The Journal of Open Source Software

A GPU-Accelerated Open-Source Python Package for Calculating the Debye Scattering Equation: Applications in Powder Diffraction, Small-Angle-, and Total Scattering

Frederik L. Johansen [1,2*¶], **Andy S. Anker** [1*¶], **Ulrik Friis-Jensen** [1,2], **Erik B. Dam** [2], **Kirsten M. Ø. Jensen** [1¶], and **Raghavendra Selvan** [2,3¶]

1 Department of Chemistry & Nano-Science Center, University of Copenhagen, Denmark **2** Department of Computer Science, University of Copenhagen, Denmark **3** Department of Neuroscience, University of Copenhagen, Denmark ¶ Corresponding author * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

The Debye scattering equation, derived in 1915 by Peter Debye, is used to calculate the scattering intensities considering the position of each atom in the structure:(Debye, 1915; Scardi et al., 2016)

$$I(Q) = \sum_{\nu=1}^{N} \sum_{\mu=1}^{N} b_\nu b_\mu \frac{\sin(Qr_{\nu\mu})}{Qr_{\nu\mu}} \tag{1}$$

In this equation Q is the momentum transfer of the scattered radiation, N is the number of atoms and $r_{\nu\mu}$ is the distance between atoms ν and μ. For X-ray radiation, the atomic scattering factor, b, depends strongly on Q and is usually denoted as f(Q), but for neutrons, b is independent of Q. The Debye scattering equation can be used to compute the scattering pattern of any atomic structure and is commonly used to study both crystalline and non-crystalline materials with a range of scattering techniques like powder diffraction (PD), total scattering (TS) with pair distribution function (PDF) analysis, and small-angle scattering (SAS).(Scardi et al., 2016) Although the Debye scattering equation is extremely versatile, the computation of the double sum, which scales $O(N^2)$, has limited the practical use of the equation.

With the advancement in computer hardware,(Schaller, 1997) analysis of larger structures is now feasible using the Debye scattering equation. Modern central processing units (CPUs), ranging from tens to hundreds of cores offer an opportunity to parallelise computations, significantly enhancing compute efficiency. The same goes for graphics processing units (GPUs), which are designed with multiple cores acting as individual accelerated processing units that can work on different tasks simultaneously. In contrast, CPUs usually have fewer cores optimised for more general-purpose computing. This means that a GPU can execute multiple simple instructions in parallel, while a CPU might handle fewer parallel tasks.(Garland et al., 2008) Therefore, GPUs are better suited for calculations such as the Debye scattering equation, where many computations can be performed simultaneously. Taking advantage of such GPU acceleration could yield computational speeds that surpass those of even the most advanced multi-core CPUs; by orders of magnitude. We introduce a GPU-accelerated open-source Python package, named DebyeCalculator, for rapid calculation of the Debye scattering equation from xyz-files. The xyz-format is commonly used in materials chemistry for the description of discrete particles and simply consists of a list of atomic identities and

their respective Cartesian coordinates (x, y and z). Additionally, 'DebyeCalculator' can take a crystallographic information file (CIF) and a user-defined spherical radius as input to generate an xyz-file from which a scattering pattern is calculated. We further calculate the PDF as described in Underneath the Bragg Peaks.(Egami & Billinge, 2003) We show that our software can simulate the PD, TS, SAS and PDF data orders of magnitudes faster than DiffPy-CMI.(Juhás et al., 2015) DebyeCalculator is an open-source project that is readily available through GitHub (https://github.com/FrederikLizakJohansen/DebyeCalculator) and PyPI (XXXXXXX).

The DebyeCalculator, illustrated in the following pseudocode, begins with an initialisation function that sets user-defined parameters or sets them to default. They include aspects of the computational environment (such as Q-range, Q-step, r-range, r-step, batch size, and device), atomic vibrations, radiation type, and instrumental parameters. During this initialisation phase, the calculation of the atomic form factors is prepared based on the radiation type. Once initialised, the DebyeCalculator can compute various quantities: the scattering intensity I(Q) through the Debye scattering equation, the Total Scattering Structure Function S(Q), the Reduced Total Scattering Function F(Q), and the Reduced Atomic Pair Distribution Function G(r). In this section, we specifically illustrate the pseudocode for the G(r) calculation. This is because the process for calculating G(r) inherently involves the calculations for I(Q), S(Q), and F(Q). When calling the gr function, the DebyeCalculator loads the structure and computes the atomic form factors.(Waasmaier & Kirfel, 1995) Following this, it calculates the scattering intensity I(Q) using the Debye scattering equation and subsequently determines the structure factor S(Q). The function F(Q) is derived using Q-values and S(Q). Necessary modifications, such as dampening and Lorch modifications, are applied before computing the G(r). The DebyeCalculator outputs the calculated functions to the CPU by default to allow for immediate analysis of the results, but users have the flexibility to retain the output on the GPU. It is worth noting that the majority of the compute time is spent on the double sum calculation in the Debye scattering equation. This is where GPU acceleration can enhance performance compared to single core CPUs. The double sum calculation is then multiplied by the momentum transfers and form factors. This approach is computationally inexpensive but demands significant memory, restricting the ability to calculate these for larger structures. Therefore, the DebyeCalculator provides a batching schema that enables users with restricted access to GPU memory to compute I(Q), S(Q), F(Q) and G(r) for larger structures, trading off only a small increase in computation time. Users with more substantial GPU memory can accommodate large structures while maintaining high computation speeds.

```
CLASS DebyeCalculator:
  FUNCTION Initialise(parameters...):
      - Set class parameters based on given input or defaults
      - Setup computational environment (e.g., Q-values, r-values)
      - Load atomic formfactor coefficients
      - Setup form factor calculation based on radiation type

  FUNCTION gr(structure_path, keep_on_device=False):
      - Load atomic structure from given structure_path
      - Calculate atomic formfactors
      - Calculate scattering intensity I(Q) (Debye scattering equation)
      - Compute structure factor S(Q) based on I(Q)
      - Calculate F(Q) based on Q-values and S(Q)
      - Apply modifications if necessary (like dampening and Lorch)
      - Calculate pair distribution function G(r) based on F(Q)
      - Return G(r) either on GPU or CPU
```

In order to benchmark our implementation, we compare simulated scattering patterns from DebyeCalculator against DiffPy-CMI,(Juhás et al., 2015) which is a widely recognised software for scattering pattern computations. Here, our implementation obtains the same scattering

patterns as DiffPy-CMI (Supporting Information), while being faster on CPU for structures up to ~72 Å (Figure 1). Both calculations are run on a x86-64 CPU with 64GB of memory and a batch size of 10,000. Running the calculations on the GPU provides another notable boost in speed (Figure 1). This improvement primarily stems from the distribution of the double sum calculations across a more extensive set of cores than is feasible on the CPU. With smaller atomic structures, an overhead associated with initiating GPU calculations results in the NVIDIA RTX A3000 Laptop GPU computations being slower than DiffPy-CMI and our CPU implementation. Once the atomic structure size exceeds ~14 Å in diameter, we observe a ~5 times speed-up using an NVIDIA RTX A3000 Laptop GPU with 6GB of memory and a batch size of 10,000. The choice of GPU hardware has a substantial influence on this speed advantage. As demonstrated in Figure 1, using an NVIDIA Titan RTX GPU, which offers 24GB of memory, the speed benefits become even more evident. The NVIDIA Titan RTX GPU delivers a performance that is ~10 times faster, seemingly across all structure sizes, underlining the significant role of the hardware. With the advancements of GPUs like NVIDIA's Grace Hopper Superchip,(NVIDIA, 2023) which boasts 576GB of fast-access to memory, there is potential for DebyeCalculator to achieve even greater speeds in the future.

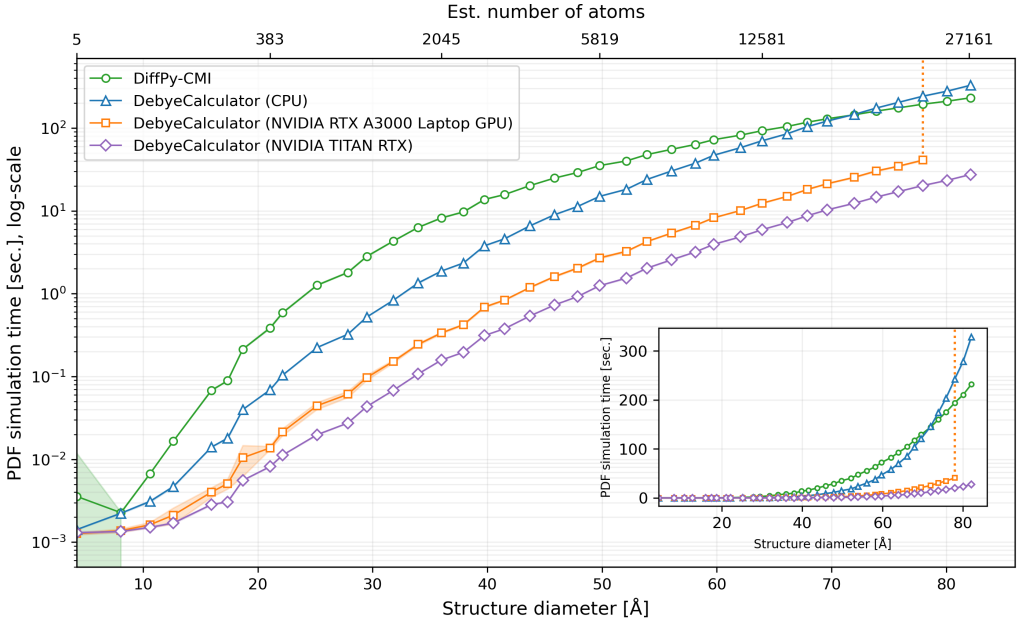

Figure 1: Computation-time comparison of the G(r) calculation using our CPU- and GPU-implementations against DiffPy-CMI.(Juhás et al., 2015) For the CPU-implementation, a batch size of 10,000 was chosen (x86-64 CPU with 6GB). Both the GPU implementations were run with a batch size of 10,000 (NVIDIA RTX A3000 Laptop GPU with 6GB of memory and NVIDIA Titan RTX GPU with 24GB of memory). The mean and standard deviation of the PDF simulation times are calculated from 10 runs. Note that, due to limited memory, the Laptop GPU was unable to handle structures larger than approx. 24,000 atoms.

Statement of need

Several software packages already exist for simulating the Debye scattering equation, including DiffPy-CMI,(Juhás et al., 2015) debyer,(Wojdyr, 2023) Debussy,(Cervellino et al., 2010, 2015) TOPAS,(Coelho, 2018) and BCL::SAXS.(Putnam et al., 2015) Our software distinguishes itself in several ways. Firstly, it is freely available and open-source licensed under the Apache License 2.0. Moreover, it is conveniently implemented as a 'pip' install package and has been integrated with Google Colab[https://github.com/FrederikLizakJohansen/DebyeCalculatorGPU/blob/main/quickstart/QuickStart.ipynb], allowing users to rapidly calculate the Debye

scattering equation without the need of local software installations. DebyeCalculator can be run on both CPU, GPU and through an interactive interface (see Figure 2), where users can calculate I(Q), S(Q), F(Q) and G(r) from structural models with one click.

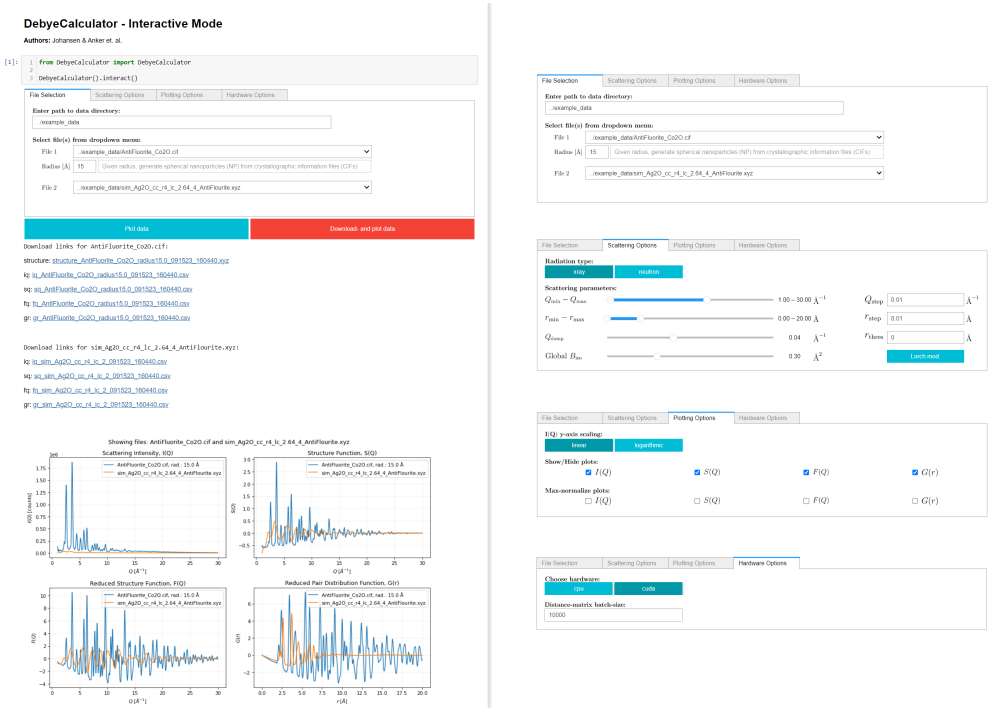

Figure 2: The interact mode of DebyeCalculator provides a one-click interface, where user can update parameters and visualise I(Q), S(Q), F(Q) and G(r). Additionally, the I(Q), S(Q), F(Q), G(r) and xyz-file can be downloaded, including metadata.

Acknowledgements

This work is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation Programme (grant agreement No. 804066).

Supporting Information

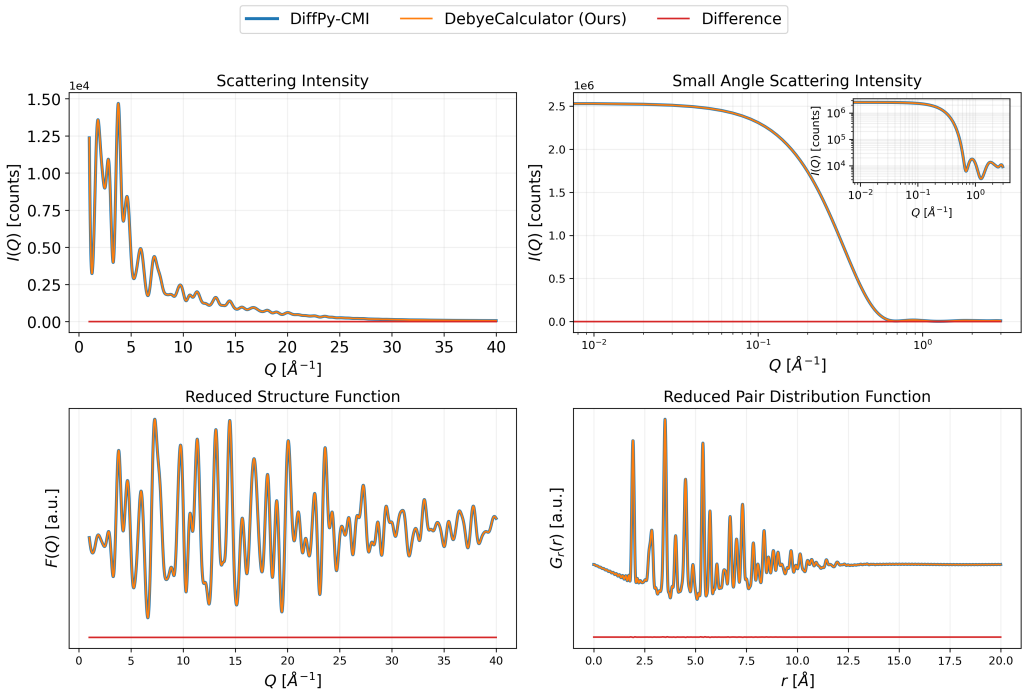

Figure 3: Comparison of the calculated I(Q), SAXS, F(Q) and G(r) of DebyeCalculator and DiffPy-CMI(Juhás et al., 2015) on a discrete, spherical cutout with 6 Å in radius from a $V_{0.985}Al_{0.015}O_2$ crystal.(Ghedira et al., 1977).

References

Cervellino, A., Frison, R., Bertolotti, F., & Guagliardi, A. (2015). DEBUSSY 2.0: The new release of a debye user system for nanocrystalline and/or disordered materials. *J. Appl. Crystallogr.*, *48*(6), 2026–2032.

Cervellino, A., Giannini, C., & Guagliardi, A. (2010). DEBUSSY: A debye user system for nanocrystalline materials. *J. Appl. Crystallogr.*, *43*(6), 1543–1547.

Coelho, A. A. (2018). TOPAS and TOPAS-academic: An optimization program integrating computer algebra and crystallographic objects written in c++. *J. Appl. Crystallogr.*, *51*(1), 210–218.

Debye, P. (1915). Zerstreuung von röntgenstrahlen. *Annalen Der Physik*, *351*(6), 809–823.

Egami, T., & Billinge, S. J. (2003). *Underneath the bragg peaks: Structural analysis of complex materials*. Elsevier.

Garland, M., Le Grand, S., Nickolls, J., Anderson, J., Hardwick, J., Morton, S., Phillips, E., Zhang, Y., & Volkov, V. (2008). Parallel computing experiences with CUDA. *IEEE Micro*, *28*(4), 13–27.

Ghedira, M., Vincent, H., Marezio, M., & Launay, J. C. (1977). Structural aspects of the metal-insulator transitions in v 0.985 al 0.015O2 . *J. Solid State Chem.*, *22*(4), 423–438. https://doi.org/https://doi.org/10.1016/0022-4596(77)90020-2

Juhás, P., Farrow, C., Yang, X., Knox, K., & Billinge, S. (2015). Complex modeling: A strategy and software program for combining multiple information sources to solve ill posed

structure and nanostructure inverse problems. *Acta Crystallogr. A*, *71*(6), 562–568.

NVIDIA. (2023). In *NVIDIA*. https://resources.nvidia.com/en-us-grace-cpu/grace-hopper-superchip

Putnam, D. K., Weiner, B. E., Woetzel, N., Lowe Jr, E. W., & Meiler, J. (2015). BCL:: SAXS: GPU accelerated d ebye method for computation of small angle x-ray scattering profiles. *Proteins: Struct., Funct., Genet.*, *83*(8), 1500–1512.

Scardi, P., Billinge, S. J., Neder, R., & Cervellino, A. (2016). Celebrating 100 years of the debye scattering equation. In *Acta Crystallogr. A* (No. 6; Vol. 72, pp. 589–590). International Union of Crystallography.

Schaller, R. R. (1997). Moore's law: Past, present and future. *IEEE Spectrum*, *34*(6), 52–59.

Waasmaier, D., & Kirfel, A. (1995). New analytical scattering-factor functions for free atoms and ions. *Acta Crystallographica Section A*, *51*(3), 416–431. https://doi.org/10.1107/S0108767394013292

Wojdyr. (2023). Wojdyr/debyer: Debye's scattering equation and other analysis of atomistic models. In *GitHub*. https://github.com/wojdyr/debyer