     plt.ylabel("Counts")
57     plt.legend()
58     plt.grid(color=struct.color('grid'), linestyle='--', linewidth=0.5)
59     plt.savefig(folder+str(i+1)+".png",dpi=100)
60     plt.close()
61
62 def plot_surface(xdata,ydata,folder="IMAGES/SURFACE/",lab=""):
63   '''
64   Plots real and imaginary parts of the dielectric functions in ydata
65   Assumes xdata as a (1xN) numpy array (in eV), and ydata as a (NxM) matrix
66   Images are stored in IMAGES/EPSILON/ as default
67   '''
68   for i in range(0,np.size(ydata,1)):
69     plt.plot(xdata,ydata[:,i],color=struct.color('1'),linestyle='-',label=lab,linewidth
     =1)
70     plt.xlabel("Energy-loss (eV)")
71     plt.ylabel("Counts")
72     #plt.legend()
73     plt.grid(color=struct.color('grid'), linestyle='--', linewidth=0.5)
74     plt.savefig(folder+str(i+1)+".png",dpi=100)
75     plt.close()
76
77 def unextend(a,n):
78   return a[0:n]
79
80 def exponential(x,a,b):
81   return a*np.exp(-b*x)
82
83 def exponential_zero(x,a,b):
84   return a*np.exp(b*x)-a
85
86 def exponential_zero2(x,b):
87   return np.exp(b*x)
88
89 def coth(x):
90   return np.divide(np.cosh(x),np.sinh(x))
91
92 def write_csv(data,filename):
93   with open(filename,'w') as csvfile:
94     filewriter = csv.writer(csvfile,delimiter=';',quotechar='|',quoting=csv.
     QUOTE_MINIMAL)
95     for i in range(0,np.size(data)):
96       a = [str(np.real(data[i])[0]),str(np.imag(data[i])[0])]
97       filewriter.writerow(a)
98
99 def gauss(x,o):
100   return (1/(o*np.sqrt(np.pi))) * np.exp(-np.square(x/o))
```

E

OTHER SCRIPTS

Two other files have been written. The file *constants.py* contain all the required physical constants for the calculations.

```
import numpy as np

e = 1.602e-19 # Electric charge
me = 9.109e-31 # Electron mass
epsilon_0 =  8.854e-12 # Vacuum permittivity
c = 299792458 # Speed of light
u = 1.66e-27 # Atomic mass unit
h = 6.62607e-34 # Plank's constant
hbar = h/(2*np.pi) # Reduced Plank's constant
a0 = 5.2917721067e-11 # Bohr's radius
```

The file *structure.py* contains a class for the experiment, colors for the plots and a class which prints outputs to the command line in different colors.

```
import numpy as np
import sys

def color(c = 'black'):
  #standard colors:
  if(c=='white'):
    return "#FFFFFF"
  elif(c=='grey'):
    return "#A2A2A2"
  elif(c=='black'):
    return "#333333"
  elif(c=='dark-blue'):
    return "#3460CF"
  #colors for plots:
  elif(c=='grid'):
    return "#A2A2A2"
  elif(c=='1'):
    return "#377eb8"
  elif(c=='2'):
    return "#e41a1c"
  elif(c=='3'):
    return "#4daf4a"
  elif(c=='4'):
    return "#ff7f00"

class ColorPrint:

    @staticmethod
    def print_fail(message, end = '\n'):
        sys.stderr.write('\x1b[1;31m' + message.strip() + '\x1b[0m' + end)

    @staticmethod
    def print_pass(message, end = '\n'):
        sys.stdout.write('\x1b[1;32m' + message.strip() + '\x1b[0m' + end)
```

```
35
36      @staticmethod
37      def print_warn(message, end = '\n'):
38          sys.stderr.write('\x1b[1;33m' + message.strip() + '\x1b[0m' + end)
39
40      @staticmethod
41      def print_info(message, end = '\n'):
42          sys.stdout.write('\x1b[1;34m' + message.strip() + '\x1b[0m' + end)
43
44      @staticmethod
45      def print_bold(message, end = '\n'):
46          sys.stdout.write('\x1b[1;37m' + message.strip() + '\x1b[0m' + end)
47
48  class Experiment:
49    def __init__(self,m,c,h,e,E,phi_out):
50      self.beam_energy = E #in keV
51      self.phi_out = phi_out #in radians
52      self.m = m #in kg
53
54      E = E*1e3*e
55      v = c*np.sqrt(1-((m*c**2)/(E + m*c**2))**2)
56      #v = c*np.sqrt(1-1/((E/(m*c**2)+1)**2))
57      gamma = 1/np.sqrt(1-(v/c)**2)
58      p = gamma*m*v
59      l = h/p
60
61      self.v = v
62      self.gamma = gamma
63      self.p = p
64      self.l = l
65      self.k = 2*np.pi/l
```

F

EXAMPLE SCRIPT

An example of the functionality of the script is displayed in the code below. This is specifically the code used to run the script for the MoS_2 bulk spectrum.

```
# Import libraries
import numpy as np

# Import our files
import constants as c
import structure as struct
import functions as func
import data
import bulk
import surface

from scipy import fftpack
from scipy import integrate
from scipy.signal import find_peaks
from scipy.signal import peak_widths
from scipy.signal import lfilter
from scipy.signal import savgol_filter
from scipy.signal import convolve
from scipy.signal import deconvolve
from scipy.optimize import curve_fit

import dask.array as da

import matplotlib as mpl
mpl.use('Agg')
import matplotlib.pyplot as plt

#Set experiment
experiment = struct.Experiment(c.me,c.c,c.h,c.e,300,2.95e-3)

I0 = data.retrieve.MoS2_bulk_zlp()

EELS_data = data.retrieve.MoS2_bulk()

I0 = integrate.trapz(I0[:,1],I0[:,0]*c.e)-integrate.trapz(EELS_data[:,1],EELS_data
    [:,0]*c.e)
t = 84e-9

EELS_data = EELS_data[187:486]

I0 = np.asarray([I0])
t = np.asarray([t])

n = np.size(EELS_data,0)

EELS = np.copy(EELS_data)
SSD = np.copy(EELS_data)
```

```
SSD,d = data.frontextend(SSD)
plot_lim=400
z=n

plot_lim_low = plot_lim
if(plot_lim_low>(z)):
  plot_lim_low = z

SSD = data.backextend(SSD)
SSD = data.ssd(SSD,experiment,I0,t,L=0)

EELS_data = EELS_data[0:z+d]

SSD_data = np.copy(SSD)

l=60000

SSD_bulkguess = np.copy(SSD)

#print(eta.shape)

for i in range(0,l):
  #Get ELF
  ELF = bulk.ELF(SSD,experiment,I0,t)

  #Get bulk dielectric function
  epsilon = bulk.kka.fft(ELF)

  surface_loss = surface.correction(SSD[0:z],epsilon[0:z],eta[0:z],experiment,I0)

  cer = bulk.cerenkov(SSD[0:z],epsilon[0:z],experiment,I0,t)

  SSD = np.copy(SSD_bulkguess)
  SSD[0:z,1:] -= surface_loss[0:z,:]
  SSD[0:z,1:] -= rad[0:z,:]

  convergence_req = np.sum(surface_loss[0:np.size(EELS,0),:])

  if(i>0):
    convergence_req = np.mean(np.abs(SSD[0:z+d,1]-SSD_prev[0:z+d,1]))
    if(convergence_req<1e-3):
      struct.ColorPrint.print_pass("Converged at i="+str(i))
      break
    else:
      struct.ColorPrint.print_info("Iteration "+str(i)+" with error="+str(
    convergence_req))

  SSD_prev = np.copy(SSD)

n = np.sqrt(np.sqrt(np.square(np.real(epsilon))+np.square(np.imag(epsilon))) + np.real(
    epsilon))/np.sqrt(2)
alpha = np.sqrt(np.sqrt(np.square(np.real(epsilon))+np.square(np.imag(epsilon))) - np.
    real(epsilon))/np.sqrt(2)
R = np.divide(np.square(n-1)+np.square(alpha),np.square(n+1)+np.square(alpha))

f = "IMAGES/ITERATIONS/0/"
func.plot_bulk(SSD[0:plot_lim_low,0],SSD[0:plot_lim_low,1:],folder=f+"BULK/",lab="Bulk
    loss at j="+str(i))
func.plot_surface(SSD[0:plot_lim_low,0],surface_loss[0:plot_lim_low],folder=f+"SURFACE/
    ",lab="Surface loss at j="+str(i))
func.plot_surface(SSD[0:plot_lim_low,0],R[0:plot_lim_low],folder=f+"SURFACE_TOT/",lab="
    Tot surface loss at j="+str(i))
func.plot_epsilon(SSD[0:plot_lim_low,0],epsilon[0:plot_lim_low],folder=f+"EPSILON/")
func.plot_bulk(SSD[0:plot_lim_low,0],n[0:plot_lim_low],folder=f+"THIN_SURFACE/")
func.plot_bulk(SSD[0:plot_lim_low,0],rad[0:plot_lim_low],folder=f+"RETARDATION/")


peak = SSD[np.imag(epsilon[0:500]).argmax(),0]
f = np.abs(np.real(epsilon[0:500])-np.imag(epsilon[0:500]))
z = np.argsort(f,axis=0)
f2 = np.abs(np.real(epsilon[0:100])-np.imag(epsilon[0:100]))
```

```
z2 = np.argsort(f2,axis=0)

print(np.real(epsilon[0]))
print(SSD[z[0],0])
print(SSD[z2[0],0])

peak1 = SSD[SSD[0:500,1].argmax(),0]
peak2 = SSD[surface_loss[0:500].argmax(),0]

print(peak1)
print(peak2)

plt.plot(SSD[0:plot_lim_low,0],n[0:plot_lim_low],color=struct.color('1'),linestyle='-',
    label="Nanowall",linewidth=1)
plt.xlabel("Energy-loss (eV)")
plt.ylabel('n')
plt.grid(color=struct.color('grid'), linestyle='--', linewidth=0.5)
plt.savefig("IMAGES/ITERATIONS/DATA/n.png",dpi=200)
plt.close()

plt.plot(SSD[0:plot_lim_low,0],alpha[0:plot_lim_low],color=struct.color('1'),linestyle=
    '-',label="Nanowall",linewidth=1)
plt.xlabel("Energy-loss (eV)")
plt.ylabel(r'$\alpha$')
plt.grid(color=struct.color('grid'), linestyle='--', linewidth=0.5)
plt.savefig("IMAGES/ITERATIONS/DATA/alpha.png",dpi=200)
plt.close()

plt.plot(SSD[0:plot_lim_low,0],R[0:plot_lim_low],color=struct.color('1'),linestyle='-',
    label="Nanowall",linewidth=1)
plt.xlabel("Energy-loss (eV)")
plt.ylabel('R')
plt.grid(color=struct.color('grid'), linestyle='--', linewidth=0.5)
plt.savefig("IMAGES/ITERATIONS/DATA/R.png",dpi=200)
plt.close()
```