JOURNAL OF APPLIED CRYSTALLOGRAPHY

ISSN 1600-5767

SEB: a computational tool for symbolic derivation of the small-angle scattering from complex composite structures

Tobias W. J. Jarrett and Carsten Svaneborg*

University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark. *Correspondence e-mail: zqex@sdu.dk

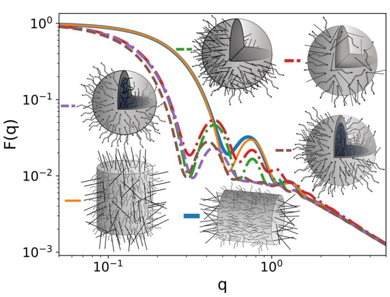
Received 29 November 2023
Accepted 21 February 2024

Edited by S. Disch, Universität Duisburg-Essen, Germany

Keywords: small-angle neutron scattering; SANS; small-angle X-ray scattering; SAXS; computational tools; data analysis.

Supporting information: this article has supporting information at journals.iucr.org/j

Analysis of small-angle scattering (SAS) data requires intensive modeling to infer and characterize the structures present in a sample. This iterative improvement of models is a time-consuming process. Presented here is *Scattering Equation Builder* (*SEB*), a C++ library that derives exact analytic expressions for the form factors of complex composite structures. The user writes a small program that specifies how the sub-units should be linked to form a composite structure and calls *SEB* to obtain an expression for the form factor. *SEB* supports *e.g.* Gaussian polymer chains and loops, thin rods and circles, solid spheres, spherical shells and cylinders, and many different options for how these can be linked together. The formalism behind *SEB* is presented and simple case studies are given, such as block copolymers with different types of linkage, as well as more complex examples, such as a random walk model of 100 linked sub-units, dendrimers, polymers and rods attached to the surfaces of geometric objects, and finally the scattering from a linear chain of five stars, where each star is built up of four diblock copolymers. These examples illustrate how *SEB* can be used to develop complex models and hence reduce the cost of analyzing SAS data.



1. Introduction

Small-angle scattering (SAS) is an ideal technique to characterize the size, shape and orientation of nanoscale structures in a sample (Guinier *et al.*, 1955; Feigin *et al.*, 1987). In order to infer the structures present in a sample, SAS scattering profiles are often analyzed by fitting models (Pedersen, 1997). Thus SAS data analysis is an iterative process where models for structures are proposed, their corresponding scattering profiles are mathematically derived and model scattering profiles are fitted to the experimental scattering profiles. If the fits are poor, the models have to be improved and the process is repeated until a good model has been developed. A ‘good’ model is one which provides an acceptable fit to the experimental data and is thus the most likely candidate for the structures present in the sample.

SAS scattering spectra contain information about the nanoscale structure but not the detailed atomic scale structure, and hence relatively simply geometric models are often used when analyzing SAS data. Fortunately, the scattering from a large number of models has already been derived [see *e.g.* Pedersen (1997)]. In the case where *e.g.* objects of similar shape are dispersed in a liquid, the problem of modeling the scattering from the sample can be split into (i) what the shapes of the objects are and (ii) what the spatial correlations of the objects are due to their mutual interactions (Pedersen, 1997). The first problem is described by the form factor while the

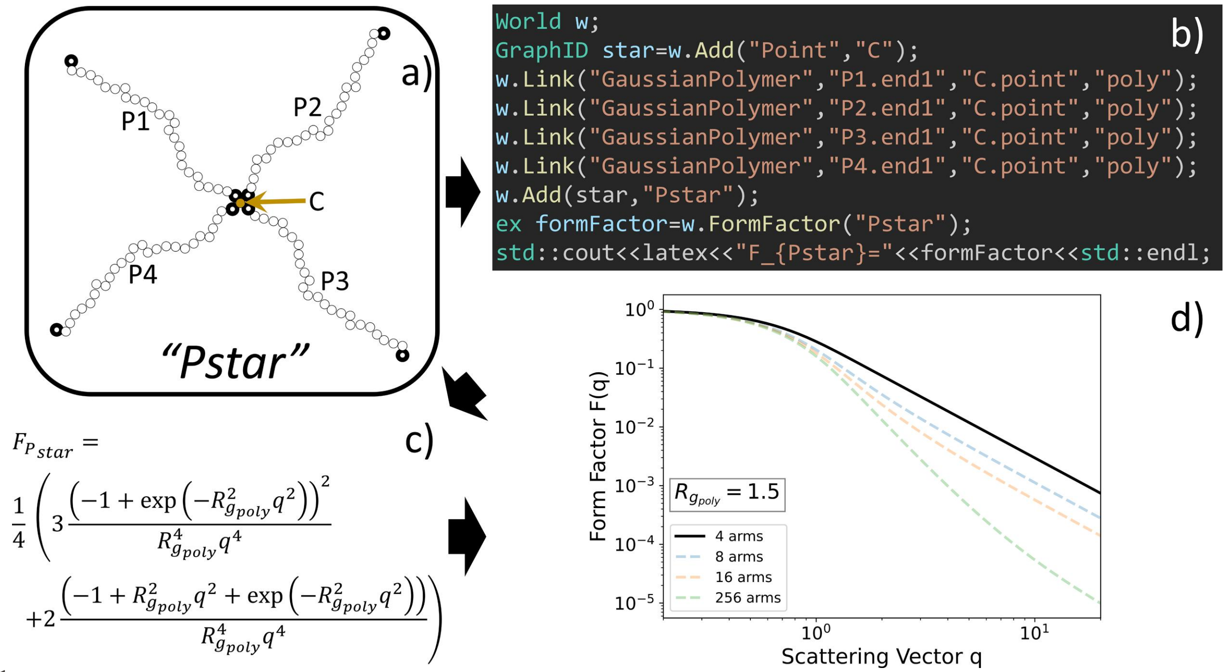


Figure 1
The *SEB* workflow: (*a*) defining a structure, (*b*) implementing the structure in *SEB*, (*c*) obtaining the analytic form factor equation, and (*d*) evaluating and plotting the form factor for given structural parameters.

latter part is described by the structure factor, and in dilute samples the scattering is dominated by the form factor.

Here we present *Scattering Equation Builder* (*SEB*), which is a C++ software library that analytically derives symbolic expressions for the form factor of composite structures built by linking an arbitrary number of sub-units together. A typical workflow is shown in Fig. 1. Starting from a model of a structure (here a four-armed star built up of polymers), a short C++ program is written to define this structure within *SEB*. It typically takes *SEB* less than a minute to derive analytically a symbolic expression for the form factor of the structure. What is outside the scope of *SEB* is code required for representing and fitting experimental data or providing a graphical user interface. Already numerous excellent software tools have been developed with the specific aim of fitting models to scattering data. A non-exhaustive list includes *e.g. ATSAS* (Konarev *et al.*, 2006; Manalastas-Cantos *et al.*, 2021), *CRYSOL* (Svergun *et al.*, 1995), *CRYSON* (Svergun *et al.*, 1998), *FoXS*/*MultiFoXS* (Schneidman-Duhovny *et al.*, 2016), *GENFIT* (Spinozzi *et al.*, 2014), *GenApp* (Perkins *et al.*, 2016), *IRENA* (Ilavsky & Jemian, 2009), *SASfit* (Breßler *et al.*, 2015; Kohlbrecher & Breßler, 2022), *SASview* (Doucet *et al.*, 2021), *Scatter* (Förster *et al.*, 2010) and *WillItFit* (Pedersen *et al.*, 2013).

Our aim with *SEB* is to provide a computationally efficient tool to derive form factor expressions for arbitrary complex branched structures. The expressions can be exported in a variety of formats, allowing them to be imported into C, C++ or Python programs for numerical evaluation, into MATLAB (The MathWorks, Massachusetts, USA) or *Mathematica* (Wolfram Research, Illinois, USA) for further analysis, or into LaTeX documents for publication. Finally, if the user specifies the length scales of the various sub-units, *SEB* can also evaluate the scattering equations to generate the corresponding scattering profile.

Fig. 2 illustrates the sub-units that we have implemented in this initial release. The figure caption states which reference points we have implemented. These sub-units, together with the large number of linkage options offered by the reference

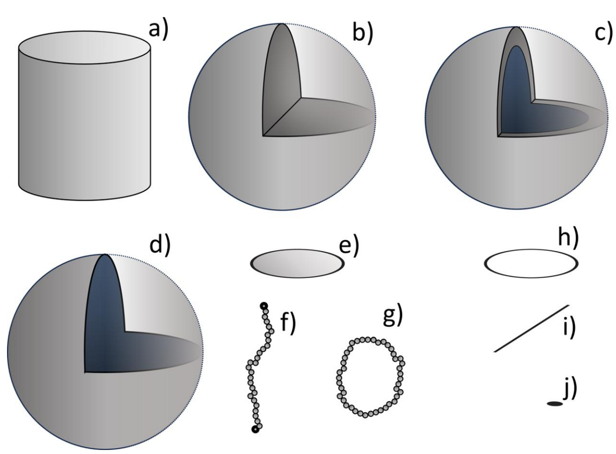


Figure 2
Overview of supported sub-units. (*a*) A solid cylinder (*SEB* name SolidCylinder, reference points by which it can be linked to other sub-units: center, hull, ends, surface), (*b*) a solid sphere (SolidSphere: center, surface), (*c*) a solid spherical shell (SolidSphericalShell: center, surfacei, surfaceo, surface), (*d*) a thin spherical shell (ThinSphericalShell: center, surface), (*e*) a thin disk (ThinDisk: center, surface, rim), (*f*) a linear polymer (GaussianPolymer: end1, end2, middle, contour), (*g*) a polymer loop (GaussianLoop: contour), (*h*) a thin circle (ThinCircle: center, contour), (*i*) a thin rod (ThinRod: end1, end2, middle, contour) and (*j*) an invisible point (Point: point).

points, define a large family of structures for which *SEB* can analytically derive scattering expressions.

SEB has been written in object-oriented C++, which allows the expert user to expand *SEB*, *e.g.* with additional sub-units and/or linkage options, with relative ease. This choice also makes it possible to embed *SEB* within other software programs. *SEB* is open source and is freely available for download from GitHub at https://github.com/tobionecenobi/seb, where technical documentation can also be found. *SEB* depends on the *GiNaC* library (Bauer *et al.*, 2002) for internally representing symbolic expressions, and the *GNU Scientific Library* (Gough, 2009) for evaluating certain special functions.

The paper is structured as follows. In Section 2 we briefly introduce the formalism and logic behind *SEB*. The design and implementation of *SEB* are presented in Section 3. We present four advanced examples in Section 4. Section 5 wraps up the article with a summary and outlook.

2. Formalism

We regard a composite structure as being created by linking sub-units together. For example, the structure of a semi-flexible polymer can be built by linking a sequence of rods end to end to form a linear chain of rods. The structure of a block copolymer or a star polymer can be built by linking two or more polymers together at one end. The structure of a diblock copolymer micelle can be built by linking polymers to the surface of a solid sphere representing the core; here both the polymers and the sphere are sub-units. A bottle-brush polymer structure can be built by linking a number of short polymers to a random point along a long polymer chain.

Common to these example structures is that they are composites made of distinct sub-units linked in specific ways. Sub-units come in two varieties: simple geometric sub-units such as rods and spheres, and sub-units with internal conformational degrees of freedom such as polymers. In the latter case, we need to perform conformational averages when predicting their scattering contributions.

For each type of sub-unit, we define specific reference points on the sub-unit where links can be made. For instance, a linear sub-unit such as a polymer or a rod has two distinct ends. These are points where we can link other sub-units. Each link represents the constraint that a reference point on one sub-unit is co-localized with a reference point on another sub-unit. A sphere can be linked to other sub-units at any random point on its surface. We could also imagine linking at any random point along the contour of the polymer or rod. This illustrates that reference points come in two varieties: specific geometric reference points such as the ends of a polymer or a rod, or distributed reference points such as random points on a geometric surface or along a polymer chain. When predicting scattering contributions, we also have to perform averages over distributed reference points. Even with *e.g.* a polymer sub-unit, we can link it together in many ways forming many structures, *e.g.* block copolymers, star polymers, dendrimers or bottle-brush structures, or any combination of these.

To calculate the scattering from a composite structure we utilize the formalism of Svaneborg & Pedersen (2012*a*,*b*). The formalism is based on three assumptions: (i) a structure does not contain sub-units that are linked into closed loops, (ii) the links are completely flexible and (iii) the sub-unit pairs are mutually non-interacting. These three assumptions ensure that the internal conformation and orientation of all sub-units are statistically independent. Interactions between different sub-units (iii) would for instance create conformational correlations, for example in dense polymers the excluded-volume interactions give rise to correlation hole effects in the scattering (Schweizer & Curro, 1988). When *e.g.* two rods are linked (ii), the joint is flexible and can adopt any angle. If this were not the case, the links would create orientational correlations between the two rods. Finally, if a structure contains loops (i), the closure constraint creates long-range orientational and conformational correlations between all the sub-units involved in the loop. When the internal conformation and orientation of all sub-units are statistically independent, the scattering from a composite structure can be factorized in terms of contributions from individual sub-units. No assumptions are made on the internal structure of the sub-units and no additional assumptions or approximations are made. In this sense the formalism is exact. *SEB* is an implementation of this formalism in C++. Below we introduce *SEB* and the formalism in more detail.

2.1. Links

A sub-unit can have any number of specific and distributed reference points depending on its geometry. To keep track of them *SEB* has hard-coded labels for each reference point. For

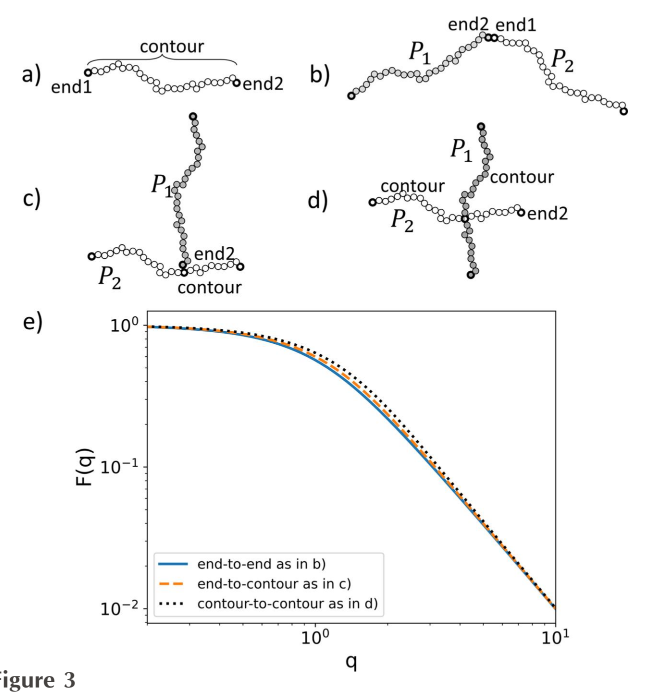


Figure 3
Illustration of a polymer sub-unit. (*a*) The three different reference points. (*b*)–(*d*) The three ways two polymers can be linked. (*e*) The scattering form factors for the different linkage options.

example, a polymer sub-unit has two specific reference points labeled 'end1' and 'end2', while it has one distributed reference point labeled 'contour' [Fig. 3(*a*)]. Hence with just two polymers P1 and P2, we can create three different structures by linking P1.end2 to P2.end1 which produces a linear structure, P1.end2 to P2.contour which produces a random three-functional star structure, or P1.contour to P2.contour which produces a random four-armed star structure. Figs. 3(*b*)–3(*d*) illustrate these structures.

When calculating scattering from structures with distributed reference points, we need to perform an average over random realizations of the link. Hence we will obtain slightly different scattering profiles for these structures. Fig. 3(*e*) shows the scattering form factor for these structures. In the Guinier regime observe that the radius of gyration is largest for the linear structure and smallest for the four-armed star. At small q values [$q = (4\pi/\lambda)\sin\theta$, where θ is half the scattering angle and λ is the wavelength of the incident radiation] the structures produce the same scattering since they have the same scattering lengths, whereas for large q values we observe power-law scattering due to the internal random walk structure of the polymer, which is the same for all three structures.

2.2. Sub-units

A sub-unit is the building block of a structure. It is typically composed of many individual scatterers grouped together. We make no assumptions about the internal structure of a sub-unit. Here and below we use capital Latin letters to denote sub-units.

The scattering contributions of the sub-unit are characterized by the following factors. The form factor is defined as

$$F_I(q) = \left(\sum_i \beta_i\right)^{-2} \sum_{i,j} \beta_i \beta_j \left\langle \frac{\sin(qr_{ij})}{qr_{ij}} \right\rangle, \tag{1}$$

where $r_{ij} = |\mathbf{R}_i - \mathbf{R}_j|$ is the spatial distance between the two scatterers and β_i denotes the excess scattering length of the ith scatterer. The form factor describes the interference contribution from all pairs of scatterers within the Ith sub-unit. Here and below we will use Greek symbols to denote reference points.

For each reference point α, the sub-unit has a corresponding form factor amplitude defined as

$$A_{I\alpha}(\mathbf{q}) = \left(\sum_j \beta_j\right)^{-1} \sum_j \beta_j \left\langle \frac{\sin(qr_{j\alpha})}{qr_{j\alpha}} \right\rangle, \tag{2}$$

where $r_{j\alpha} = |\mathbf{R}_j - \mathbf{R}_\alpha|$ is the spatial distance between the jth scatterer and the reference point. The amplitude describes the phase difference introduced by the spatial distance between the scatterers in the sub-unit and the reference point.

For each pair of reference points α, ω, the sub-unit has a corresponding phase factor defined as

$$\Psi_{I\alpha\omega}(q) = \left\langle \frac{\sin(qr_{\alpha\omega})}{qr_{\alpha\omega}} \right\rangle, \tag{3}$$

where $r_{\alpha\omega} = |\mathbf{R}_\alpha - \mathbf{R}_\omega|$ is the spatial distance between the two reference points. The phase factor describes the phase difference between two specified reference points.

In these expressions we have already performed the orientational average, but an additional average has potentially to be made over internal conformations and/or distributed reference points, for example for a polymer described by Gaussian chain statistics. For the end1 form factor amplitude, one has to perform an average over the distribution of distances between end1 and any scatterer along the chain. For the end1 to end2 phase factor, one has to perform an average of the polymer chain connecting the two ends. For the contour form factor amplitude of a polymer, one has to perform a double average over random positions of the reference point along the chain and any scatterer along the chain. Finally, for the contour to contour phase factor, one has to average over two random positions of the reference point along the chain as well as the Gaussian statistics of the polymer.

In the special case where distributed reference points (*e.g.* contour) and scatterers are characterized by the same distribution, such as a homogeneous distribution along the polymer, then the average expressions for the form factor amplitude and phase factor result in the same expression: the Debye expression for the form factor (Debye, 1947). We refer the reader to Svaneborg & Pedersen (2012*a*,*b*) for the specific expressions.

2.3. Diagrammatic interpretation

A formal derivation of the general scattering expressions for a composite structure has been provided by Svaneborg & Pedersen (2012*a*,*b*). Before stating the general equations, we first motivate the formalism with a diagrammatic derivation of the scattering from an example.

To abstract from the concrete internal details of different sub-units, we illustrate all sub-units as ellipses as shown in Fig. 4. Specific reference points are illustrated as dots on the circumference of the ellipse. Distributed reference points are illustrated as a thick line segment on the circumference of the ellipse to indicate that many points contribute. Fig. 4(*a*) shows a polymer and its diagrammatic representation. To illustrate links, the reference points on two sub-unit ellipses are shown

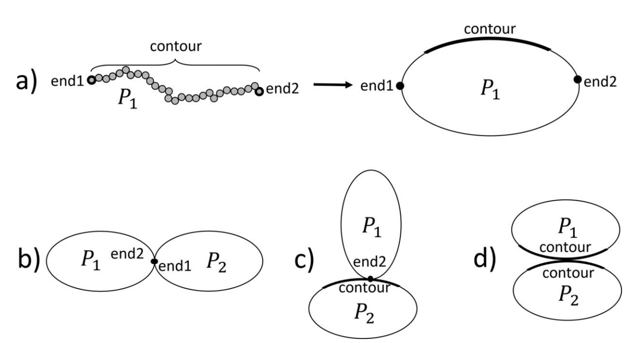


Figure 4
Illustrations of (*a*) how a polymer and its reference points can be represented diagrammatically, and (*b*)–(*d*) how the different linkage options shown in Fig. 3 are represented.

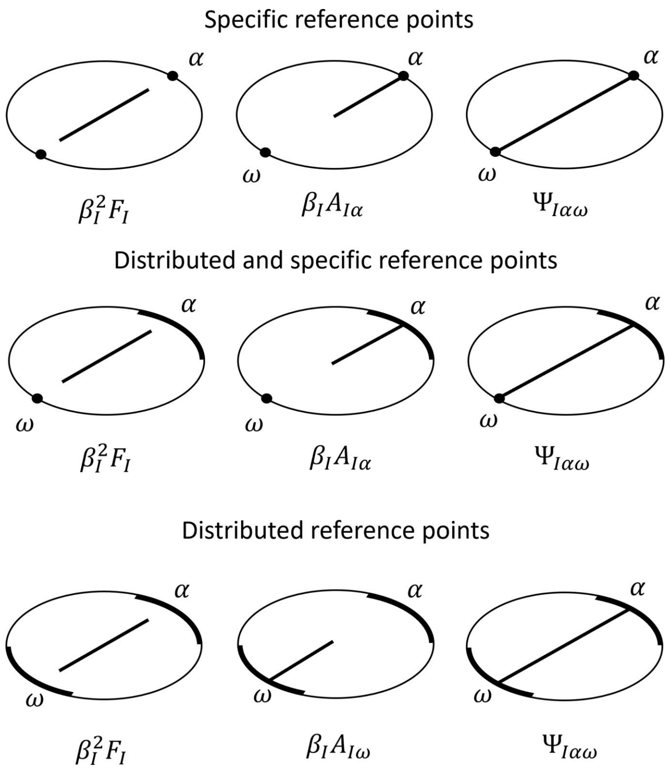


Figure 5
Library of all the possible diagrams and the corresponding factors to use when deriving scattering equations.

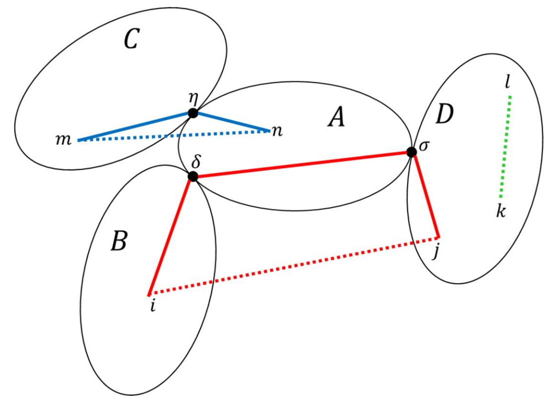


Figure 6
Example structure showing one sub-unit (A) with three pendant sub-units (BCD). The sub-units are linked at three reference points (η, δ and σ). Some scatterers within the sub-units are illustrated as well (lowercase letters). A few distances between scatterers are illustrated (colored dashed lines), together with their representations in terms of paths going through the structure (colored solid lines).

as touching circumferences. The three linkage options shown in Figs. 3(b)–3(d) are illustrated in Figs. 4(b)–4(d), respectively. For simplicity, often herein we will only show and label the reference points of interest when showing structures.

The total library of possible steps and the factors they contribute are shown in Fig. 5. Diagrammatically, form factors are derived from distances between pairs of scatterers within the same sub-unit, and hence they are illustrated as a line inside the ellipse. The form factor is also scaled by the square excess scattering length of the sub-unit. Form factor amplitudes are derived from distances between scatterers and a reference point, and they are illustrated by a line that starts inside the ellipse and ends on the circumference on the reference point. Form factor amplitudes are scaled by the excess scattering length of the sub-unit. Phase factors describe the phase introduced by the distance between two reference points, and hence are illustrated by a line between the two reference points. Since no scatterers are involved, phase factors do not depend on any excess scattering lengths. Finally, when summing over all pairs of sub-units we note that form factors are counted only once, but all interference contributions are counted twice, since both the I, J and J, I paths contribute.

2.4. Algorithm

To calculate the form factor of a composite structure, *SEB* has to account for interference contributions between pairs of scatterers, while also keeping in mind that scatterers are grouped into linked sub-units. Fig. 6 shows three illustrative cases, (i) the l, k scatterers belong to the same sub-unit D, (ii) the n, m scatterers belong to directly linked sub-units A, C, and (iii) scatterers i, j belong to sub-units BD, that are indirectly connected via sub-unit A.

The first case of internal interference contributions between all scatterers within the same sub-unit is described by the form factor of the sub-unit F_D; here and below we suppress the dependency on q for the sake of brevity.

In the second case, the interference contribution between A and C depends on (the average of) the vector $\Delta\mathbf{R} = \mathbf{R}_n - \mathbf{R}_m$, although stepping through the structure we note that $\Delta\mathbf{R} = (\mathbf{R}_n - \mathbf{R}_\eta) + (\mathbf{R}_\eta - \mathbf{R}_m) = \Delta\mathbf{R}_{n\eta} + \Delta\mathbf{R}_{\eta m}$, where each pair of parentheses corresponds to an intra-sub-unit step. Since we have assumed that the sub-units are uncorrelated, the spatial probability distribution of pair distances between scatterers $P_{AC}(\mathbf{R}_{n\eta}, \mathbf{R}_{\eta m})$ can be written as a convolution of the two intra-sub-unit pair-distance distributions relative to the common reference point, $P_A(\Delta\mathbf{R}_{n\eta}) * P_C(\Delta\mathbf{R}_{\eta m})$. In Fourier space, that convolution turns into the product of two sub-unit form factor amplitudes $A_{A\eta}A_{C\eta}$, both of which are evaluated relative to the common reference point η. This is the resulting interference contribution for case (ii).

Finally, the third case generalizes this logic. The interference contribution between scatterers i, j depends on (the average of) the vector $\Delta\mathbf{R} = \mathbf{R}_i - \mathbf{R}_j$. We note again that we can use reference points as stepping stones to write $\Delta\mathbf{R} = (\mathbf{R}_i - \mathbf{R}_\delta) + (\mathbf{R}_\delta + \mathbf{R}_\sigma) + (\mathbf{R}_\sigma - \mathbf{R}_j) = \Delta\mathbf{R}_{i\delta} + \Delta\mathbf{R}_{\sigma\delta} + \Delta\mathbf{R}_{\delta j}$. Each of the three pairs of parentheses describes an intra-sub-unit step. The distribution P_{BAD} is a convolution of individual sub-unit contributions which factorizes into a product of three terms. However, since the middle step involves two reference points, the corresponding contribution is a phase factor. Thus the interference contribution becomes $A_{B\delta}\Psi_{D\delta\sigma}A_{D\sigma}$ for case (iii).

Hence the algorithm used by *SEB* for obtaining the scattering from a composite structure is to analyze all possible

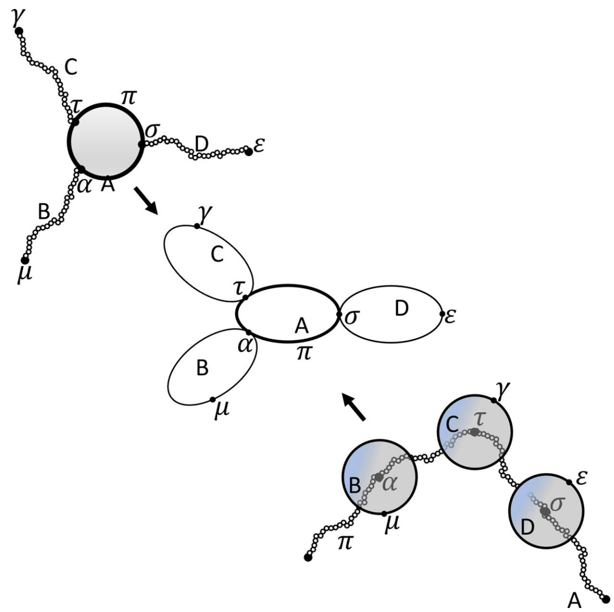


Figure 7
Example structures built up of (top) three polymers linked to the surface of a sphere and (bottom) three spheres linked by their centers to the contour of a polymer, and (center) a generic diagram with the same connectivity.

pairs of scatterers in the same or different sub-units. The form factor is thus a double sum over all sub-units where we encounter three possible types of contributions: The first is a form factor for scattering pairs belonging to the same sub-unit. Each pair of sub-units is either directly or indirectly connected. If they are directly connected, they contribute the product of their form factor amplitudes relative to the common reference point by which they are linked. If they are indirectly connected, we find the unique path through the structure connecting the two sub-units. This path uses reference points as stepping stones. The path is unique since the structure is assumed to be acyclic. The path contributes a form factor amplitude for the first and final sub-units relative to the first and final reference points in the path, respectively. Furthermore, each sub-unit along the path contributes a phase factor, which is to be calculated relative to the two reference points used to step across that sub-unit.

2.5. Example

Fig. 7 shows an example of a block-copolymer micelle modeled as three polymers linked to the surface of a spherical core (Pedersen & Gerstenberg, 1996). The figure also shows an example of three spheres linked by their centers to random positions along the contour of a polymer chain. This could be a beads-on-a-string model of a surfactant-denatured protein (Giehm *et al.*, 2010). In the center of the figure, we show the diagrammatic representation where three sub-units are linked to a central sub-unit. We note that the generic diagram emphasizes the connectivity of the structure and allows us to write down a generic equation for the form factor independent of the specific sub-units involved. In the figure, π denotes the distributed reference point to which the other sub-units are linked.

For the simple example in Fig. 7 we can enumerate all the possible scattering contributions from pairs of scatterers. This is illustrated in Fig. 8, where the top row shows scattering pairs within the same sub-unit and the bottom row displays scattering pairs within directly and indirectly linked sub-units. We note again that all interference terms are counted twice since I, J and J, I interferences contribute the same terms. Form factors only contribute twice. The reason is that, while both r_{ij} and r_{ji} vectors between two scatterers i, j contribute to the form factor, this is already accounted for by equation (1). Summing all the scattering terms we get the (unnormalized) form factor of the structure.

To derive the final expression for a block-copolymer micelle, we have to substitute the concrete polymer expressions for sub-units BCD and the sphere expressions for sub-unit A. To derive the final expression for the beads-on-a-string model, we instead substitute the concrete sphere expressions for sub-units BCD and the polymer expressions for sub-unit A. These expressions are given by Svaneborg & Pedersen (2012*b*).

When requesting the form factor of a structure in *SEB*, either the user obtains a generic structural equation, like the one in Fig. 8, or the default is for *SEB* to perform all the sub-unit substitutions and return a form factor equation for the specific choice of sub-units. For more complex structures, enumerating all the potential scattering contributions by hand is a very tedious and error-prone process. *SEB* automates the

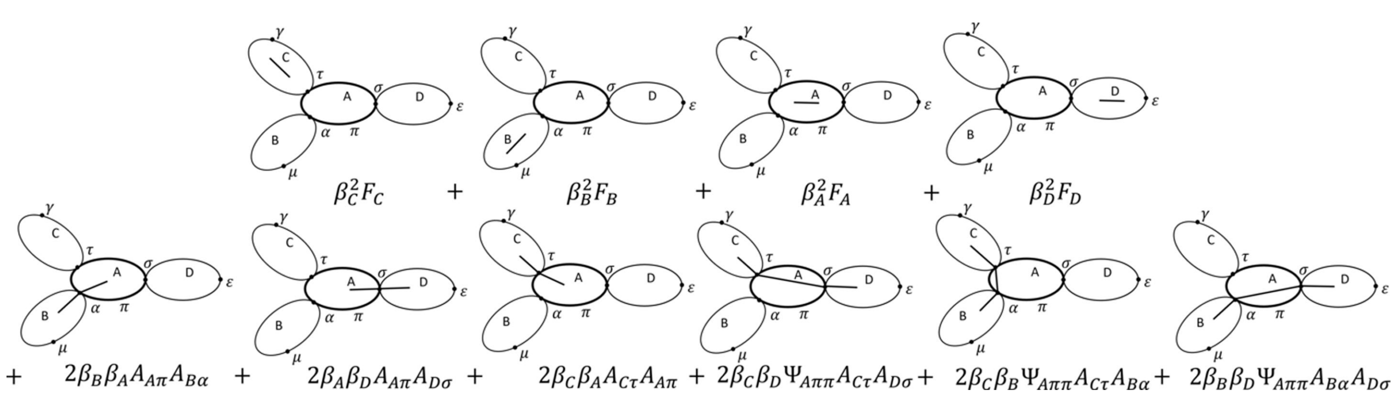


Figure 8
Diagrams of all the contributions to the form factor of an $ABCD$ structure where sub-units BCD are linked to sub-unit A.

process of identifying paths and tallying the corresponding factors.

2.6. Generic equations

Just as a sub-unit has form factor amplitudes and phase factors, so does a composite structure that we have built out of sub-units. Using the diagrammatic logic above, we can also draw the diagrams for the form factor amplitudes of a structure relative to a reference point (not shown). In this case we have to sum over all sub-units in the structure. We find a path from the reference point to the sub-unit. The path contributes a product of phase factors for each sub-unit it traverses, and a form factor amplitude for the last sub-unit along the path relative to the last reference point. To calculate the phase factor of a structure relative to two reference points, we find the path through the structure connecting the reference points. The phase factor of the structure is the product of all the phase factors of sub-units along that path.

Generalizing the logic above, we can state the general expression for the form factor of a structure of sub-units. For each sub-unit pair I, J we identify the first and final reference points α and ω and the path $P(\alpha, \omega)$ through the composite structure that connects them. The scattering interference contribution is then the product of the form factor amplitudes of the first and final sub-units and of all the phase factors of sub-units along the path. The form factor of the composite structure is given by (Svaneborg & Pedersen, 2012*a*,*b*)

$$F_{\rm S}(q) = \beta_{\rm S}^{-2}\Bigg[\sum_I \beta_I^2 F_I(q) + \sum_{I\neq J} \beta_I\beta_J A_{I\alpha}(q)\,A_{J\omega}(q) \prod_{(K,\tau,\eta)\in {\rm P}(\alpha,\omega)} \Psi_{K\tau\eta}(q)\Bigg]. \quad (4)$$

Having derived the form factor, it is straightforward to apply the same logic to state the equivalent form factor amplitude of a structure relative to any reference point it contains, as well as the phase factor of a structure relative to any reference point pair. These are given by (Svaneborg & Pedersen, 2012*a*,*b*)

$$A_{{\rm S}\alpha}(q) = \beta_{\rm S}^{-1}\Bigg[\sum_I \beta_I A_{I\omega}(q) \prod_{(K,\tau,\eta)\in {\rm P}(\alpha,\omega)} \Psi_{K\tau\eta}(q)\Bigg] \quad (5)$$

and

$$\Psi_{S\alpha\omega} = \prod_{(K,\tau,\eta)\in {\rm P}(\alpha,\omega)} \Psi_{K\tau\eta}(q). \quad (6)$$

Usually the focus is on deriving form factors for different structures, and phase factors and form factor amplitudes are just intermediate expressions in the derivation. However, having all three scattering expressions for a structure allows us to use it as a sub-unit. In terms of mathematics, this corresponds to recursively inserting the left-hand sides of equations (4)–(6) into the right-hand sides of the equations. In terms of *SEB*, the code for generating scattering expressions makes recursive calls to itself until terminating at the sub-unit level. This hierarchical view of building structures using simpler sub-structures and sub-units as building blocks is a cornerstone of *SEB*'s design.

The logic is illustrated in Figs. 9(*a*)–9(*c*), where the *ABCD* structure is wrapped into a single structure of type 'star'. In this case, we can think of *e.g.* P1.end2 and S.surface as being the labels of reference points inside a star structure. In Fig. 9(*d*) four instances of a star structure (named star1 to star4) are linked P3.end1 to S.surface. The resulting structure, a linear chain of stars, is shown in Fig. 9(*e*). With *SEB*, we would write code to link sub-units as in Fig 9(*a*), write a line to name the structure 'star' thus realizing Fig. 9(*c*), and proceed to write code to build the structure shown in Fig. 9(*d*) using stars. Finally, with a line of code we obtain the form factor of the structure in Fig. 9(*e*). Towards the end of this paper we give an example where we build a diblock copolymer by joining two polymers. We then build a star by linking four diblock copolymers by one end, and proceed to build a chain where five stars are linked tip to tip. This takes just 13 lines of code to do with *SEB*. Building hierarchical structures from more basic sub-structures vastly accelerates the time it takes to derive the scattering expressions.

Expressions for form factor amplitudes are also useful for modeling structure factor effects. If a structure has a reference point that could be regarded as the 'center' of the structure, then *SEB* can also calculate the form factor amplitude relative to the center point $A_{\rm C}$. In that case, an approximate model for the scattering including structure factor effects would be $I(q) = F(q) + A_{\rm C}^2(q)\,[S_{\rm CC}(q) - 1]$, where $S_{\rm CC}$ is a structure factor that describes the distribution of center to center distances between different structures (Pedersen, 2001; Pedersen *et al.*, 2003). This is analogous to the decoupling approximation (Kotlarchyk & Chen, 1983) for polydisperse or anisotropic particles. The structure factor could for example be modeled

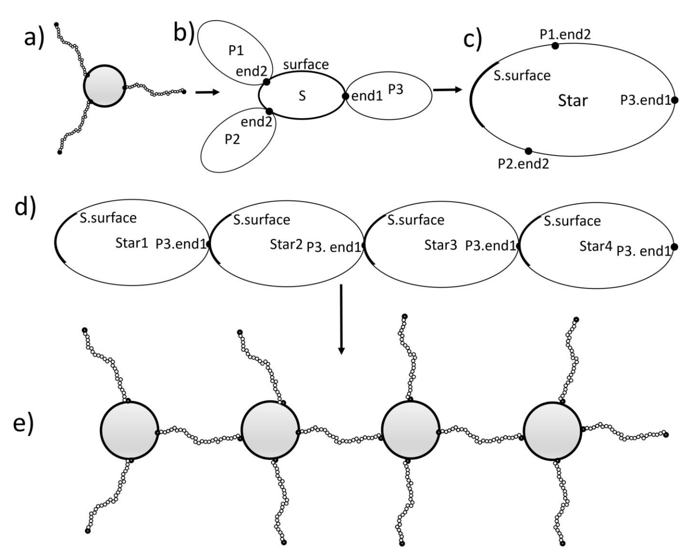


Figure 9
Examples of hierarchical descriptions. (*a*)–(*c*) A bottom-up description: (*a*) a specific star structure made up of sub-units linked to a core, (*b*) a diagrammatic representation of sub-units in the star structure and (*c*) a diagrammatic representation of a star sub-unit. (*d*)–(*e*) A top-down description: (*d*) four linked star sub-units and (*e*) the detailed structure when inserting the internal structure.

as that of a hard-sphere liquid (Wertheim, 1963; Thiele, 1963) or a hard-sphere liquid augmented with a Yukawa tail (Herrera *et al.*, 1999; Cruz-Vera & Herrera, 2008). Structure factor effects can also described using *e.g.* the RPA approximation (Benoit & Benmouna, 1984) or using integral equation theory, *e.g.* in the form of PRISM theory (Schweizer & Curro, 1987; Curro & Schweizer, 1987; Schweizer & Curro, 1994; David & Schweizer, 1994; Yethiraj & Schweizer, 1992). *pyPRISM* is a software package for numerically solving the PRISM equations (Martin *et al.*, 2018). We note that liquid-state theories require the form factor of a structure as an input, which can be derived with *SEB*.

2.7. Estimating sizes

While predicting scattering profiles is the main focus of *SEB*, we can also use analytic Guinier expansions of the scattering expressions to provide expressions for the size of composite structures. The size of a structure or a sub-unit can be gauged by three different measures. The radius of gyration $\langle R_g^2\rangle$ which describes the apparent mean-square distance between unique pairs of scatterers is obtained when expanding the form factor. The (apparent) mean-square distance between a given reference point and any scatterer $\langle R_{I\alpha}^2\rangle$ is obtained when expanding a form factor amplitude. Finally, the mean-square distance between a pair of reference points $\langle R_{I\alpha\omega}^2\rangle$ is obtained when expanding a phase factor. We define the three Guinier expansions for a sub-unit I as

$$F_I(q) \simeq 1 - \frac{q^2}{6}\frac{\left\langle \sum_{ij}\beta_i\beta_j r_{ij}^2\right\rangle}{\left(\sum_i \beta_i\right)^2} + \dots \equiv 1 - \frac{2q^2\left\langle R_{g,I}^2\right\rangle}{6} + \dots, \quad (7)$$

$$A_{I\alpha}(q) \simeq 1 - \frac{q^2}{6}\frac{\left\langle \sum_i \beta_i r_{\alpha i}^2\right\rangle}{\left(\sum_i \beta_i\right)} + \dots \equiv 1 - \frac{q^2\sigma_{I\alpha}\left\langle R_{I\alpha}^2\right\rangle}{6} + \dots, \quad (8)$$

$$\Psi_{I\alpha\omega}(q) \simeq 1 - \frac{q^2\left\langle r_{\alpha\omega}^2\right\rangle}{6} + \dots \equiv 1 - \frac{q^2\sigma_{I\alpha\omega}\left\langle R_{I\alpha\omega}^2\right\rangle}{6} + \dots. \quad (9)$$

Here the right-hand side of the expressions defines the three size measures in terms of the expression in the middle. On the basis of the generic equations (4)–(6), we can derive three similar generic expressions for the size of any composite structure expressed in terms of the sizes of sub-units and paths through the structure. However, for simplicity we have directly implemented the Guinier expanded scattering terms for all sub-units in *SEB*, such that *SEB* explicitly calculates the Guinier expansion above (middle equations) and derives the size from the q^2 term in the expansion (right-hand side).

Extra care has to be taken with regard to double counting of distances. The form factor includes the distance between any pair of scatterers twice, since both r_{ij} and r_{ji} contribute to the form factor. We have made this double counting explicit by the prefactor of two in equation (7). This has the effect of defining the radius of gyration from the *unique set of distances* between pairs of scatterers. For the form factor amplitude and phase factor, we occasionally have to account for a double counting. This done by introducing the double counting factors $\sigma_{I\alpha}$ and $\sigma_{I\alpha\omega}$.

In cases with specific reference points, pair distances between scatterers and reference points are unique by construction and the double counting factor is unity. For instance, for the Guinier expansion of the form factor amplitude of a polymer relative to end1, distances between end1 and scatterers along the polymer are only summed once, and hence $\sigma_{\rm polymer,\,end1} = 1$. Similarly, for the Guinier expansion of the phase factor between end1 and end2 of the polymer, the distance between the two ends of the polymer is summed only once, and hence $\sigma_{\rm polymer,\,end1,\,end2} = 1$.

In cases involving distributed reference points, double counting can occur due to the additional average that has to be performed. For instance, Guinier expansion of the form factor amplitude of a polymer relative to a contour reference point sums every distance between random points and scatterers twice, because both scatterers and reference points are uniformly distributed along the contour of the polymer. Hence $\sigma_{\rm polymer,\,contour} = 2$. Similarly, for the Guinier expansion of the phase factor between a pair of random contour points, we encounter every distance twice. Hence $\sigma_{\rm polymer,\,contour,\,contour} = 2$ in this case as well. In fact, the set of distances between a random point on a polymer and a scatterer, or between two random points on a polymer, is exactly the same as the set of distances between pairs of scatterers, *i.e.* the mean-square distance from contour to scatterer and between two contour points is exactly the radius of gyration of the polymer. If we did not account for double counting in this case, we would have an inconsistency where *e.g.* the distance between randomly chosen points on a polymer would be twice the radius of gyration of the polymer. Note that *SEB* is not able to deduce whether double counting occurs in a given structure. Hence *SEB* returns $\sigma_{I\alpha}\left\langle R_{I\alpha}^2\right\rangle$ and $\sigma_{I\alpha\omega}\left\langle R_{I\alpha\omega}^2\right\rangle$ to the user, and it is up to the user to divide the result by two in the rare cases where double counting has occurred.

3. *SEB*

In the preceding section, we have illustrated the formalism. While it is entirely possible to use the formalism to write down scattering expressions for complex structures by hand, this rapidly becomes tedious and error prone when many paths through a complex structure have to be enumerated, the various expressions for sub-unit factors have to be inserted and the resulting expression implemented in SAS analysis software.

Scattering Equation Builder, *SEB*, is an object-oriented C++ library that automates the process. *SEB* calculates the form factor of a structure by identifying and traversing all the paths between unique sub-unit or sub-structure pairs. *SEB* can also calculate the form factor amplitude for a given reference point by exploring all the paths connecting that reference point to every other sub-unit or sub-structure. Similarly, the phase factor between any two reference points is obtained by identifying the path between the reference points. In the case of hierarchical structures, the algorithm generates ‘horizontal’ paths at a given structural level, and then evaluates scattering expressions by recursively exploring paths through sub-structures until the level of individual sub-units is reached. Internally, we have designed *SEB* to store a hierarchical graph representation of the structures efficiently, and it uses efficient recursive algorithms to generate paths through the hypergraphs at a specified depth into the structure.

SEB uses the *GiNaC* library (Bauer *et al.*, 2002) for representing symbolic expressions. *SEB* depends on the *GNU Scientific Library* (Gough, 2009) for evaluating sine integrals, Bessel functions and Dawson functions. *SEB* also includes code from Moreau (2014) for evaluating Struve functions.

The core functionality of *SEB* is to allow the user to write a short program that (i) builds structures by linking specific uniquely named sub-units, (ii) names a composite structure built up by sub-units, such that it can be used as another sub-unit, (iii) builds hierarchical structures by linking simpler structures together, (iv) obtains analytic expressions characterizing the scattering and sizes of those structures, and/or (v) saves a file with a scattering profile for a chosen set of parameters.

From the user's perspective, *SEB* exposes a very lean interface. Just four methods are available for building structures. The user can choose to obtain generic structural scattering expressions with all sub-unit scattering terms inserted, yielding an equation that depends explicitly on q and a set of structural parameters. The user can also obtain an intermediate representation where scattering terms are inserted but expressed with dimensionless variables, all structural length scales being scaled by q. Lastly, if the user defines the structural parameters and a vector of q values, *SEB* can evaluate the scattering expressions to provide a vector of scattering intensities that can be saved to a file for plotting.

Before going into detail with the implementation and design choices, we start with two simple illustrative examples: a diblock copolymer and a micelle/decorated polymer. These and more examples can be downloaded along with the *SEB* code from https://github.com/tobionecenobi/seb.

3.1. Diblock copolymer

Creating a structure similar to the one seen in Fig. 3(*b*) involves a space (termed a 'world') to host the sub-units, and then creating two polymers and specifying how they are to be linked. The following complete C++ program does this.

```
1:#include "SEB.hpp"
2:int main()
3:{
4: World w;
5: GraphID g=w.Add("GaussianPolymer", "A");
6: w.Link("GaussianPolymer", "B.end1", "A.end2");
7: w.Add(g, "DiBlockCopolymer");
8: cout << latex;
9: cout << w.FormFactor("DiBlockCopolymer");
10:}
```

The first line includes the *SEB* header file, which declares what functions *SEB* provides. Lines 2, 3 and 10 set up the function `main`, which is executed when a program is run. Line 4 in the program creates an instance `w` of the `World` class. This instance provides all *SEB*'s functionality to the user.

To create a structure in the world, we must first add and link the two polymers. In the fifth line, the user uses the `w.Add()` method to add a polymer to the world. `"GaussianPolymer"` refers to a type of polymer described by Gaussian chain statistics. With the second argument, the user assigns the unique name `"A"` to this sub-unit. The world returns a GraphID to the user in response to adding the sub-unit. The GraphID is a common identity shared by all sub-units linked together to form a graph.

In the sixth line, the user uses the `w.Link()` method to add and link a second GaussianPolymer sub-unit. With the second argument the user names this new sub-unit `"B"`. With the second and third arguments the user defines that the new `"B"` should be linked by the end1 reference point to end2 on the already existing `"A"` sub-unit. To calculate the form factor and print it out, we must first wrap the graph formed by these two polymers into a structure. This is done in the seventh line with `w.Add()`, but this time it is called with a GraphID of the structure we want to name and the string `"DiBlockCopolymer"`. Note that all sub-unit and structure names are case sensitive and unique. Types of sub-units and their reference point names are hard coded in *SEB* (see Fig 2). Reference point names are also case sensitive.

Having defined a structure in lines 5–7, we now want to print out the equation for its form factor. The eighth line specifies that we want the expression to be printed in the form of a LaTeX expression. With the command `w.FormFactor("DiBlockCopolymer")` in the ninth line, the user requests the symbolic expression for the form factor. This is printed to the screen (`cout <<`). The form factor equation will be expressed in terms of the magnitude of the momentum transfer q, the structural parameters R_{gA}, R_{gB}, and the excess scattering lengths β_A, β_B. The names of the sub-units are used as subscripts in the parameters used in the scattering expressions.

Here we have chosen LaTeX-formatted output (`latex`), but we could also have output the equation in formats compatible with C/C++ (`csrc`), Python (`python`) or the default *GiNaC* format (`dflt`). The resulting equations are similar, although they use different syntax for powers. The default format uses the '^' operator and Python uses the '**' operator, while the C/C++ format uses the `pow()` function for fractional powers and expands small integer powers into products of terms. The default format is compatible with most CAS systems such as MATLAB, Octave and *Mathematica*, while *SymPy* (https://www.sympy.org) can simplify expressions generated with the Python style. The user can in principle extend *GiNaC* with custom routines for printing equations in other formats. A disadvantage of *GiNaC* is that it does not attempt to simplify expressions or sort terms, so the equations it produces are lengthy. Most often we would export scattering expressions to a symbolic mathematics program for simplification. The resulting expression can then be used *e.g.* in a fitting program, although expressions can also be evaluated directly in *SEB* as shown below.

To change the diblock from end2 to end1 linking to random linking, such as in Fig. 3(*c*), we need to link `A.end2` to a randomly chosen point on `B.contour`. Replacing line 6 with the following code snippet achieves this.

```
6: w.Link("GaussianPolymer", "B.contour#r1", "A.end2");
```

Here, simultaneously with specifying the distributed reference point ″`contour`″ on the ″`B`″ sub-unit, we also label that (now specific) reference point with the arbitrary string ″`r1`″. If we instead want to create the structure of Fig. 3(*d*), we need to link one random reference point on ″`B.contour#r2`″ to a random reference point on ″`A.contour#r3`″. Replacing line 6 with the following code snippet achieves this

```
6: w.Link("GaussianPolymer", "B.contour#r2", "A.contour#r3");
```

The scattering profile corresponding to Figs. 3(*b*)–3(*d*) is shown in Fig. 3(*e*). The difference is not large, but illustrates the point that even with the same sub-units different linkage options affect the scattering profile. The reference point name ″`contour`″ is hard coded in *SEB*, but the user is free to choose the labels (here ″`r1`″, ″`r2`″, ″`r3`″). Having a unique name for each reference point allows us to add more sub-units to the same random point. Having both options for linking allows the user to develop well defined arbitrarily complex branched structures of end-to-end linked polymers, or bottle brush structures where many side chains are randomly attached to a main polymer.

As the default, *SEB* expresses scattering expressions in terms of an explicit q value and a set of structural parameters and excess scattering lengths. The default option is also to output normalized scattering expressions such that they converge to unity in the limit of small q values. Replacing `w.FormFactor(″DiBlockCopolymer″)` by `w.FormFactorAmplitude(″DiBlockCopolymer:A.end1″)` would generate the form factor amplitude expression for the whole DiBlockCopolymer, but expressed relative to the specified reference point. With `w.PhaseFactor(″DiBlockCopolymer:A.end1″, ″DiBlockCopolymer:B.end2″)` *SEB* would instead generate the phase factor of the DiBlockCopolymer relative to the two specified reference points. With `w.FormFactorGeneric(″DiBlockCopolymer″)` we would get the generic form factor of a structure of two connected sub-units without the specific scattering expressions inserted, and this is often useful for debugging. Finally, with `w.RadiusOfGyration2(″DiBlockCopolymer″)` *SEB* would generate the expression for the radius of gyration.

3.2. Diblock copolymer micelle

SEB is not limited to using one type of sub-unit – we can use and link all types of sub-unit to each other. We can, for instance, model a diblock copolymer micelle as a number of polymer chains attached to the surface of a spherical core (Pedersen & Gerstenberg, 1996). Here we limit the number of polymers to three for the sake of simplicity. To generate the micelle shown in Fig. 7 (top), we need to create a solid sphere (″`A`″) and add three polymers (″`B`″, ″`C`″ and ″`D`″) to its surface. The following code snippet does this.

```
World w;
GraphID g=w.Add("SolidSphere","A","s");
w.Link("GaussianPolymer","B.end1","A.surface#p1","p");
w.Link("GaussianPolymer","C.end1","A.surface#p2","p");
w.Link("GaussianPolymer","D.end1","A.surface#p3","p");
w.Add(g, "Micelle");
```

A polymer sub-unit (type GaussianPolymer) has end1, end2 and contour as reference points, while a solid sphere sub-unit (type SolidSphere) has center and surface as reference points. Just as we need to add labels for random points on the contour of the polymer above, we also add labels for the random points on the surface of the sphere. If we used the same label in all three `Link` commands, the three polymers would be linked to the same random point. This would influence the scattering interference between the polymers and is not the structure we are aiming to create.

We also introduce tags in the example, which are an optional parameter of `w.Add()` / `w.Link()`. We tag all polymers as ″`p`″ and the spherical core as ″`s`″. The result is that the scattering expressions are not stated in terms of the unique names A, B, C and D, but are stated using the radius of gyration of the polymers $R_{\rm gp}$ and radius of the sphere $R_{\rm s}$ as well as the two excess scattering lengths $\beta_{\rm p}$ and $\beta_{\rm s}$. If a tag is not specified, then the unique name is used in its place as in the diblock example above. By specifying tags, we can mark a set of sub-units as being identical in terms of their scattering properties and structural parameters.

3.3. Decorated polymer

A model of a surfactant-denatured protein could be a long polymer with some spherical surfactant micelles along its contour. To generate a polymer decorated by three spheres as in Fig. 7 (bottom), we would use the following code snippet.

```
World w;
GraphID g=w.Add("GaussianPolymer","A", "p");
w.Link("SolidSphere","B.center","A.contour#p1","s");
w.Link("SolidSphere","C.center","A.contour#p2","s");
w.Link("SolidSphere","D.center","A.contour#p3","s");
w.Add(g, "DecoratedPolymer");
```

We note that this is nearly identical to the micelle code above, since we link three sub-units to a single sub-unit in both cases. The only difference is that, instead of linking three polymers to a sphere, we link three spheres to one polymer. The three spheres ″`B`″, ″`C`″ and ″`D`″ are tagged with ″`s`″ such that the scattering expression depends on the same parameters as described above.

4. Advanced examples

Having discussed the basics of how to add and link sub-units, create structures and output *GiNaC* expressions, here we show how to implement some of the more advanced examples. In particular, we show a complete example of how to write a program that generates the scattering from 100 identical linked sub-units for a variety of sub-units and linkage options, how to generate a dendritic structure of linked sub-units, an example of polymers and rods linked to the surfaces of different solid geometric objects, and finally how to implement a chain of five linked diblock copolymer stars using hierarchically defined building blocks.

4.1. Chain

To illustrate the versatility of *SEB*, below we show a short C++ program that generates a chain of 100 identical polymers linked end to end. The program obtains the symbolic expression for the form factor, and then uses several helper methods to evaluate this equation for specific parameters, finally producing a file with $F(q)$ versus q.

```
1: World w;
2: GraphID rw = w.Add("GaussianPolymer","P1","p");
3: for (int i=2; i<=100; i++)
4:    {
5:         string now ="P"+to_string(i)  +".end1";
6:         string last="P"+to_string(i-1)+".end2";
7:         w.Link("GaussianPolymer",now,last,"p");
8:    }
9:  w.Add(rw, "RandomWalkPolymer");
10: ex F=w.FormFactor("RandomWalkPolymer");
11: ParameterList params;
12: w.setParameter(params,"beta_p",1);
13: w.setParameter(params,"Rg_p",1);
14: DoubleVector qvec = w.logspace(0.01, 50.0, 400);
15: w.Evaluate(F, params, qvec, "chain_end2end.q");
```

Lines 2–8 create the chain. Initially we add a single polymer ″P1″, and then we use a for loop to add and link 99 more polymers. The polymers have unique names ″P(i)″, where i denotes the number of the sub-unit. The string variables now and last are the names of the current and previous sub-units, respectively. These are all identical and both are tagged as ″p″. The linkage is ″P(i).end1″ to ″P(i-1).end2″ for all polymers such that they form one long continuous chain. In line 9 we name this structure ″RandomWalkPolymer″ and obtain the symbolic expression for its form factor F in line 10. In lines 11–13 we define a list of parameters, and set the excess scattering length ″beta_p″ to one and the radius of gyration ″Rg_p″ also to one. In line 14 we generate qvec, which is a vector of all the q values at which we want to evaluate the form factor. We choose 400 log-equidistant points between $q_{\min} = 0.01$ and $q_{\max} = 50$. From the point of view of *SEB*, units are irrelevant. All scattering expressions depend on dimensionless products of structural length scales and a q value, and as long as both are expressed with a consistent choice of unit, the unit will cancel when evaluating the scattering profile numerically. Finally, in line 15 we evaluate the symbolic expression by inserting the list of parameters and each of the q values into the expression. The result is saved to a file ″chain_end2end.q″. A plot of this file is shown in Fig. 10(*a*).

We can now study how the scattering profile changes when we keep the chain structure but change the sub-unit and/or the linkage. Replacing ″GaussianPolymer″ by ″ThinRod″ directly generates a file with the scattering for a chain of rods linked end to end. This is shown in Fig. 10(*c*). Replacing end1 and end2 by ″contour.r(N)″ and ″contour.s(N-1)″, respectively, produces the contour-to-contour linkage shown in Figs. 10(*b*) and 10(*d*)–10(*f*), where for the last two curves we chose ″GaussianLoop″ or ″ThinCircle″ as sub-units.

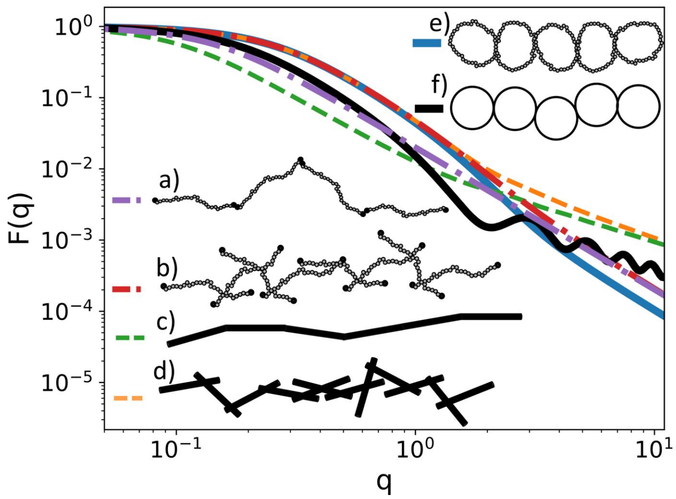


Figure 10
Scattering from a chain of $N = 100$ identical linked sub-units for (*a*) end2-to-end1 linked Gaussian polymers, (*b*) contour-to-contour linked Gaussian polymers, (*c*) end2-to-end1 linked rods, (*d*) contour-to-contour linked rods, (*e*) contour-to-contour linked polymer loops and (*f*) contour-to-contour linked circles. The structural parameters of the sub-units are chosen such that their radius of gyration is one.

In the Guinier regime of Fig. 10, we observe that the end-to-end linked rods have the largest radius of gyration, followed by the end-to-end linked polymers. These form the most loose and extended chain structure. The contour-to-contour linked rods, polymers and loops have the smallest radii of gyration, which is consistent with these chains being the most dense and collapsed structures. Since a chain of 100 end-to-end linked polymers with $R_g^2 = 1$ corresponds to a single polymer with $R_g^2 = 100$, the scattering is the Debye form factor. At large q values, for all polymer structures we observe the $(qR_g)^{-2}$ power law consistent with local random walk statistics. For chains built with rods, we see a $(qL)^{-1}$ power law behavior at large q values, which is expected from a rigid rod. The chain-of-circles structure shows oscillations due to the regular distance between scatterers on a circle, but the trend line of the oscillations follows a q^{-1} power law consistent with a local rod-like structure.

4.2. Dendrimers

Generating a dendritic structure calls for a recursive function.

```
1: GraphID dendrimer = w.Add("Point","center");
2: int count=0;
3: Attach(4, 3, "center.point", count, w);
4: w.Add(dendrimer, "Dendrimer");
```

The challenge here is how to assign names systematically so the links are consistent with a dendritic structure. In line 1 we define a Point, which we call ″center″. This is an invisible sub-unit with zero excess scattering length, but which is useful as a seed to attach other sub-units to. In line 2 we define a counter which will be counting the number of sub-units added. The recursive function `Attach()` generates the dendrimer (see code below) and is called in line 3. The argument `4` is the number of generations to generate, and `3` is the functionality of each connection point. The ″`center.point`″ is the initial reference point on which to graft additional polymers. The two last arguments are the counter and the world we are adding sub-units into. In the last line we name the resulting structure ″`Dendrimer`″. The rest of the code for generating a file with the form factor is identical to the chain example above.

```
1:void Attach(int g, int f, string ref, int& c, World& w)
2:{
3: int arm=f-1;
4: if (ref=="center.point") arm=f;
5: for (int i=0;i<arm;i++)
6:  {
7:   string name = "S"+to_string(c)+".end1";
8:   w.Link("GaussianPolymer", name, ref, "poly");
9:   string newref  = "S"+to_string(c)+".end2";
10:  c++;
11:  if (g>1) Attach( g-1, f, newref, c, w);
12: }
13:}
```

The recursive function receives `g` the number of generations that remain to be attached, `f` the functionality of each link and `ref`, which is the reference point from the previous generation onto which we link the current generation. `c` and `w` are a global counter and the world, respectively. In lines 3–4 we define the numbers of arms to attach to this reference point. Usually this is `f` − 1 since we are linking to the tip of an existing branch, but in the special case where we are linking arms to the `center.point` we need to add `f` arms instead. This ensures that all connection points have the desired functionality.

In lines 5–12 we add the arms and link them to the previous generation. In line 7 we define a name for each new sub-unit ″`S(c)`″ and in line 8 we add GaussianPolymer sub-units and link them to the tip of the previous generation. The links are ″`S(c).end1`″ to `ref`, where `ref` is the tip of the last generation of polymers. In line 9 we define the new reference point on which to add the next generation. This reference point is ″`S(c).end2`″. Finally in line 10, we increment the counter of sub-units that have been added so far. If at this point we have not finished building, that is if `g` is larger than one, in line 11 we again call the `Attach` function to attach the next generation to the tip of the current arm, that is to `newref`. The attachment process continues recursively, decrementing `g` with each generation of branches attached. The final generation is `g=1`, which corresponds to the outermost leaves on our dendritic structure.

The resulting structure contains 45 sub-units (three from the first generation, six from the second generation, 12 from the third generation and 24 from the fourth generation) The code above generates the structure plotted in Fig. 11(*a*). Again, by changing line 8 we can link other sub-units such as thin rods. Changing lines 7 and 9, we can change the reference

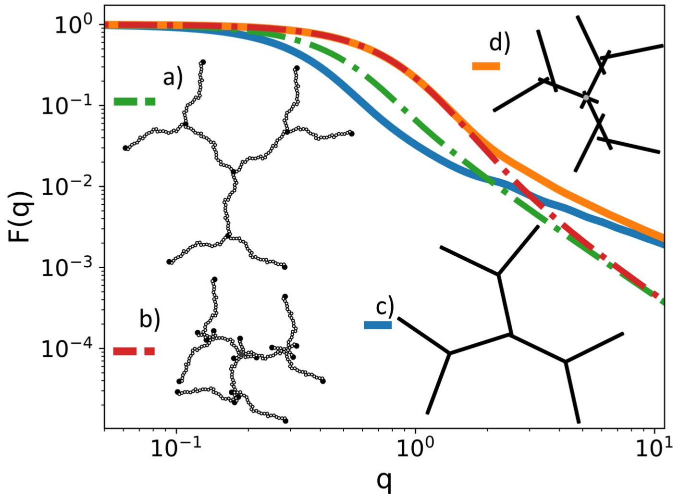


Figure 11
Scattering from a dendrimer with four generations and three functional links. (*a*) end1-to-end2 linked polymers, (*b*) contour-to-contour linked polymers, (*c*) end1-to-end2 linked rods and (*d*) contour-to-contour linked rods. The structural parameters of the sub-units are chosen so the radius of gyration is always one. The sketches of the dendrimer structures only show the first two generations for the sake of brevity.

points from end-to-end to contour-to-contour links. The results are the four curves shown in Fig. 11. Again, we observe in the Guinier regime that dendrimers made by end-to-end linked rods and polymers have the largest radii of gyration. We also observe that at large q values the power laws $(qL)^{-1}$ for rods and $(qR_g)^{-2}$ for polymers show what sub-units they are built with. We also observe that contour-to-contour linked structures have the same radius of gyration, independent of their sub-unit structure.

4.3. Solids

With *SEB* we can investigate how different linkage options of sub-units on the surfaces of solid bodies affect the scattering.

```
1: GraphID str = w.Add( "SolidSphericalShell", "shell");
2: for (int i=1; i<=50; i++)
3:  {
4:   string name1= "poly"+to_string(i)+".end1";
5:   string ref1 = "shell.surfaceo#p"+to_string(i);
6:   w.Link( "GaussianPolymer", name1, ref1, "poly");
7:   string name2= "rod"+to_string(i)+".end1";
8:   string ref2 = "shell.surfacei#r"+to_string(i);
9:   w.Link( "ThinRod", name2, ref2, "rod");
10: }
11: w.Add(str, "Structure");
```

In the example code above, we generate a solid spherical shell in line 1. The shell is a homogeneous solid body defined by an exterior radius R_o and an interior radius R_i. In lines 4–6, we add and link a Gaussian polymer. The polymer is named ″`poly(i)`″ and linked by ″`poly(i).end1`″ to ″`shell.surfaceo#p(i)`″, where `surfaceo` denotes distributed reference points on the ‘outer’ or exterior surface of the shell. The unique label ″`p(i)`″ ensures that all polymers are linked to different random points on the surface. In lines 7–9, we add and link a thin rod. The rod is named "rod(i)" and linked by "rod(i).end1" to "shell. surfacei#r(i)", where surfacei denotes the interior surface. Again the unique label "r(i)" ensures that rods are linked to different random points. In line 11 we name the resulting structure "Structure". As in the chain example, we evaluate the form factor and generate a file with the corresponding scattering curve.

Changing line 1, we can change which solid body we are attaching sub-units to, *e.g.* solid spheres or cylinders. Changing lines 6 and/or 9, we can change what sub-units we link to the surface and by which reference point the link should be made. Changing the reference points in lines 5 or 8, we can choose different linkage options on the solid body. Fig. 12 shows a comparison of some of the possible linkage options. The code above corresponds to the Fig. 12(*d*) curve. Here, we choose to contrast match the solid body $\beta_{\rm shell} = 0$, and choose $\beta_{\rm poly} = \beta_{\rm rod} = 1$. Hence the scattering is due to both the polymers and rods and their interference contribution, which depends on the shape of the body to which they are attached.

In the Guinier regime of the scattering profiles shown in Fig. 12, we observe that the solid spheres and spherical shells are nearly identical, as is the scattering from cylinders. This is not surprising since the scattering between different sub-units is modulated by the phase factor of the solid body to which the sub-units are attached. At very large q values we observe a power-law behavior with an exponent slightly larger than -1. This is to be expected, since the scattering is dominated by the sub-unit form factors, and asymptotically the rod $(qL)^{-1}$ will dominate over the polymer $(qR_{\rm g})^{-2}$ unless the number of polymers vastly outnumbers the number of rods. In the crossover regime, we observe different oscillations for the different linkage options. These oscillations are due to the different distributions of surface-to-surface distances between the tethering points of pairs of rods and/or polymers.

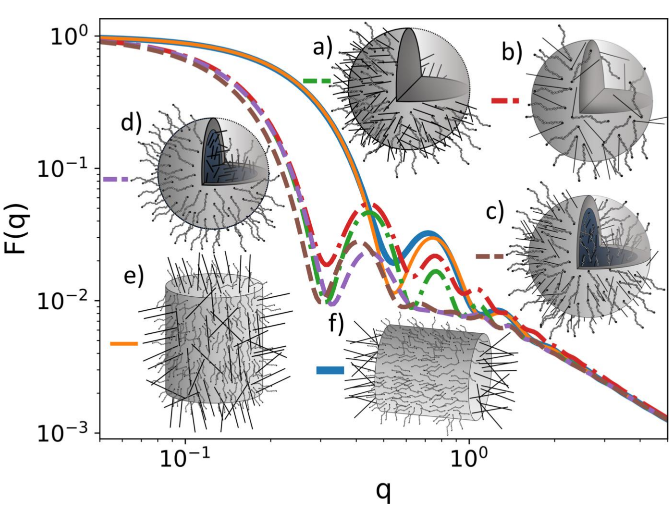


Figure 12
Scattering from various solid bodies with 50 rods and 50 polymers attached to different surfaces. The solid body is contrast matched, $\beta_{\rm solid} = 0$, and $\beta_{\rm poly} = \beta_{\rm rod} = 1$. (*a*) A solid sphere $R = 10$ with rods and polymers randomly attached to the surface. (*b*) A solid sphere $R = 10$ with pairs of rods and polymers attached to the same random point. (*c*) A solid spherical shell $R_{\rm i} = 8$, $R_{\rm o} = 12$ with rods and polymers randomly attached to the interior and exterior surfaces. (*d*) A solid spherical shell $R_{\rm i} = 8$, $R_{\rm o} = 12$ with 50 rods attached to the interior surface and 50 polymers attached to the exterior surface. (*e*) A cylinder $L = 10$, $R = 5$ with rods and polymers randomly attached to the surfaces. (*f*) A cylinder $L = 10$, $R = 5$ with rods attached to the two cylinder ends and polymers attached to the hull. For curves (*c*) and (*e*) where several surfaces contribute area, we have weighted the scattering terms with their respective area fractions to ensure homogeneous area coverage in the case of random attachment.

4.4. Hierarchical structures

In the examples above we have built structures by connecting sub-units to each other. The result of each was described by a GraphID that we could name as a type of structure, and then we could use that name to derive various scattering expressions. Since the formalism is complete any sub-structure can be used as a sub-unit. World has a Link method that takes a GraphID (referring to a type of structure) and names and links it to an existing structure. This works analogously to Link called with a string denoting a type of sub-unit. The code below illustrates the concept.

```
1: GraphID d = w.Add("GaussianPolymer","A");
2: w.Link( "GaussianPolymer","B.end1","A.end2");

3: GraphID s = w.Add(d,"diblock1");
4: w.Link(d,"diblock2:A.end1","diblock1:A.end1");
5: w.Link(d,"diblock3:A.end1","diblock1:A.end1");
6: w.Link(d,"diblock4:A.end1","diblock1:A.end1");

7: GraphID c = w.Add(s,"star1");
8: w.Link(s,"star2:diblock1:B.end2","star1:diblock3:B.end2");
9: w.Link(s,"star3:diblock1:B.end2","star2:diblock3:B.end2");
10:w.Link(s,"star4:diblock1:B.end2","star3:diblock3:B.end2");
11:w.Link(s,"star5:diblock1:B.end2","star4:diblock3:B.end2");

12:w.Add(c, "chain");
```

In line 1 we add a Gaussian polymer sub-unit "A", and in line 2 we add and link another Gaussian polymer "B" sub-unit to it as we have done in several of the examples above. The names "A" and "B" should be thought of as two instantiations of the type of object with an internal structure described by the type "GaussianPolymer". It is important to distinguish between concrete objects of a certain type of structure and the type of structure itself. The type does not exist per se, but is just a generic description. In the case of "A" and "B" these have their own structural parameters and contribute specific terms to scattering expressions. The type GaussianPolymer is a description of the internal chain statistics of a polymer molecule. When creating a new sub-unit or structure in *SEB*, we instantiate it from a type of structure. GraphID variables are also types of structure; in particular, the GraphID variable d describes a diblock copolymer structure. In line 3, we add a new structure to the world named "diblock1", which is an instantiation of the diblock type. Hence "diblock1" is a concrete structure in the same sense as "A" and "B" are concrete sub-units.

In lines 4–6, we do something new; we call Link() not with a sub-unit type but with the diblock type (GraphID variable d). We name these three new structures "diblock2", "diblock3" and "diblock4", respectively. Each structure is linked by a reference point inside the structure to a reference point that already exists in the world. For the diblock2 structure, we link "diblock2:polyA.end1" to "diblock1:polyA.end1"; since "diblock1" already exists in the world we can link to it. To link structures, we need to specify the path to get from the structure level via sub-structures down to the reference point which is associated with a specific sub-unit. Since all names are unique, so is any path from a sub-structure to a reference point. The resulting structure is a four-armed diblock copolymer star, where the "A" blocks are linked by their end1 reference points which form the center of the star, while the corona is formed by the four "B" blocks and their free chain ends are at the end2 reference points.

While we usually define the GraphID by the returned value of the first Add() method, all subsequent Link() calls also return the same GraphID value, since this is associated with the whole graph first created by Add() and then grown each time the Link() is called. In line 3, we store the type of graph formed by "diblock1" to "diblock4" in the GraphID variable s, which is now the type of a four-armed diblock star structure.

In line 7, we now instantiate a star sub-structure and name it "star1". This defines a new GraphID which we save in a variable c. Then in lines 8–11 we proceed to instantiate four more star sub-structures named "star2" to "star5". Each time we link "star(n):diblock1:B.end2" to "star(n-1):diblock3:B.end2", since "star(n-1)" already exists and has a "diblock3:B.end2" reference point inside it. The result is a linear chain of stars formed by linking the tips of "diblock1" and "diblock3"; hence "diblock2" and "diblock4" form dangling ends analogous to a bottle-brush structure. Finally, to calculate the form factor of this type of chain, we must name it to instantiate it in the world. The rest of the code is similar to the chain example above.

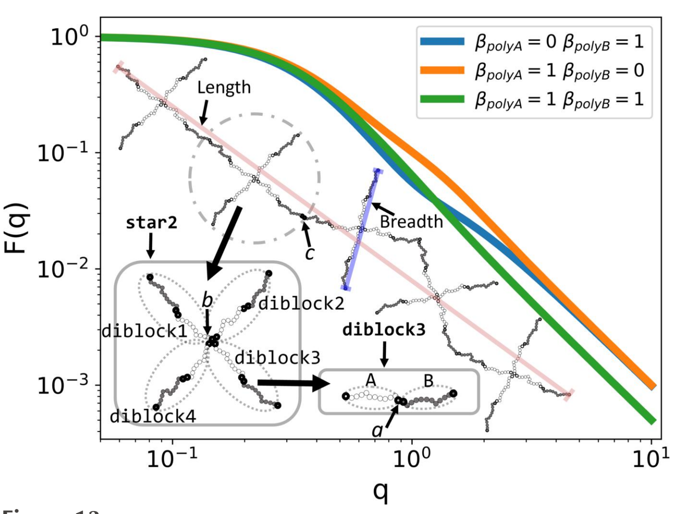


Figure 13
Scattering from a chain of five four-armed stars where each arm is a diblock copolymer for three different choices of contrast. The illustrated links are (*a*) the block copolymer formed by "A.end2" to "B.end1", (*b*) the star formed by "diblock2:A.end1" to "diblock1:A.end1" and similar for the other arms, and (*c*) the chain formed by "star2:diblock1:B.end2" to "star1:diblock3:B.end2" and similar for the other stars.

This example illustrates the power of building structures using more simple sub-structures as building blocks. With 12 lines of code, we have generated a hierarchical structure with 40 sub-units. Fig. 13 shows an illustration of the resulting structure, together with the form factor evaluated for three different contrast options. In the Guinier regime, we observe that the radius of gyration is nearly the same, independent of contrast, which we would also expect for such a structure. At large q values we obtain the characteristic power law of polymer sub-units. For intermediate q values the structure is slightly different. When the "polyA" blocks are contrast matched, $\beta_A = 0$, they play the role of invisible spacers inside the stars. When the "polyB" blocks are contrast matched, they play the role of invisible spacers between different stars.

Besides calculating scattering expressions, *SEB* can also provide expressions characterizing the size of a structure. For instance, w.RadiusOfGyration2("chain") returns an expression for the radius of gyration by applying a Guinier expansion of all sub-unit scattering terms. After simplification, the result is

$$\langle R_{\rm g}^2\rangle = (\beta_A + \beta_B)^{-2}\Big(12.9R_{gA}^2\beta_B^2 + 9.6R_{gB}^2\beta_A^2 + 21R_{gB}^2\beta_A\beta_B + 24.3R_{gA}^2\beta_A\beta_B + 11.3R_{gB}^2\beta_B^2 + 11.3R_{gA}^2\beta_A^2\Big). \quad (10)$$

The radius of gyration measures the distances between all pairs of scatterers, so we could for instance also ask what is the mean-square distance between the center of the star and all scatterers in the structure. A Guinier expansion of the corresponding form factor amplitude provides the result, and "star3:diblock1:polyA.end1" is the reference point at the center of the star. Hence this mean-square distance gives an idea of the radial extent of the structure. Calling w.SMSD_ref2scat("chain:star3:diblock1:polyA.end1") returns that result. The method is called SMSD for sigma mean-square distance to remind the user to account for a potential symmetry factor. Finally, we could ask what are the length and breadth of the structure? To calculate the length, we call w.SMSD_ref2ref("chain:star1:diblock1:polyB.end2", "chain:star5:diblock3:polyB.end2") which returns the mean-square distance between the two reference points at either end of the structure. The result is $\langle R_{\rm length}^2\rangle = 60(R_{gB}^2 + R_{gA}^2)$ To estimate the breadth of the structure, we change the reference points to w.SMSD_ref2ref("chain:star3:diblock2:polyB.end2", "chain:star3:diblock4:polyB.end2"), since "diblock2" and "diblock4" are the two dangling diblocks, and "polyB.end2" are the dangling ends of these diblocks. The result is $\langle R_{\rm breadth}^2\rangle = 12(R_{gB}^2 + R_{gA}^2)$.

These results are easy to obtain by hand. For a single polymer $R_{\rm g}^2(N) = \langle R_{\rm end2end}^2\rangle/6 = b^2N/6$, where b is the random walk step length and N is the number of steps in the polymer. Then to estimate the number of steps along the length of the chain, we note that it has ten A blocks and ten B blocks from one end to the other. Hence $\langle R_{\rm length}^2\rangle = b^2N_{\rm length} = b^2(10N_A + 10N_B) = 60(R_{gA}^2 + R_{gB}^2)$. For the breadth, a star has a breadth of $N_{\rm breadth} = 2N_A + 2N_B$. The result is that the chain is five times longer than its breadth, which is what one would expect.

5. Summary

The main problem in analyzing small-angle scattering (SAS) data is the availability of model expressions for fitting. Here we have presented *Scattering Equation Builder* (*SEB*), which is an open-source C++ library available at https://github.com/tobionecenobi/seb. *SEB* automates part of this problem by generating symbolic expressions for complex composite models of structures using the formalism presented by Svaneborg & Pedersen (2012*a*,*b*). The formalism is built on the assumptions that sub-units are mutually non-interacting, that structures do not contain loops and that all links are assumed to be completely flexible. No further mathematical simplifications or approximations are made. In particular, no assumptions are made regarding the internal structure of the sub-units.

With *SEB* the user writes short programs that construct a structure using sub-units and simpler structures as building blocks. Much like the construction toy LEGO, sub-units can be linked at certain points called reference points. Either these can be specific geometric points, such as one of the ends of a polymer, or they can be randomly distributed, *e.g.* on the surface of a sphere. With the building blocks of sub-units and the reference points, a large number of complex structures can be built with relative ease. See Fig. 2 for the sub-units and reference points supported by this initial release.

SEB derives analytic symbolic expressions for the form factor, form factor amplitude and phase factor of a structure. *SEB* can also derive expressions for the radius of gyration, as well as for the mean-square distance between a reference point and all scatterers in a structure. Finally, *SEB* can derive the mean-square distance between pairs of reference points. The expressions can be evaluated to a number *e.g.* when fitting, evaluated to produce a file for plotting, output in several formats for LaTeX documentation or C/C++ and Python-compatible equations, or exported to MATLAB or *Mathematica*.

In the present article, we have given simple illustrative examples and some more complex examples of what *SEB* can do. *SEB* is available on GitHub (https://github.com/tobionecenobi/seb), and a frozen version related to the present work is deposited on Zenodo (*SEB* Version 1.0, https://doi.org/10.5281/zenodo.10204364). We hope the *SEB* library will grow as more sub-units become supported, and we welcome contributions from users in developing future versions of the library.

References

Bauer, C., Frink, A. & Kreckel, R. (2002). *J. Symbolic Comp.* **33**, 1–12.

Benoit, H. & Benmouna, M. (1984). *Polymer*, **25**, 1059–1067.

Breßler, I., Kohlbrecher, J. & Thünemann, A. F. (2015). *J. Appl. Cryst.* **48**, 1587–1598.

Cruz-Vera, A. & Herrera, J. N. (2008). *Physica A*, **387**, 5696–5706.

Curro, J. G. & Schweizer, K. S. (1987). *Macromolecules*, **20**, 1928–1934.

David, E. F. & Schweizer, K. S. (1994). *J. Chem. Phys.* **100**, 7767–7783.

Debye, P. (1947). *J. Phys. Chem.* **51**, 18–32.

Doucet, M., Cho, J. H., Alina, G., Attala, Z., Bakker, J., Bouwman, W., Butler, P., Campbell, K., Cooper-Benun, T., Durniak, C., Forster, L., Gonzalez, M., Heenan, R., Jackson, A., King, S., Kienzle, P., Krzywon, J., Murphy, R., Nielsen, T., O'Driscoll, L., Potrzebowski, W., Prescott, S., Ferraz Leal, R., Rozyczko, P., Snow, T. & Washington, A. (2021). *SasView.* Version 5.0.4. https://doi.org/10.5281/zenodo.4467703 and https://www.sasview.org.

Feigin, L. A. & Svergun, D. I. (1987). *Structure Analysis by Small-Angle X-ray and Neutron Scattering.* Heidelberg: Springer.

Förster, S., Apostol, L. & Bras, W. (2010). *J. Appl. Cryst.* **43**, 639–646.

Giehm, L., Oliveira, C. L. P., Christiansen, G., Pedersen, J. S. & Otzen, D. E. (2010). *J. Mol. Biol.* **401**, 115–133.

Gough, B. (2009). *GNU Scientific Library Reference Manual.* Godalming: Network Theory Ltd.

Guinier, A., Fournet, G. & Yudowitch, K. L. (1955). *Small-Angle Scattering of X-rays.* Wiley New York.

Herrera, J. N., Cummings, P. T. & RuÍz-Estrada, H. (1999). *Mol. Phys.* **96**, 835–847.

Ilavsky, J. & Jemian, P. R. (2009). *J. Appl. Cryst.* **42**, 347–353.

Kohlbrecher, J. & Breßler, I. (2022). *J. Appl. Cryst.* **55**, 1677–1688.

Konarev, P. V., Petoukhov, M. V., Volkov, V. V. & Svergun, D. I. (2006). *J. Appl. Cryst.* **39**, 277–286.

Kotlarchyk, M. & Chen, S. H. (1983). *J. Chem. Phys.* **79**, 2461–2469.

Manalastas-Cantos, K., Konarev, P. V., Hajizadeh, N. R., Kikhney, A. G., Petoukhov, M. V., Molodenskiy, D. S., Panjkovich, A., Mertens, H. D. T., Gruzinov, A., Borges, C., Jeffries, C. M., Svergun, D. I. & Franke, D. (2021). *J. Appl. Cryst.* **54**, 343–355.

Martin, T. B., Gartner, I. T. E., Jones, R. L., Snyder, C. R. & Jayaraman, A. (2018). *Macromolecules*, **51**, 2906–2922.

Moreau, J.-P. (2014). *Special Functions in C/C++*, https://web.archive.org/web/20231003013905/https://jean-pierre.moreau.pagesperso-orange.fr/.

Pedersen, J. (1997). *Adv. Colloid Interface Sci.* **70**, 171–210.

Pedersen, J. S. (2001). *J. Chem. Phys.* **114**, 2839–2846.

Pedersen, J. S. & Gerstenberg, M. C. (1996). *Macromolecules*, **29**, 1363–1365.

Pedersen, J. S., Svaneborg, C., Almdal, K., Hamley, I. W. & Young, R. N. (2003). *Macromolecules*, **36**, 416–433.

Pedersen, M. C., Arleth, L. & Mortensen, K. (2013). *J. Appl. Cryst.* **46**, 1894–1898.

Perkins, S. J., Wright, D. W., Zhang, H., Brookes, E. H., Chen, J., Irving, T. C., Krueger, S., Barlow, D. J., Edler, K. J., Scott, D. J., Terrill, N. J., King, S. M., Butler, P. D. & Curtis, J. E. (2016). *J. Appl. Cryst.* **49**, 1861–1875.

Schneidman-Duhovny, D., Hammel, M., Tainer, J. A. & Sali, A. (2016). *Nucleic Acids Res.* **44**, W424–W429.

Schweizer, K. S. & Curro, J. G. (1987). *Phys. Rev. Lett.* **58**, 246–249.

Schweizer, K. S. & Curro, J. G. (1988). *Macromolecules*, **21**, 3070–3081.

Schweizer, K. S. & Curro, J. G. (1994). *Adv. Polym. Sci.* **116**, 319–377.

Spinozzi, F., Ferrero, C., Ortore, M. G., De Maria Antolinos, A. & Mariani, P. (2014). *J. Appl. Cryst.* **47**, 1132–1139.

Svaneborg, C. & Pedersen, J. S. (2012*a*). *J. Chem. Phys.* **136**, 104105.

Svaneborg, C. & Pedersen, J. S. (2012*b*). *J. Chem. Phys.* **136**, 154907.

Svergun, D., Barberato, C. & Koch, M. H. J. (1995). *J. Appl. Cryst.* **28**, 768–773.

Svergun, D. I., Richard, S., Koch, M. H. J., Sayers, Z., Kuprin, S. & Zaccai, G. (1998). *Proc. Natl Acad. Sci. USA*, **95**, 2267–2272.

Thiele, E. (1963). *J. Chem. Phys.* **39**, 474–479.

Wertheim, M. S. (1963). *Phys. Rev. Lett.* **10**, 321–323.

Yethiraj, A. & Schweizer, K. S. (1992). *J. Chem. Phys.* **97**, 1455–1464.