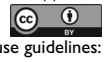
Research Article

Small angle neutron scattering in McStas: Optimization for high-throughput virtual experiments

Journal of Neutron Research
2024, Vol. 26(4) 173–185



DOI: 10.1177/10238166251313936
journals.sagepub.com/home/jne

José Ignacio Robledo[1], Klaus Lieutenant[1] and Peter Willendrup[2,3]

Abstract

In this work, we present the development of small-angle scattering components in McStas that describe the neutron interaction with 70 different form and structure factors. We describe the considerations taken into account for the generation of these components, such as the incorporation of polydispersity and orientational distribution effects in the Monte Carlo simulation. These models can be parallelized by means of multi-core simulations and graphical processing units. The acceleration schemes for the aforementioned models are benchmarked, and the resulting performance is presented. This allows the estimation of computation times in high-throughput virtual experiments. The presented work enables the generation of large datasets of virtual experiments that can be explored and used by machine learning algorithms.

Keywords

virtual experiments, small-angle neutron scattering, acceleration, graphical processing unit, machine learning

Received: 14 October 2024; accepted: 03 January 2025

1 Introduction

McStas[1,2] is an open-source package for Monte Carlo (MC) neutron ray-tracing simulations widely used by the neutron scattering community for instrument design in neutron facilities and also for educational purposes. A recent trend in the use of MC simulation software, inspired by increasing computing power, is the generation of large virtual experimental datasets. These datasets can then be explored using machine learning algorithms to learn underlying structures that may provide insight into novel data analysis techniques and instrument design optimization.[3] If the system is anisotropic, then the expression for the small-angle scattering (SAS) model now depends on the orientation of the particle relative to the incident beam, and therefore a shape-specific modulation of the scattering pattern as a function of the scattering vector $\vec{\mathbf{Q}}$ is present and should be adequately described.[4]

Large-scale datasets can be constructed by varying simulation parameters in virtual experiments of neutron instruments. The information that can be extracted from such datasets depends greatly on the possibility that the MC software grants in exploring the simulation's hyper-parameter space. This space consists mainly of parameters related to the instrument configuration in a given MC simulation, as well as parameters that describe the neutron interaction with the sample. In McStas, the complexity of the simulation's parameter space is determined by the available components that describe the instrument and sample in a virtual experiment. This work aims to expand the latter by introducing 70 new SAS model components that can be used in McStas to describe a sample in a virtual neutron experiment.

[1]Jülich Centre for Neutron Science 2, Forschungszentrum Jülich, Jülich, Germany
[2]Data Management and Software Centre (DMSC), European Spallation Source, Kongens Lyngby, Denmark
[3]Physics Department, Technical University of Denmark, Kongens Lyngby, Denmark

Corresponding author:
José Ignacio Robledo, Jülich Centre for Neutron Science 2, Forschungszentrum Jülich, Wilhelm-Johnen-Straße, 52428, Jülich, Germany.
Email: j.robledo@fz-juelich.de

In a first approximation and for simple problems where only first-order interactions and no secondary productions are taken into account,[5] MC neutron ray-tracing simulations are embarrassingly parallel.[6] That is, each neutron generated from the source distribution by means of MC choices is independent of the previously generated neutrons, and therefore the task of generating neutrons can be parallelized. The propagation of the neutrons in a beamline and their interaction with instrument components can also be parallelized because they are also independent. If only single scattering events are considered, the inclusion of a form factor or structure factor model for the sample description does not change this characteristic. Therefore, large dataset generation can have a significant speed-up if parallelization strategies are used in the MC neutron virtual experiments that generate the data.

This work describes the inclusion of 70 SAS models in McStas, which are optimized for parallel execution. A benchmark of the computation time of these models is also presented to emphasize and give estimates of the computing times necessary to build up big datasets.

2 Methods

2.1 Form factor and structure factor models

In scattering theory, the form factor $P(\vec{\mathbf{Q}})$ of a given model describes the dependence of the intensity of the scattering that is due solely to the shape of the scattering object. In SAS, the information due to the sample composition is encoded in the excess of scattering length density (SLD), $\Delta\rho$, and the modulation of the intensity due to the relative positions of scattering objects is described by the structure factor $S(\vec{\mathbf{Q}})$.

In the most general case of scattering from a scattering object of the model m, if we define the scattering amplitude as[7]

$$A_m(\vec{\mathbf{Q}}) = \int_{V_m} \Delta\rho_m(\vec{\mathbf{r}}) \exp\left[-\mathrm{i}\vec{\mathbf{Q}}\cdot\vec{\mathbf{r}}\right] \mathrm{d}V, \tag{1}$$

with V_m the volume of the scattering particle, then the measured intensity as a function of the scattering vector $\vec{\mathbf{Q}}$ can be written as

$$I_m(\vec{\mathbf{Q}}) = \frac{\Phi}{V_m} A_m(\vec{\mathbf{Q}}) A_m^*(\vec{\mathbf{Q}}), \tag{2}$$

where Φ is a scale factor. In a definition of the form factor $P(\vec{\mathbf{Q}})$ as the contribution due solely to the shape of the scattering object, since $\Delta\rho$ depends on the composition and not on the shape of the scatterers, it becomes necessary to separate it from the modulus square of the scattering amplitude $A^*(\vec{\mathbf{Q}})A(\vec{\mathbf{Q}})$. In most cases, this is straight-forward, such as in an elastic, isotropic scattering experiment of a dilute mono-disperse system, where the expression reduces to

$$I_m(Q) = n\Delta\rho^2 V_m^2 P_m(Q) S(Q), \tag{3}$$

with n the number density of particles. Since the intensity is related to the modulus squared of the amplitude, then the quantity $\Delta\rho^2$ appears often in the literature and is referred to as the contrast. We can define the SAS model of a sample including the $\Delta\rho$ dependency and also normalizing it by the volume of the particle so as to easily compare the scattering intensity measured in experiments, that is,

$$M_m(\vec{\mathbf{Q}}) = \frac{1}{V_m} A_m^*(\vec{\mathbf{Q}}) A_m(\vec{\mathbf{Q}}). \tag{4}$$

Given that $\Delta\rho$ is present in this expression through $A(\vec{\mathbf{Q}})$, the SAS model can be interpreted as describing the coherent scattering amplitudes arising from regions of excess SLD.[8] Finally, each model m considered in this work describes the shape characteristics of the scattering object through the definition of a set of model parameters $\{\theta_i | i = 1, \ldots, n_m\}$, therefore the notation $M_m(\vec{\mathbf{Q}}|\theta_1, \ldots, \theta_{n_m})$ will be used to explicitly refer to $M_m(\vec{\mathbf{Q}})$ when the complete model is defined.

A total of 70 SAS models have been developed as independent components in McStas[1] and are available since version 3.4 onward. Tables 1 and 2 present a summary of the models, classified into seven similarity groups labeled A to G. The components have also been imported to the X-ray counterpart McXtrace[9] as they describe the SAS characteristics that depend on the sample and not on the incident particle. Each model is described by two functions written in C language: one defining the scattering amplitude $A_m(\vec{\mathbf{Q}}|\theta_1, \ldots, \theta_{n_m})$ and another the corresponding particle volume V_m. Special care is taken in the case of anisotropic models to orient particles relative to the incident beam by means of rotation matrices. The

Table 1. Anisotropic small-angle scattering models included in McStas, inherited from SasView.

	A. Cylinders	B. Ellipsoids	D. Paracrystals	E. Parallelepiped	F. Spheres
1	Barbell	Core shell ellipsoid	Body centered cubic (BCC)	Core shell parallelepiped	Superball
2	Capped cylinder	Ellipsoid	Face center cubic (FCC)	Hollow rectangular prism	
3	Core shell bicelle	Triaxial ellipsoid	Simple cubic (SC)	Parallelepiped	
4	Core shell bicelle elliptical			Rectangular prism	
5	Core shell bicelle elliptical belt rough				
6	Core shell cylinder				
7	Cylinder				
8	Elliptical cylinder				
9	Hollow cylinder				
10	Stacked disks				

Note. Models are divided into seven groups, identified with letters from A to G. For a complete description of each model, visit the McStas component documentation and the SasView user documentation (https://www.sasview.org/docs/user/qtgui/perspectives/Fitting/models/index.html).

analytical model descriptions were obtained from the SasView[10] software package, which is widely used for SAS data analysis. These analytical models are well-known and tested within the SAS community. Additionally, a Python script has been included in the McStas repository to automatically extract the analytical functions from SasView and create the McStas components for future updates of available SAS models.

A total of 21 models exhibit anisotropic analytical models, which are indicated in Table 1. In an isotropic scattering model, the measured intensity I is proportional to the square of the scattering amplitude A, which is solely a function of the modulus of the scattering vector $\vec{\mathbf{Q}}$, that is,

$$I(\vec{\mathbf{Q}}) \propto A_m(\vec{\mathbf{Q}}|\theta_1, \ldots, \theta_{n_m})^2 = A_m^{\text{iso}}(Q|\theta_1, \ldots, \theta_{n_m})^2, \tag{5}$$

where θ_i are the n_m parameters that define the scattering model m. This is the case for all 49 models that are in Table 2. In contrast, in the anisotropic cases, the measured intensity depends on the relative orientation of the scattering objects with respect to the scattering vector. If the scattering object has no symmetry axis, then the scattering amplitude can be written as

$$A_m(\vec{\mathbf{Q}}|\theta_1, \ldots, \theta_{n_m}) = A_m^{\text{aniso}}(Q_A, Q_B, Q_C|\theta_1, \ldots, \theta_{n_m}), \tag{6}$$

where Q_A, Q_B, and Q_C are the projections of $\vec{\mathbf{Q}}$ on the A, B, and C axes of the scattering object. The orientation of the scattering objects can be defined relative to the MC simulation global coordinate frame xyz (defining the incoming neutron or X-ray beam) by defining three angles θ, ϕ, Ψ. In this definition, θ and ϕ define the orientation of the C-axis of the particle, and the angle Ψ defines the rotation about the C-axis. The 12 models from Table 1 that belong to this type are: some cylinder models (Group A: Core shell bicelle elliptical, Core shell bicelle elliptical belt rough, and Core shell parallelepiped), some ellipsoids (Group B: Elliptical cylinder and Triaxial ellipsoid), all paracrystals (Group D: Body centered cubic paracrystal, Face center cubic paracrystal, and Simple cubic paracrystal), some parallelepipeds (Group E: Hollow rectangular prism, Parallelepiped, and Rectangular prism), and a particular type of sphere model which with a parameter can be transformed into a parallelepiped (Group F: Superball).

If the form of the scattering object has one symmetry axis, suppose C, then the scattering amplitude can be calculated by having the projection of the scattering vector on that axis and the projection of the scattering vector onto the plane AB, that is,

$$A_m(\vec{\mathbf{Q}}|\theta_1, \ldots, \theta_{n_m}) = A_m^{\text{aniso}}(Q_{AB}, Q_C|\theta_1, \ldots, \theta_{n_m}). \tag{7}$$

There are nine such cases in the developed models: most cylinders (Group A: Barbell, Capped cylinder, Core shell bicelle, Core shell cylinder, Cylinder, Hollow cylinder, and Stacked disks) and ellipsoids (Group B: Ellipsoid and Core shell ellipsoid).

To test the developed models, we simulated the KWS-1 small-angle neutron scattering (SANS) beamline at FRM-II in McStas. This simulation consisted of a monochromatic source, guides and slits to propagate the neutron beam until the sample with defined divergence, one of the SANS modules listed in Tables 1 and 2, and a 2D position sensitive detector at a defined distance from the sample. For an accurate description of the McStas instrument, the simulation description can be found in the supplementary material of Robledo et al.[3]

Table 2. Isotropic small-angle scattering models included in McStas, inherited from SasView.

	A. Cylinders	C. Lamellae	E. Parallelepiped	F. Spheres	G. Shape-Indep.
1	Flexible cylinder	Head and tail groups (HG)	Hollow rectangular prism with thin walls	Adsorbed layers	Broad peak
2	Flexible cylinder elliptical	Caille structure factor (stack caille)		Binary hard spheres	Correlation length
3	Pearl necklace	HG + Caille structure factor (HG stack caille)		Core shell sphere	Debye Anderson Brumberger (DAB)
4	Pringle	with paracrystal structure factor (Stack paracrystal)		Fuzzy sphere	Fractal
5				Linear pearls	Fractal core shell
6				Multilayer vesicle	Gauss Lorentz Gel
7				Polymer micelle	Gaussian peak
8				Raspberry	Gel fit
9				Sphere	Guinier
10				Vesicle	Guinier porod
11				Hard sphere	Line
12				Hayter mean squared approximation (Hayter MSA)	Lorentz
13				Sticky hard sphere	Mass fractal
14					Mass surface fractal
15					Mono gauss coil
16					Peak Lorentz
17					Polymer excluded volume
18					Poly gauss coil
19					Porod
20					Power law
21					Spinoidal
22					Square well
23					Star polymer
24					Surface fractal
25					Teubner strey
26					Two Lorentzian
27					Two power law

Note. Models are divided into seven groups, identified with letters from A to G.

2.2 Polydispersity and orientational distributions

The components presented in this work offer the possibility of defining distributions for some model parameters in which this can be relevant. This includes polydispersity in all distances and radii parameters and orientational distributions in θ, ϕ, and Ψ orientation angles (described in the previous section) of anisotropic models. These features are included by considering a different size and orientation of the scattering object for each scattering event, which is randomly chosen from defined distributions.

For the moment, the polydispersity of a parameter r_a is incorporated through the random sampling of its value at each neutron interaction from a Gaussian distribution, that is,

$$r_a \sim N(r_a, \sigma_a^2), \tag{8}$$

with r_a and $\Delta r = \sigma_r / r$ are input parameters of the simulation. Radii and distances are always checked to be positively defined, else a new iteration of the random choice is performed.

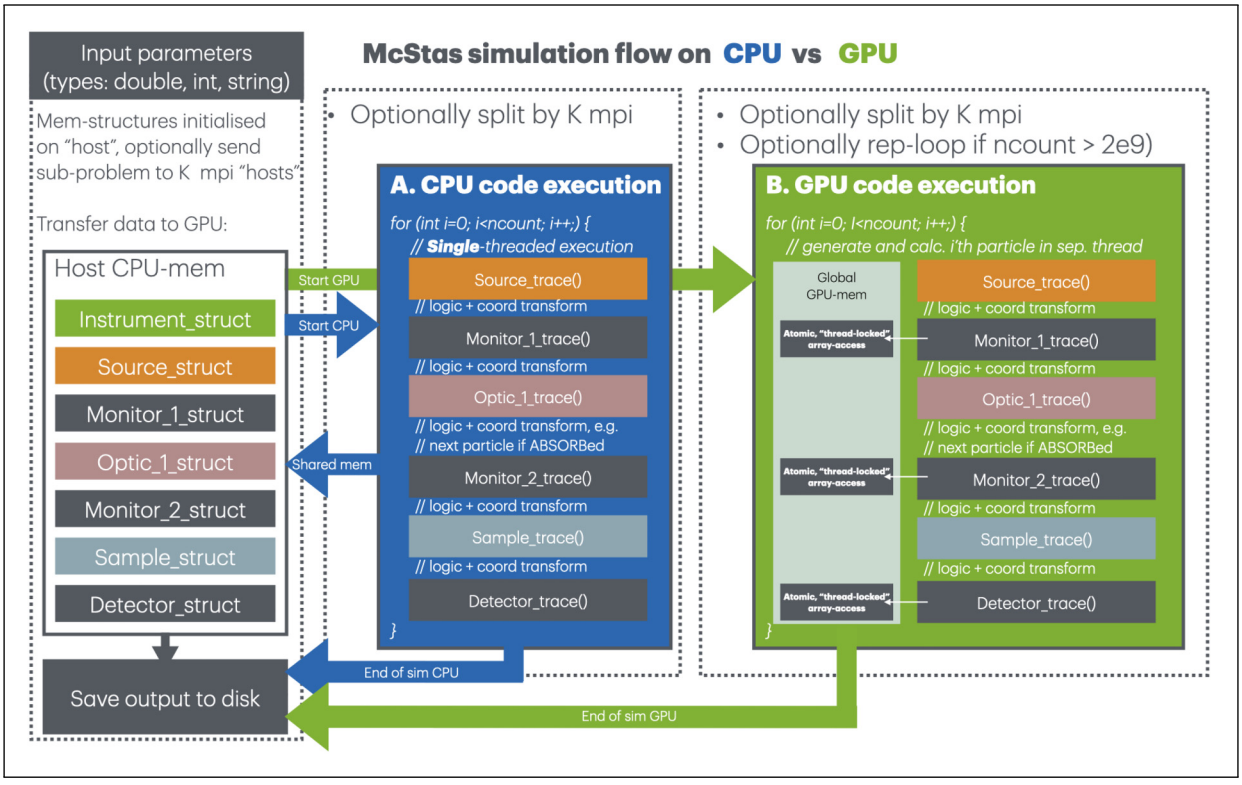


Figure 1. Schematic of a simulation flow in McStas, given a simulation statistic of $ncount$ neutron rays. In the CPU execution setting (dark blue), the neutron rays are treated one at a time in a single thread. In the MPI approach, the CPU execution is split into K cores that can run independently. In the GPU setting (bright green), the neutron rays are executed in many parallel threads simultaneously, with thread-locked write access to global GPU memory. Both the CPU and GPU simulation flow may be *further* parallelized by means of MPI execution in a scatter–gather manner. CPU: central processing unit; GPU: graphical processing unit; MPI: message passing interface.

The orientational distributions are also defined on each neutron interaction by random sampling from uniform distributions of the angles θ, ϕ, and Ψ as follows:

$$\theta \sim U[0, \pi],$$
$$\phi \sim U[0, 2\pi],$$
$$\Psi \sim U[0, 2\pi].$$

Increasing the number of incident neutrons in the sample in a simulation improves the MC estimates of polydispersity and orientational distribution contributions to the scattering pattern measured at a detector.

2.3 Acceleration schemes

The developed components allow multi-thread parallelization by means of a message passing interface (MPI)[11] and can also be deployed to the graphical processing unit (GPU) for higher-order acceleration. This is managed internally in each component using OpenACC.[12]

Figure 1 illustrates the McStas simulation flow during execution in three settings: single-threaded CPU, multi-threaded MPI, and parallel GPU execution. Given that McStas neutron rays are treated in a fully independent manner, a simulation problem of `ncount` neutron rays is in principle of *embarrassingly parallel* nature. In the single-core simulation case (blue box without splitting), a single neutron is simulated at a time in a single CPU thread, with direct memory access to the host memory. In the case of MPI on N_{cores} we apply a *scatter–gather* approach where each of the N_{cores} treat $\frac{ncount}{N_{\text{cores}}}$ of the original problem, with a following merge and save of the collected data on the MPI master node. In the GPU implementation (green box), we instead run in a massively parallel execution where many neutron rays are treated simultaneously, each in their thread. To ensure numerical validity and agreement with the CPU implementation, thread-locked access to the *global*

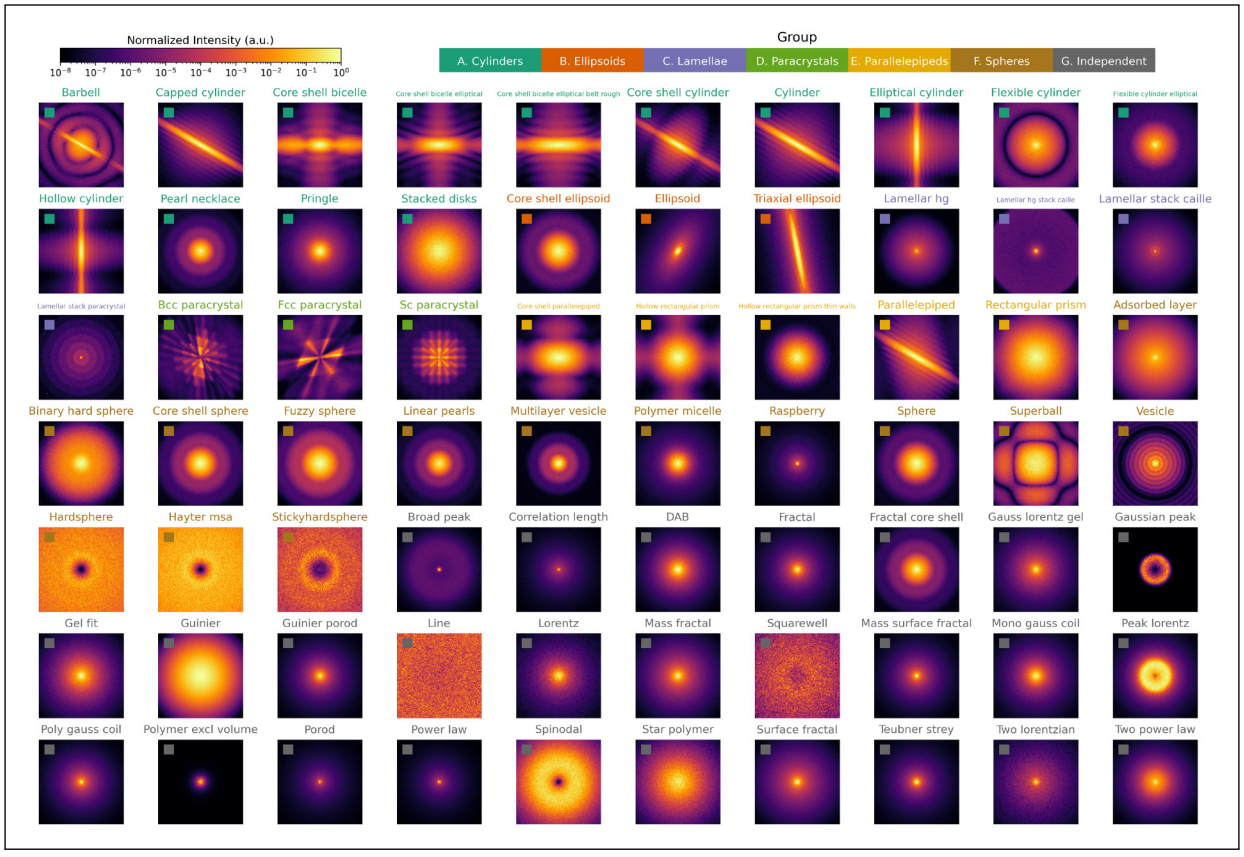


Figure 2. Normalized scattering pattern results from small-angle neutron scattering (SANS) simulations in McStas for all models developed in this work. Each image corresponds to one simulation of the same instrument configuration but changes in the sample description. All samples were defined by setting the default parameters for each SAS model and no polydispersity. For all anisotropic models, no orientation distribution was included, resulting in perfectly oriented scattering objects to emphasize anisotropic scattering features. Rectangles indicate the groups in Tables 1 and 2.

GPU memory was required and implemented via the OpenACC `atomic` clause. Transfers to and from the GPU happen at initialization time and simulation finalization, as the host CPU handles input/output of files (note also that a multi-GPU implementation was trivial: we simply scatter–gather the original problem such as in the standard MPI case, sending $\frac{\mathtt{ncount}}{N_{\text{cores}}}$ to each of N_{GPUs}).

To compare the acceleration capabilities, three different simulation schemes were tested:

(1) a single-core simulation, to be used as reference;
(2) a Multi-threading approach with $N_{\text{cores}} = 16$ cores using MPI; and
(3) a GPU approach with an NVIDIA A100.

For each one of these schemes, all 70 models (listed in Tables 1 and 2) were used for the comparison of the acceleration schemes. Each one was simulated in an independent virtual experiment with a different number of incident neutrons ($N = 10^5, 10^7, 10^8, 10^9, 10^{10}, 10^{11}$). The simulation run time was measured in every case.

3 Results

3.1 SAS models

The results of 70 simulations in which the instrument configuration was fixed and the sample description was changed in all of the models presented in this work are shown in Figure 2. Each scattering pattern corresponds to testing one of the SAS models M_m listed in Tables 1 and 2 with the default values for the model parameter definition of the component in McStas.

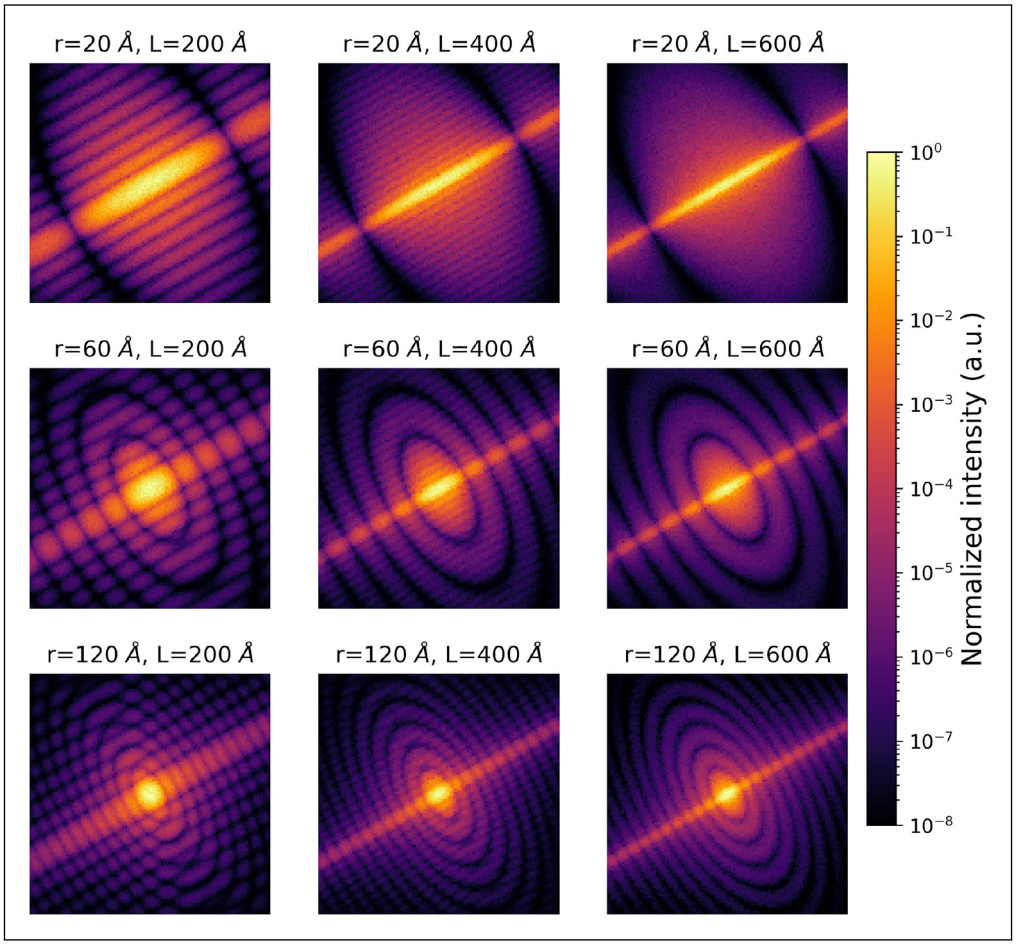


Figure 3. Example of intra-class variability for a cylinder form factor model with anisotropy. Each image corresponds to one simulation where the shape of the scattering objects is defined by the radius r and length L of the cylinders (written above). In all simulations, the cylinders were perfectly oriented at $\theta = \phi = 60°$ ($\Delta\theta = \Delta\phi = 0$) and were completely mono-disperse ($\Delta r = \Delta L = 0$).

These default values are directly obtained from SasView, and have been selected since they are good prior estimates for reasonable physical models (they fulfill constraints and are estimated toward user needs in real scenarios). The scattering patterns in Figure 2 show the variability in neutron intensity that can be present at the detector when selecting the SAS model M_m. This variability can be referred to as inter-class variability. Some models are clearly different than others, and some others look very similar under the parameter definition used to generate Figure 2. Nevertheless, intra-class variability can also provide very different scattering patterns for the same model M_m. To exemplify this, the parameter space of the anisotropic model *Cylinder* is varied drastically on different simulations and the resulting scattering patterns are shown in Figures 3 and 4. In this example, the model considers no structure factor (dilute regime) and only the form factor contributes to the anisotropic scattering pattern. To better understand these figures, the analytical expression of the cylinder form factor is revised. Following equation (4), the form factor of a cylinder can be obtained by defining the scattering amplitude as[13,14]

$$A_{\text{cylinder}}(Q_{AB}, Q_C|\rho, \rho_s, r, L) = 2(\Delta\rho)V \frac{\sin\left(\frac{1}{2}Q_{AB}L\right)}{\frac{1}{2}Q_{AB}L} \frac{J_1(Q_C r)}{Q_C r}, \tag{9}$$

where $J_1(x)$ is the first-order Bessel function, $\Delta\rho$ is the excess of SLD of the cylinder and the solvent, and r and L are the cylinder's radius and length, respectively. The volume is a function of the parameters and must be calculated, in this case for a cylinder $V_{\text{cylinder}} = \pi r^2 L$. In the case of the McStas simulation, the parameter space that can be varied is larger

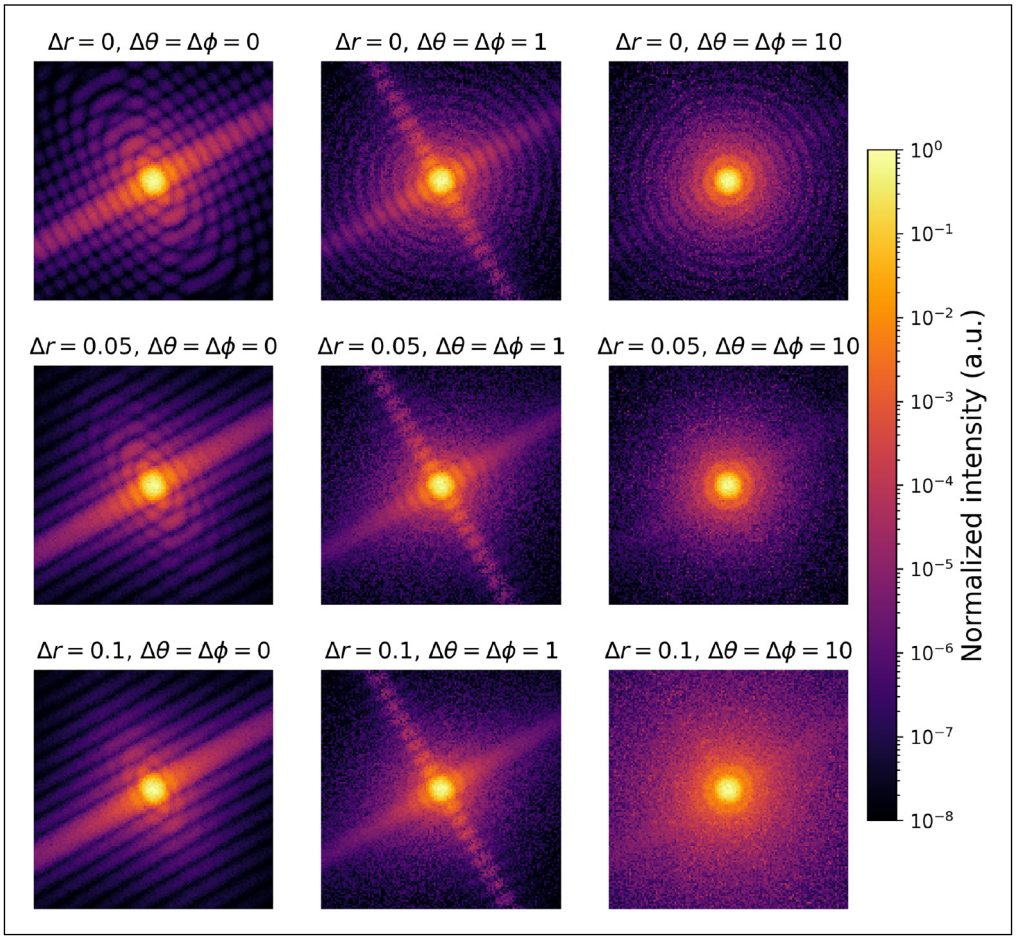


Figure 4. Example of intra-class variability due to polydispersity in radius and orientational distributions for a cylinder form factor model with anisotropy. Every image corresponds to a different simulation with varying parameters as described above the image, and common parameters mean radius $r = 120$ Å and cylinder length $L = 200$ Å.

than that of the form factor definition given, since the polydispersity and orientational distributions must be defined as explained in Section 2.2. In this case, the model parameter space is $\vec{\theta}_{\text{cylinder}} = \{\rho, \rho_s, r, L, \theta, \phi, \Delta r, \Delta L, \Delta\theta, \Delta\phi\}$, where ρ and ρ_s are the SLD of the cylinder and solvent respectively, θ and ϕ define the orientation angles with respect to the incident neutrons (and therefore the projections Q_{AB} and Q_C), and Δx is the corresponding relative variance parameter of parameter x that defines the variance of the polydispersity or orientational distribution. To simplify the interpretation of each parameter, the results presented in Figure 3 are obtained for fully oriented mono-disperse cylinders ($\Delta x = 0$ for $x = r, L, \theta, \phi$) and Figure 4 shows the effect of polydispersity and orientational distribution for fixed form factor model parameters.

As can be seen in both, Figures 3 and 4, the scattering patterns corresponding to a given model may be very different given that all parameters express themselves in different features on the 2D scattering pattern. The variation of model parameters r and L shown in Figure 3 exemplifies how the shape of the scattering objects can produce varying scattering patterns even though they arise from fully oriented cylinders. Adding variation on the orientation of the cylinders transforms an anisotropic scattering pattern toward an isotropic one (first row on Figure 4), while adding polydispersity to a shape parameter blurs the characteristic oscillations corresponding to it (evanescent ellipses on the first column in Figure 4). Given that in these simulations all the cylinders were of the same length L, the spacing of the diagonal line features resulting from oriented particles of a fixed length L do not vary unless the relative orientation between cylinders is lost.

Finally, we benchmark the cylindrical model with an experimental data curve of cylindrically shaped micelles measured at the SANS instrument KWS-1 located at FRM-II.[15] This dataset has been widely studied since it is part of the Lab-course material of the Jülich Centre for Neutron Science.[16] The sample consists of amphiphilic polymers POO_{10}–PEO_{10} in heavy

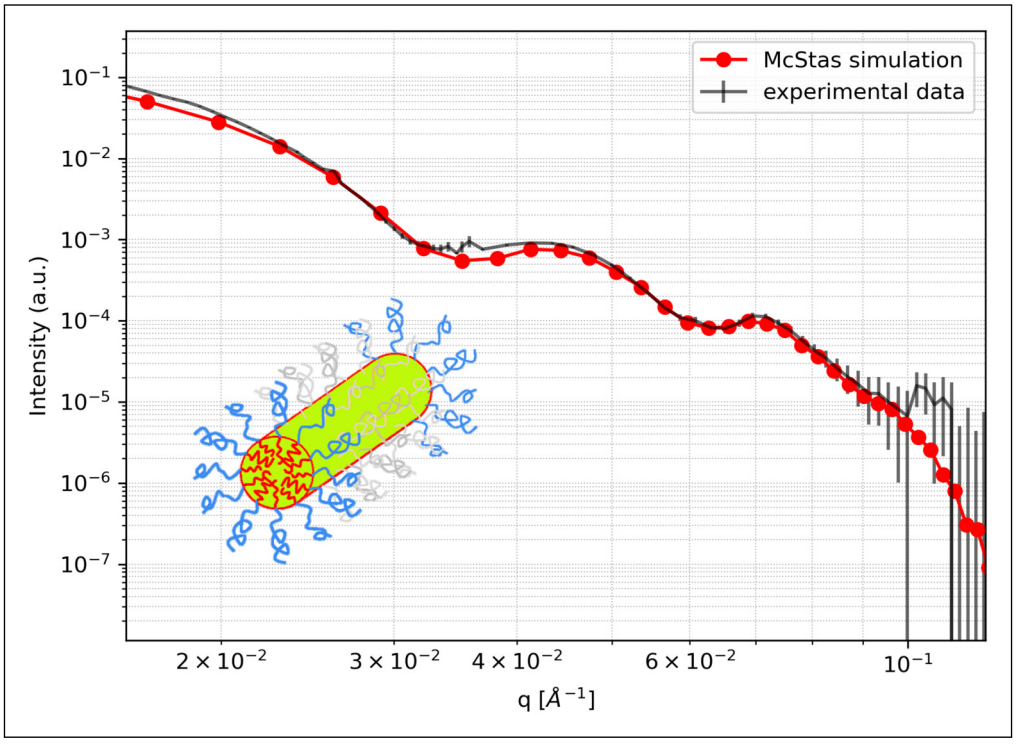


Figure 5. Comparison of small-angle scattering curves of a cylindrical model McStas simulation with experimental data measured at KWS-1, FRM-II. The inlay image depicts the amphiphilic polymers POO_{10}–PEO_{10} forming the scattering object, which by contrast matching can be considered of cylindrical shape.

water as shown in the inset of Figure 5. During the experiment, the contrast was matched so that only the core of the micelle was visible for neutrons. The measured data is plotted with error bars in Figure 5 (black curve). To test our cylindrical form factor model, the simulation of the McStas instrument of KWS-1 published in Robledo et al.[3] was used to describe the instrument, and the novel component of the cylindrical model (A.7 in Table 1) was selected to describe the sample. Given that the position of the lowest Q drop in the experimental data is $Q_{\min} = 0.032\,\text{Å}^{-1}$, and using the relation for cylindrical form factors of $R = 3.78/Q_{\min}$,[17] then the radius of the cylinder sample was set in the McStas simulation at 119 Å. For the cylinder length, the experimental data was fit using SasView and constraining the radius to the previously obtained value. From this fitting, a value of $L = 800\,\text{Å}$ and polydispersities of 5% for the radius and length parameters were used in the MC simulation. In addition, the orientation of the cylinders was set to random. The result of this McStas simulation can be seen in Figure 5 with red circles. The agreement between the curves is an indicator that it is now possible to simulate a large variety of types of samples by means of the McStas SAS models presented in this work.

3.2 Acceleration schemes

All models described in Tables 1 and 2 were tested in simulations with different numbers of incident neutrons N under the three acceleration schemes described in Section 2.3. Their corresponding run times were measured. Since a motivation for accelerating MC simulations is the ability to improve statistics by scaling the number of simulated neutrons N to improve MC estimates, the median value of the run time of all models as a function of N is shown in Figure 6. At a low number of particles N, the cost in time of threading may be higher than the gain in computation time of the actual parallelized run, resulting in a faster single-core simulation. In the case of the GPU approach, the tasks of uploading and downloading data are very time-consuming, resulting in the worst performance. Therefore, for a small number of particles ($N < 10^6$), choosing a single-core simulation is the best choice. At intermediate N, both the GPU and MPI approaches start to show advantages while they get filled with neutron trajectories, and the cost of distributing the tasks becomes smaller than the time gained by parallelization. We can conclude from Figure 6 that an MPI acceleration scheme is the optimal choice in the range $N \in [10^6, 10^8]$. For larger values of N ($N > 10^8$), optimizing simulations by including GPU parallelization gives the best performance among the three methods discussed, in this case being approximately three times faster than the

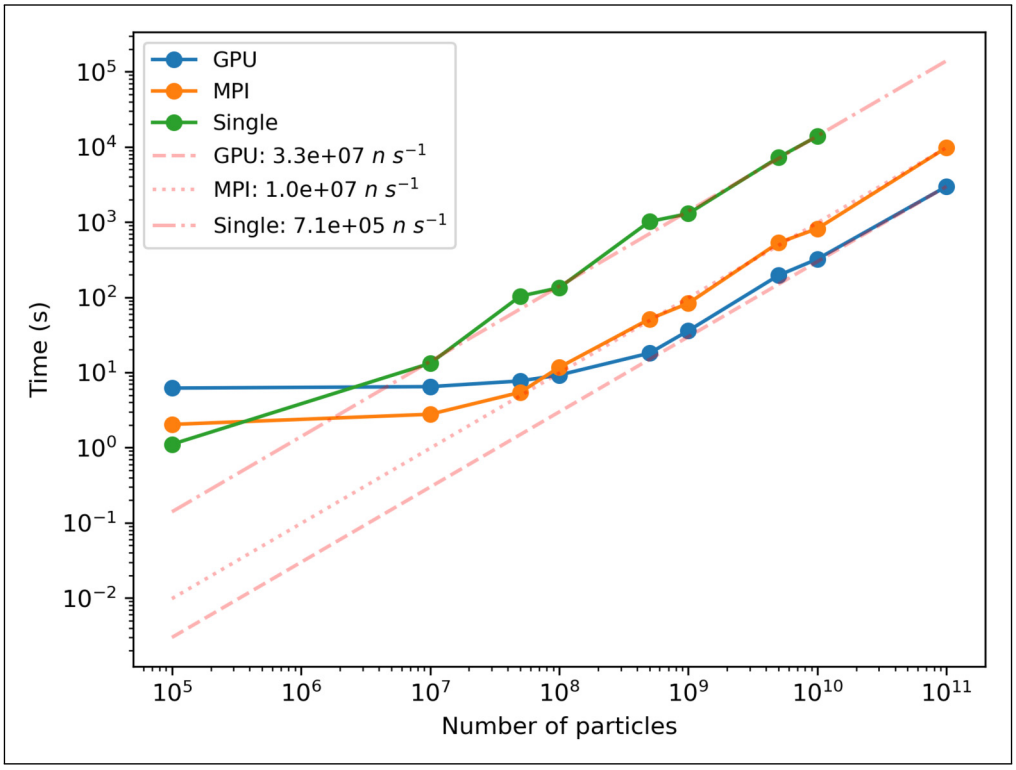


Figure 6. Median time of the models presented in this work as a function of the number of simulated neutrons, using the GPU, MPI, or single-core (single). The inverse of the slope, in neutrons per second, of the linear regression for high number of particles is given in the labels for each type of simulation. GPU: graphical processing unit; MPI: message passing interface.

16-core MPI approach. Scaling to larger amounts of cores or GPUs in a cluster will lead to increasing accelerations and the run times of these models can be estimated by means of the results presented in this work. The acceleration schemes presented here are those that can be found in a personal computer.

When either the GPU or all of the cores available get filled with trajectories, then the simulation time increase becomes linear with increasing N. The slope values of the linear fits presented in this figure are shown in the labels of Figure 6. Compared to a single-core simulation, the MPI approach with 16 cores results in a speed-up ratio ($t_{\text{single}}/t_{\text{mpi}}$) of 14.26. This can be interpreted as if each core results in a net factor of 0.89 of the calculation power of a single core. The deviation from a 16 times gain using 16 cores is given by the fact that the threading and summation of results are time-consuming. On the other hand, by using one single NVIDIA A100 (40 GB) GPU, the speed-up ratio is 46.87, which can be thought of as equivalent to using 52 cores in an MPI simulation. As mentioned before, the deviation from the linear tendency at a low number of simulated particles is due to the GPU having a capacity for more trajectories.

It is interesting to observe that the median times of the distributions cross somewhere between 5×10^7 and 5×10^8 neutrons. Also, the increase from 1×10^7 neutrons to 5×10^8 is almost costless in computation time for all the models when using the GPU. This is a consequence of the under-use of the full capacity of the GPU for low N. The asymptotic value at low N of about 5 seconds for the mean (among all models) runtime is an estimator of the time needed by the GPU card to upload and download the simulation data by means of OpenACC.

Figure 6 also shows that simulations accelerated by GPU (blue curve) take roughly the same time as those accelerated by MPI (orange curve) for $N = 1 \times 10^8$. We decided to study the simulation run time variation as a function of the model component and the acceleration scheme for this particular value of N. The results, presented in Figure 7, show that both optimization schemes presented are more convenient than the single-core sequential simulation. It can be seen that the multi-threading approach is more convenient than the GPU approach in the majority of the SAS models. However, in those computationally expensive models, such as Barbell, Capped cylinder, Hollow rectangular prism thin walls, Flexible cylinder elliptical, Hayter mean squared approximation, and Polymer micelle, the GPU calculation with OpenACC is faster than the multi-threading one with MPI. For higher values of N, all models run faster on GPU than on MPI. This

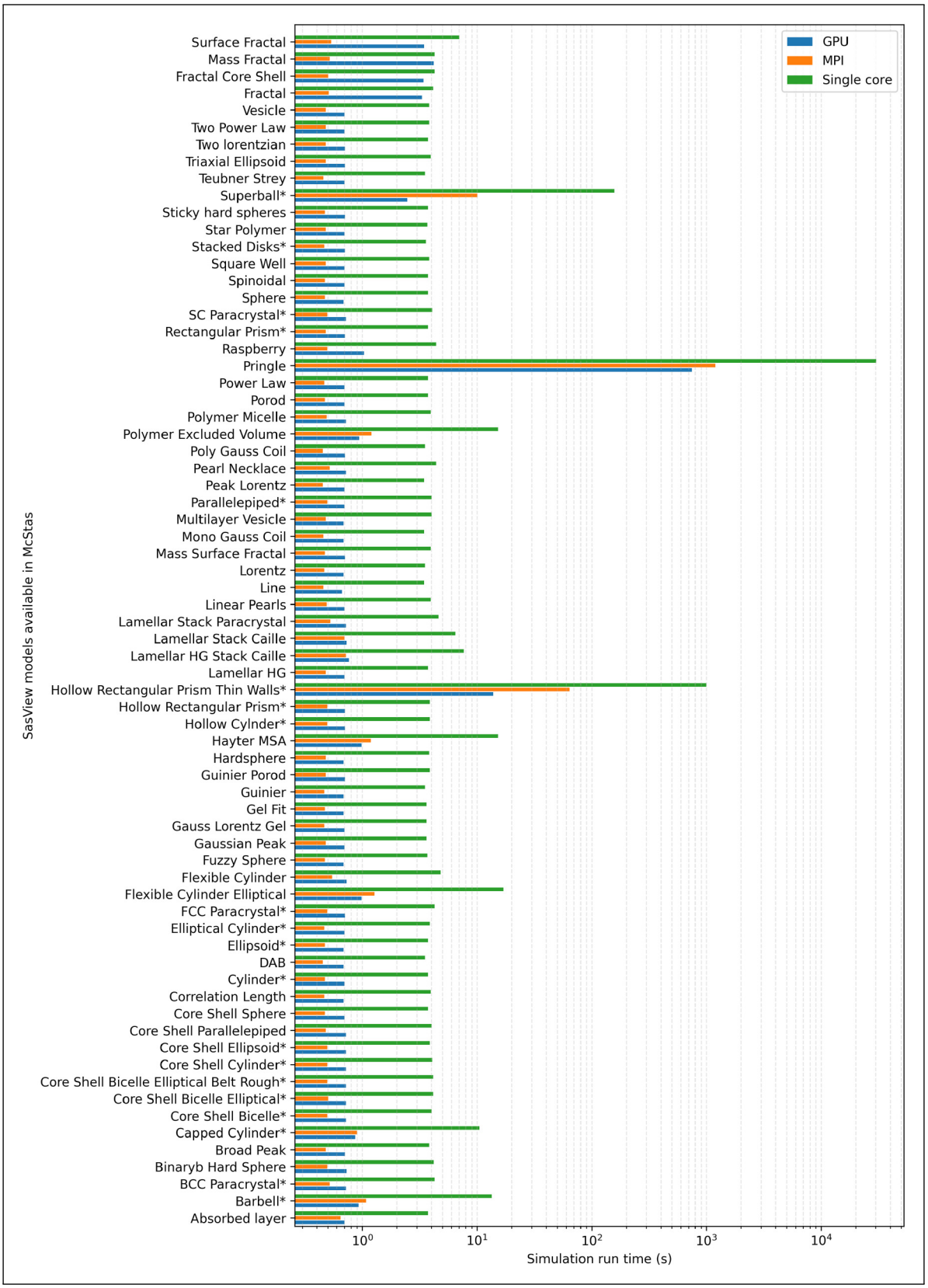


Figure 7. Simulation time of $N = 1 \times 10^8$ neutrons in McStas for each of the 70 analytical SANS models described in SasView (now available in McStas 3.4). Colors indicate the different speed-up methods employed: a simulation using the GPU NVIDIA A100 (40 GB) by means of OpenACC, MPI for multi-threading using OpenMPI, and Single core for sequential simulations using only one thread in a single CPU. The asterisk in name labels indicates the anisotropic models. GPU: graphical processing unit; CPU: central processing unit; MPI: message passing interface.

figure also provides a relative estimate of the complexity, and therefore of the execution time, of the components presented in this work and can be used as a reference when selecting the adequate model for a simulation.

4 Conclusion

This work describes the optimized SAS sample components that are readily available in McStas, which opens up the landscape for the possible small-angle neutron scattering virtual experiments that can be performed with the software. All the SAS components have the effects of polydispersity as well as orientational distribution included. Components can be used under acceleration schemes, according to the necessity. For single virtual experiments, where statistics is not very important, rather qualitative features, single-core simulations are adequate. When increasing the number of particles in a given simulation, multi-core and GPU parallelization of the particle traces becomes relevant. The GPU parallelization outperforms the multi-core simulation presented in this work when the number of neutrons in the simulation is higher than 10^8. This can be useful if attempting to observe low-probability scattering effects on simulations where neutrons perform very long trajectories. Of course, by increasing the number of threads of MPI with enough cores, or the number of GPUs used, the speed-up gains of each method can be exploited even stronger. But in this work, a reasonable value of threads as well as a number of GPUS that a current notebook can offer was chosen for comparison.

The parallelization of traces in MC particle ray-tracing algorithms can be deeply exploited in simulations with large hyper-parameter spaces to explore a vast region of this space and generate large datasets of MC simulations. Simulation datasets can help in data augmentation,[18] as well as to learn from them through machine learning and then to infer on small datasets of real experiments (which is the usual case in neutron scattering experiments).[19] With the recent advances in generative adversarial networks that allow mapping from a simulated data distribution to an experimental data distribution in a bijective manner,[20] this type of mechanism for dataset generation can become very useful.

Acknowledgements

We would like to acknowledge the Helmholtz Artificial Intelligence group by providing computational resources through HAICORE project 27114.

ORCID iDs

José Ignacio Robledo https://orcid.org/0000-0001-8784-2558
Klaus Lieutenant https://orcid.org/0000-0001-8278-9401
Peter Willendrup https://orcid.org/0000-0002-4238-9478

Statements and declarations

Funding

The authors disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement no. 101034266. This work benefited from the use of the SasView application, originally developed under NSF award DMR-0520547. SasView contains code developed with funding from the European Union's Horizon 2020 research and innovation programme under the SINE2020 project, grant agreement no. 654000.

Declaration of conflicting interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

1. Willendrup PK and Lefmann K. McStas (i): Introduction, use, and basic principles for ray-tracing simulations. *J Neutron Res* 2020; 22: 1–16.
2. Willendrup PK and Lefmann K. Mcstas (ii): An overview of components, their use, and advice for user contributions. *J Neutron Res* 2021; 23: 7–27.
3. Robledo JI, Frielinghaus H, Willendrup P, et al. Learning from virtual experiments to assist users of small angle neutron scattering in model selection. *Sci Rep* 2024; 14: 14996.
4. Rooks J, Gilbert PH, Porcar L, et al. Anisotropy factors in small-angle scattering for dilute rigid-rod suspensions. *J Appl Crystallogr* 2023; 56: 683–696.
5. Hoogenboom JE. Delft Nuclear Consultancy, IJsselzoom 2, 2902 LB Capelle aan den IJssel. Is Monte Carlo embarrassingly parallel? 2012, https://www.osti.gov/biblio/22105636

6. Foster I. *Designing and building parallel programs concepts and tools for parallel software engineering*. Reading, MA: Addison-Wesley, 1995.
7. Hamley I. *Small-angle scattering*. 1st ed. New Jersey: John Wiley & Sons Ltd, 2021.
8. nad Jan Ilavsky CMJ, Martel A, Hinrichs S, et al. Small-angle X-ray and neutron scattering. *Nat Rev Methods Primers* 2021; 1: Article no: 70 .
9. Knudsen EB, et al. Mcxtrace: A Monte Carlo software package for simulating x-ray optics, beamlines and experiments. *J Appl Crystallogr* 2013; 46: 679–696.
10. SasView, 2023, https://www.sasview.org
11. Message Passing Interface Forum. MPI: A message-passing interface standard version 4.1, 2023, https://www.mpi-forum.org/docs/mpi-4.1/mpi41-report.pdf
12. Farber R. *Parallel Programming with OpenACC*. 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016.
13. Guinier A, Fournet G, Walker CB, et al. *Small-angle scattering of X-rays*. New York: Wiley, 1955.
14. Kline S. SANS models at NIST resources, 1998, https://www.ncnr.nist.gov/resources/sansmodels/Cylinder.html
15. Feoktystov AV, Frielinghaus H, Di Z, et al. KWS-1 high-resolution small-angle neutron scattering instrument at JCNS: Current state. *J Appl Crystallogr* 2015; 48: 61–70.
16. Thomas B, Stephan F, Georg R, et al. *Neutron scattering – Experimental manuals*. Verlag Jülich: Forschungszentrum Jülich GmbH Zentralbibliothek, 2019.
17. Hammouda B. The SANS toolbox. NIST Center for Neutron Research, 2008, http://tinyurlcom/SANStoolbox
18. Oviedo F, Ren Z, Sun S, et al. Fast and interpretable classification of small x-ray diffraction datasets using data augmentation and deep neural networks. *npj Comput Mater* 2019; 5: 60.
19. Franke D, Jeffries CM and Svergun DI. Machine learning methods for x-ray scattering data analysis from biomacromolecular solutions. *Biophys J* 2018; 114: 2485–2492.
20. Anker AS, Butler KT, Le MD, et al. Using generative adversarial networks to match experimental and simulated inelastic neutron scattering data. *Digit Discov* 2023; 2: 578–590.