The Journal of Open Source Software

A GPU-Accelerated Open-Source Python Package for Rapid Calculation of the Debye Scattering Equation: Applications in Small-Angle Scattering, Powder Scattering, and Total Scattering with Pair Distribution Function Analysis

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Published: unpublished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 P. Debye, is commonly used to calculate the scattering intensities considering the position of each atom in the structure:(Debye, 1915; Scardi et al., 2016)

$$I(Q) = \sum_{i=1}^{N}\sum_{j=1}^{N} f_i(Q) f_j(Q) \frac{\sin(Qr_{ij})}{Qr_{ij}} \tag{1}$$

In this equation, Q is the scattering vector, r_{ij} is the distance between atom-pair, i and j, and f is the atomic scattering factor. 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) and small-angle scattering (SAS).(Scardi et al., 2016) Although the Debye scattering equation is extremely versatile, its applicability has been limited by the double sum of the atoms in the structure which makes the equation computationally expensive to calculate. With the advancement in computing technology,(Schaller, 1997) new horizons have opened up for applying the Debye scattering equation to larger materials. Modern central processing Units (CPUs), ranging from tenths to hundreds of cores, offer an opportunity to parallelise the computation, significantly enhancing computational efficiency. This parallel architecture allows for the distribution of the double sum calculations across multiple cores. Graphics processing units (GPUs) further expand computational possibilities, consisting of hundreds or even thousands of smaller, more efficient cores designed for parallel processing.(Garland et al., 2008) Unlike traditional CPUs, GPUs are ideally suited for calculations like the Debye scattering equation, where many computations can be performed simultaneously. By leveraging GPU acceleration, computational speeds that are orders of magnitude faster than even the most advanced multi-core CPUs are obtained. We introduce a GPU-accelerated open-source Python package for rapid calculation of the scattering intensity from a xyz-file using the Debye scattering equation. The xyz-file format describes the atomic structure with the atomic identify and its xyz-coordinates and is commonly used in materials chemistry. 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 on the CPU, while being open-source and easily assessable from our GitHub (https://github.com/FrederikLizakJohansen/DebyeCalculatorGPU/tree/main).

The DebyeCalculator, illustrated in the pseudocode that follows, begins with an initialisation function that sets various parameters. These parameters are either user-defined or set 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, atomic form factor coefficients are loaded, tailoring the form factor calculation to the chosen 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 G(r) inherently involves the calculations for I(q), S(q), and F(q). If the atomic structure has not been loaded, the DebyeCalculator loads the structure and computes the atomic form factors. 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 damping and the Lorch function, are applied before computing the G(r). Users have the flexibility to retain the results on the GPU or transfer them to the CPU. It is worth noting that the majority of the computational expense arises from the double sum calculation inherent in the Debye scattering equation. In order to take full advantage of parallel computing, we introduce a batch size parameter which determines the number of calculations processed simultaneously. Larger batch sizes generally lead to faster computation times as they can exploit the parallel nature of GPUs more effectively. However, it's essential to note that larger batch sizes consume more RAM, thereby necessitating better hardware. Consequently, users with more substantial GPU memory can accommodate even larger batch sizes and achieve even greater computation speeds.

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

  FUNCTION gr(structure, _keep_on_device):
      - IF structure is not already loaded THEN                         | XX   |
          - Load atomic structure from given path                       | XX   |
          - Calculate atomic formfactors                                | XX   |
      END IF
      - Calculate scattering intensity I(Q) (Debye scattering equation) | XX   |
      - Compute structure factor S(Q) based on I(Q)                     | XX   |
      - Calculate F(Q) based on q-values and S(Q)                       | XX   |
      - Apply modifications if necessary (like damping and Lorch)       | XX   |
      - Calculate pair distribution function G(r) based on F(Q)         | XX   |
      - Return G(r) either on the GPU-device or moved to CPU            | XX   |
```

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 patterns computations. Here, our implementation obtains the same scattering patterns as DiffPy-CMI (Supporting Information), while being about three times faster on CPU (Figure 1). Both calculations are run on a LLL CPU with a 004 batch size. Shifting our calculations to 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 with the CPU. It is important to note the overhead associated with initiating GPU calculations. For atomic structures with fewer than 000 atoms, this overhead

results in our GPU computations being slower than DiffPy-CMI and our CPU implementation. Once the atomic structure size exceeds this 000-atom threshold, we observe a speed-up using a blabla GPU and a batch size of 004. Specifically, 001 atoms onwards, the performance gain is on the order of 002 times. The choice of GPU hardware has a substantial influence on this speed advantage. As demonstrated in Figure 1, using a KKK GPU, which offers XXX GB of RAM enabling a batch size of 004, the speed benefits become even more evident. Beyond the same 000-atom threshold, the KKK GPU delivers a performance that is two orders of magnitude faster, underscoring the significant role of the hardware. With the advancements of GPUs like NVIDIA's Grace Hopper Superchip(n.d.-a), which boasts 576GB of fast-access GPU memory, there is potential for DebyeCalculator to achieve even greater speeds in the future.

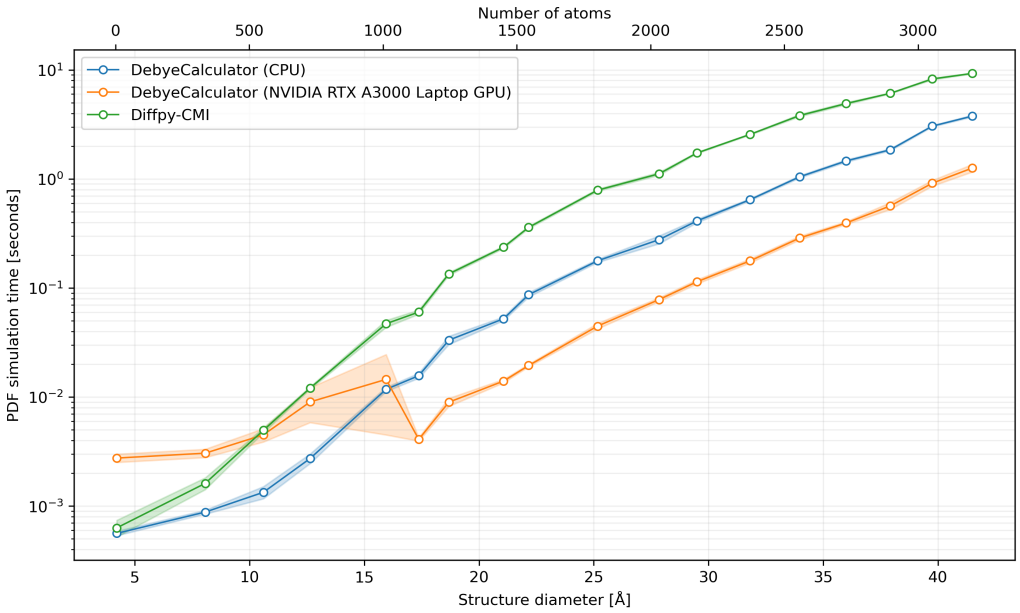

Figure 1: Computation-time comparison of the G(r) calculation using our CPU- and GPU-implementation against DiffPy-CMI.(Juhás et al., 2015) For the CPU-implementation, a batch size of XXX was chosen (blabla CPU with XX GB). Conversely, the GPU implementation was run with a batch size of XXX (NVIDIA RTX A3000 Laptop GPU with 6 GB).

Statement of need

Several software packages already exist for simulating the Debye scattering equation, including DiffPy-CMI,(Juhás et al., 2015) debyer,(Wojdyr, n.d.) 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. At the same time, our software is fast, and GPU accelerated. Crucially, our software is optimised for speed and outputs both I(Q), S(Q), F(Q) and G(r).

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

The Journal of Open Source Software

agreement No. 804066).

Raghav, Kirsten: Please add ack.

Supporting Information

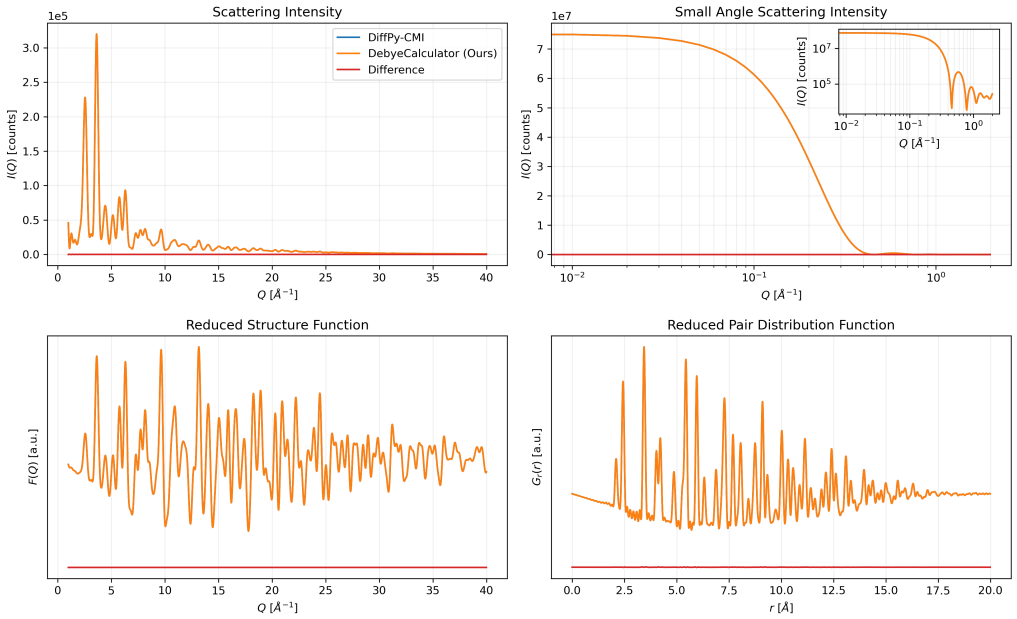

Figure 2: Comparison of the calculated I(Q), SAXS, F(Q) and G(r) of DebyeCalculator and DiffPy-CMI(Juhás et al., 2015) on a synthetic crystallographic information file describing a CoO_2 antifluorite crystal structure.(n.d.-b) Note that the scattering pattern simulated with DiffPy-CMI is hidden underneath the scattering pattern simulated with DebyeCalculator.

References

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

(n.d.-b). https://github.com/FrederikLizakJohansen/DebyeCalculatorGPU/blob/main/example_data/AntiFluorite_Co2O.cif

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. *Journal of Applied Crystallography*, *48*(6), 2026–2032.

Cervellino, A., Giannini, C., & Guagliardi, A. (2010). DEBUSSY: A debye user system for nanocrystalline materials. *Journal of Applied Crystallography*, *43*(6), 1543–1547.

Coelho, A. A. (2018). TOPAS and TOPAS-academic: An optimization program integrating computer algebra and crystallographic objects written in c++. *Journal of Applied Crystallography*, *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.

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 Crystallographica Section A: Foundations and Advances*, *71*(6), 562–568.

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: Structure, Function, and Bioinformatics*, *83*(8), 1500–1512.

Scardi, P., Billinge, S. J., Neder, R., & Cervellino, A. (2016). Celebrating 100 years of the debye scattering equation. In *Acta Crystallographica Section A: Foundations and Advances* (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.

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