hsph_Csca_V5 - Guide

This python code allows the user to perform controlled simulations using ADDA 1.4.0, read and save the results, while integrating the scattering cross section into two angular hemispheres. The code is featuring also a user-friendly GUI for Windows environment (Windows 10 and 11 have been tested, and python 3.11.11). Support for Linux is scheduled soon.

Please, read carefully this guide and the comments in the code as it is still an experimental version.

Acknowledgements

This code was primarily written by Mattia Andrini, with the contribution of Paolo Zuccala' Maganzini (Physics department of the Catholic University of Brescia). Any comments, inquiries, and suggestions are welcome. Please, send an email to mattia.andrini@unicatt.it.

A paper making use of this version of the code is currently in preparation. In the meantime, you can cite[1] as its core functions were originally developed there.

Files requirement

The code should be working out of the box by running it. If the ADDA win64 folder is present in a close directory, the code should automatically locate and copy the needed file (adda.exe, adda_ocl.exe, clFFT.dll, libfftw3-3.dll), to the folder where the script is located (*Mainfolder*) and generate the folder where the simulation results will be saved (*SimulationResults)*. If this is not the case, you should download these files (they can be found inside "**adda-1.4.0-win64.zip**" for example) and place them in the same folder of the script.

A .txt file containing the optical constants for your simulation must be provided (**Input File**). The file should contain four columns separated by spaces: **Wavelength (nm)**, **Scatterer Real Refractive Index (RI) (*n*)**, **Scatterer Imaginary RI (*k*)**, and **Substrate RI**.

Even when not using surface mode, you must fill the fourth column with a placeholder value (e.g., 1.0). The software will override this entry with 1.0 for free-space calculations. Do not include any text or headers; use raw numbers only (e.g., 224 1.72 0.166 1.92). For spectral simulations, list each wavelength on a new line. Refer to lambda_nk_test.txt and Figure 1 for an example.

```
200     1.722   0.166   1.91
204     1.734   0.139   1.90
208     1.746   0.12    1.89
212     1.758   0.11    1.88
```

Figure 1: example of the input file template, where the user can select the light wavelength, and the material and substrate RI.

Simulation parameters

Running the code should open a GUI window (Figure 2), where the user can define the following parameters.

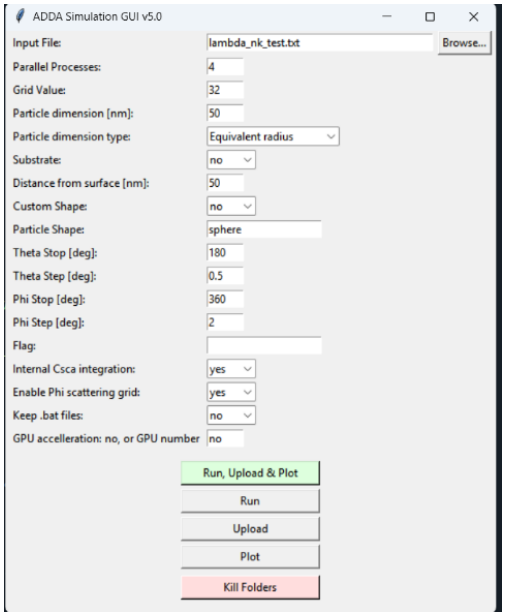

Figure 2. GUI window for performing simulations.

- **Input File:** Click "Browse" to select your .txt file containing the optical parameters.
- **Parallel Processes:** Sets the number of simulations to run simultaneously (one per wavelength). This utilizes multiple CPU cores to speed up spectral calculations but at the moment does not accelerate single-wavelength simulations.
- **Grid Value:** Defines the number of dipoles along the x-axis for the discretization of the particle.
- **Particle dimension** (nm): The size of the scatterer in nanometers. This corresponds to either the equivalent disc radius or the size along the x-axis, depending on the selection in **"Particle dimension type"**.
- **Substrate**: “Select "yes" to enable substrate mode or "no" for free-space simulations.
- **Distance between particle center and surface** (nm): Sets the distance between the particle center and the substrate surface (only used if Substrate is "yes").
- **Custom shape**:
    - **No**: Use a standard ADDA shape. Enter the shape type (e.g., sphere) in the **"Particle Shape"** box (see ADDA Guide, Sec 6.4). *Note: Complex shapes requiring extra parameters may need manual updates to the generate_bat_file function (code line 172).*
    - **Yes**: Use a user-defined shape. Enter the name of your .geom file in the **"Particle Shape"** box (omit the .geom extension, the file must be present in the *Mainfolder*).
- **Theta Stop and Step:** defines the theta scattering grid in degrees. See Figure. 3 for the angle convention.
    - The range always starts at $\theta = 0°$,
    - The usual stop value (**Theta Stop**) is $\theta = 180°$.
    - **Theta Step** (degrees): sampling interval for the θ angle. A smaller step size results in a finer grid and higher angular resolution.
- **Phi Stop and Step:** defines the phi scattering grid in degrees. See Figure. 3 for the angle convention.
    - The range always starts at $\varphi = 0°$,
    - The usual stop (**Phi Stop**) value is $\varphi = 360°$.

    - **Phi Step** (degrees): sampling interval for the φ angle. A smaller step size results in a finer grid and higher angular resolution.
- **Flag:** An optional text string appended to output filenames. Use this to tag different simulation sets and prevent overwriting. **Do not use spaces.**
- **Internal Csca integration:** If "yes", the command "-Csca" is added to the line sent to ADDA. An internal integration of the scattering cross-section is then performed. This is useful for verifying consistency with the hemispherical integration performed by this script but lengthens the simulation time.
- **Enable Phi scattering grid:**
    - **No:** Recommended for particles symmetric in the x-y plane (rotationally symmetric). ADDA will calculate the Mueller matrix dependent only on θ, optimizing the simulation. The grid is defined solely by the "Theta" parameters.
    - **Yes:** Computes the full θ and φ dependence of the Mueller matrix.
        - **Important**: In this mode, ADDA reads the grid definition from scat_params.dat. The Python script automatically overwrites this file every time to match your GUI inputs. All functions related to this mode are suffixed with *_grid* (e.g., *upload_grid()*).
- **Keep .bat files (debug option):**
    - **No (Default):** Automatically deletes batch files after the simulation to keep the folder clean.
    - **Yes:** Retains all .bat files in the *Mainfolder*. Use this for debugging by manually running the generated commands to check for errors.
- **GPU acceleration:**
    - **No:** Disables GPU acceleration. The simulation will run on the CPU using the standard adda.exe.
    - **[Number]:** Enables OpenCL acceleration. Enter the Device ID of the GPU you wish to use (typically 0 for the primary card). When a number is provided, the script automatically searches for and uses the adda_ocl.exe executable to run the simulation on the specified graphics card.

Simulation Execution & Output

- **Folder Naming Convention:** Simulations are automatically saved in the *Mainfolder* using a dynamic naming convention that reflects the input parameters:
  run{lambda}_{shape}_sub{substrate}_g{grid}_m{n}_R{radius}_{Flag}
  (e.g., run200_sphere_subno_g32_m1.5_R50_test)

- **Run:** Initiates the simulation process. Upon execution, the console will display "Starting parallel simulations...", followed by the computation time for each wavelength and the total runtime.

- **Upload:** Reads the raw data from the simulation folders, performs the necessary integrations and cross-section calculations (detailed in the *Calculations* section), and saves the resulting summary .txt files into the *SimulationResults* folder.

- **Plot:** Generates graphs visualizing the key scattering quantities calculated during the upload phase, comparing ADDA results with Mie theory (if applicable) and displaying cross-sections.

- **Run, Upload and Plot:** Executes the three steps above automatically in sequence. This is the recommended workflow for standard simulations.

- **Kill Simulation Folders:** Deletes all simulation subdirectories in the *Mainfolder* that match the currently selected GUI parameters. Use this function to clean up the workspace and save disk space after verifying your results. The .txt files generated in the *SimulationResults* folder are not eliminated.

Read the results and calculate the scattering cross sections

The code calculates integrated scattering efficiencies by integrating the scattering intensity over both the polar θ and azimuthal φ angles. The integration domain is divided into two hemispheres: **Forward** and **Backward**, defined strictly relative to the propagation direction of the incident light.

Figure 3[1,2] illustrates the angular conventions adopted in this code compared to the standard ADDA reference frame:

- **Free space mode** (Figure 3a)**:** The coordinate system coincides with the default ADDA laboratory reference frame. The incident light propagates along the positive z-axis.
- **Surface mode**: The z-axis direction used by this code (Figure 3c) is **reversed** relative to the default ADDA laboratory frame (Figure 3b). This transformation is necessary to ensure that "Forward Scattering" ($0° \leq \theta < 90°$) always corresponds to the direction of the impinging light, and "Backward Scattering" ($90° < \theta \leq 180°$) corresponds to the reflection/backscattering direction, regardless of the substrate's orientation.

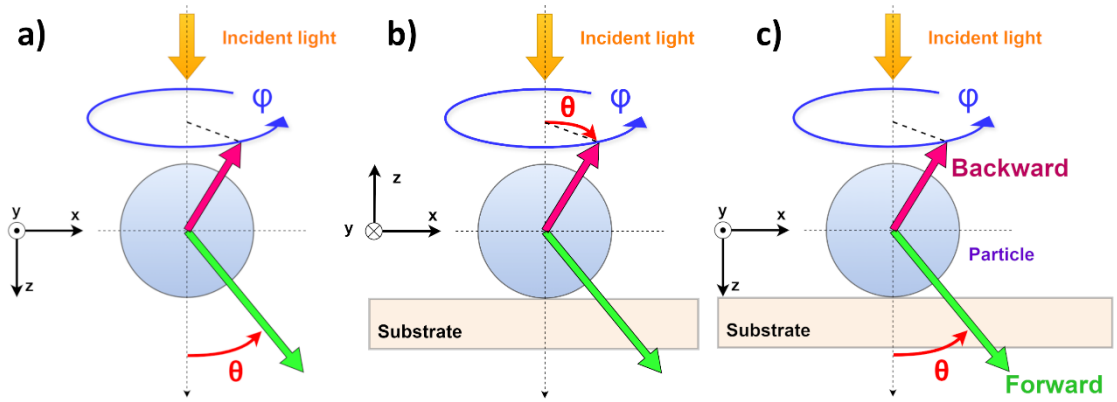

Figure 3. Definition of reference frames and scattering geometry.

a) Reference frame for free-space simulations.

b) ADDA standard laboratory reference frame for surface-mode simulations.

c) The coordinate system adopted in this code. Light scattered by the sphere (light blue) is divided into forward (green arrow) and backward (red arrow) directions. The scattering direction is defined by the polar angle θ (red) and the azimuthal angle φ. Note that θ is measured relative to the incident light propagation direction (z-axis), while φ describes the rotation around the z-axis.

To read the results, the code offers two opportunities depending on the angle grid the user has defined for the simulation.

1) **Only theta dependance** (Enable Phi scattering grid: "No"): For particles symmetric in the x-y plane (thus with respect to a φ-rotation), there should not be any significant dependence on the angle φ, thus the simulation can be optimized by storing only the θ scattering grid.
    a. Define the **theta stop** and **theta step** according to the angle resolution you need and **run** the simulation.
    b. The function *Upload* (called by the button Upload) will open each folder created during the simulation and extract the following quantities:
        i. θ grid as simulated by ADDA;
        ii. S_{11} component of the Mueller Matrix as function of θ and λ;
        iii. **Total extinction and absorption cross sections/efficiencies given by ADDA** (taken from file CrossSec-X/Y, automatically averaging for the two polarizations if needed), and the **scattering cross section** if the “Internal Csca integration” was enabled.

Then, the code computes the forward ($C_{sca}^{for}(\lambda)$) and backward ($C_{sca}^{back}(\lambda)$) scattering cross section using the equation (66) of the ADDA guide (v1.4) as function of the wavelength:

$$C_{sca}^{for}(\lambda) = \frac{\lambda^2}{2\pi * n_{sub}^2} * \sum_{\theta=0°}^{90°} S_{11}(\lambda,\theta) * sen\theta * d\theta$$

$$C_{sca}^{back}(\lambda) = \frac{\lambda^2}{2\pi * n_{sub}^2} * \sum_{\theta=90°}^{180°} S_{11}(\lambda,\theta) * sen\theta * d\theta$$

The substrate refractive index n_{sub} is considered here, and it is set to 1 for free-space mode.

Finally, the code saves in the *SimulationResults* folder two .txt files with:

- **ADDA outputs (μm^2)**:
  - C_ext_ADDA_auto (extinction cross section).
  - C_abs_ADDA_auto (absorption cross section).
  - C_sca_ADDA_int (total scattering cross section, if the Internal Csca integration was enabled).
- **Integrated cross sections (μm^2)**:
  - C_sca_ADDA_int_tot (sum of forward and backward integrated cross sections),
  - C_sca_ADDA_int_tot_for (C_{sca}^{for}, scattering cross section of the forward hemisphere),
  - C_sca_ADDA_int_tot_back (C_{sca}^{back}, scattering cross section of the backward hemisphere).
- **Efficiencies Q**:
  - **ADDA outputs** (same naming conventions of the cross sections).
  - **Corresponding Mie efficiencies** computed by the Miepython module,[3] if present, assuming spheres with radius **a**, or half the size along the x axis, and refractive index specified in "input_filename".

2) **Complete Theta-Phi grid** (Enable Phi scattering grid: "Yes"): Use this mode for particles that are **asymmetric** in the x-y plane, where the scattering may depend on both the polar angle (θ) and the azimuthal angle (φ).
   a. Define the desired angular resolution using the **theta/phi stop** and **theta/phi step** parameters in the GUI. The code will automatically update the scat_params.dat file with these settings before running the simulation.
   b. The *upload_grid* function (triggered by the **Upload** button) iterates through each simulation folder to extract the following quantities:
      i. θ and φ grid as simulated by ADDA;
      ii. S_{11} component of the Mueller Matrix as function of θ, φ and λ;
      iii. **Total extinction and absorption cross sections/efficiencies given by ADDA** (taken from file CrossSec-X/Y, averaging for the two polarizations if needed), and the **scattering cross section** if the "Internal Csca integration" was enabled.

   Then, the code computes the forward ($C_{sca}^{for}(\lambda)$) and backward ($C_{sca}^{back}(\lambda)$) scattering cross section using the equation (66) of the ADDA guide (v1.4) as function of the wavelength, considering the φ dependance:

$$C_{sca}^{for}(\lambda) = \frac{\lambda^2}{4\pi^2 * n_{sub}^2} * \sum_{\varphi=0°}^{\varphi=360°} \sum_{\theta=0°}^{90°} S_{11}(\lambda, \theta, \varphi) * sen\theta * d\varphi d\theta$$

$$C_{sca}^{back}(\lambda) = \frac{\lambda^2}{4\pi^2 * n_{sub}^2} * \sum_{\varphi=0°}^{\varphi=360°} \sum_{\theta=90°}^{180°} S_{11}(\lambda, \theta, \varphi) * sen\theta * d\varphi d\theta$$

The substrate refractive index n_{sub} is considered here, and it is set to 1 for free-space mode.

Finally, the code saves in the *SimulationResults* folder a txt files with:

- **ADDA outputs (μm^2)**:
  - C_ext_ADDA_auto (extinction cross section).
  - C_abs_ADDA_auto (absorption cross section).
  - C_sca_ADDA_int (total scattering cross section, if the Internal Csca integration was enabled).
- **Integrated cross sections (μm^2)**:
  - C_sca_ADDA_int_tot (sum of forward and backward integrated cross sections),
  - C_sca_ADDA_int_tot_for (C_{sca}^{for}, scattering cross section of the forward hemisphere),
  - C_sca_ADDA_int_tot_back (C_{sca}^{back}, scattering cross section of the backward hemisphere).
- **Efficiencies Q**:
  - **ADDA outputs** (same naming conventions of the cross sections).
  - **Corresponding Mie efficiencies** computed by the Miepython module,[3] if present, assuming spheres with radius **a**, or half the size along the x axis, and refractive index specified in "input_filename".

Bibliography

1. Andrini, M., Federici, S. & Gavioli, L. Refractive Index of Benchmark Polystyrene Nanoplastics by Optical Modeling of UV–Vis Spectra. *Anal. Chem.* **97**, 19419–19426 (2025).
2. Andrini, M., Federici, S. & Gavioli, L. Correlation of refractive index to morphology for polystyrene nanospheres by optical modelling of UV-VIS spectra. in *Multimodal Sensing and Artificial Intelligence for Sustainable Future* (eds Falldorf, C., Soldovieri, F., Bianco, V. & Picart, P.) 1 (SPIE, Munich, Germany, 2025). doi:10.1117/12.3062580.
3. Prahl, S. miepython: Pure python implementation of Mie scattering (v2.5.3). Zenodo (2023).