Wagener/`sasmodels` comparison

Nouhaila Agouzal, Dorian Lozano

Supervisor: Miguel Gonzalez

ILL, August 2023

"All things are either good or bad by comparison."

Edgar Allan Poe

Contribution of Authors

- **Dorian Lozano**: Software, Writing - Original Draft
- **Nouhaila Agouzal**: Software, Writing - Review & Editing

Contents

1. Introduction **2**
1.1. What is Wagener's method ? . . . 2
1.2. What did we code ? . . . 2

2. Comparison of data **4**
2.1. $F^2(q)$ comparison . . . 5
2.2. $I(q)$ VS $P(q)$ comparison . . . 9

3. Comparison of time **12**
3.1. Naive benchmark . . . 12
3.1.1. $F^2(q)$ . . . 13
3.1.2. $P(q)$ . . . 15
3.2. Less naive benchmark . . . 17
3.2.1. Wagener with `cytypes` VS Wagener with `sasmodels` . . . 18
3.2.2. Wagener with `sasmodels` VS (SasView with) `sasmodels` . . . 20

A. How to use the code provided **21**
A.1. Test a model . . . 21
A.2. Launch a data benchmark . . . 21
A.3. Launch a time benchmark . . . 22

B. List of Figures **23**

1. Introduction

Welcome!

This document uses several boxes to precise some points:

- the **Important note** boxes specify parameters or key code points,
- the simple **Note** boxes specify less important but still interesting points,
- the **Possible improvement** boxes explain how the program could be improved, with another model or a particular concept.

Internal links are in blue, external links are in midnight blue.

1.1. What is Wagener's method ?

We will call "Wagener's method" the method described in the "Fast calculation of scattering patterns using hypergeometric function algorithms" article written by Michael Wagener and Stephan Förster. This method is supposed to compute two scattering patterns : the form factor $P(q)$ and the scattering amplitude $F^2(q)$, while it's supposed to provide very significant speed gains compared to what is developed in `sasmodels` and used in SasView. The goal of this paper is to benchmark `sasmodels` and Wagener's method.

Wagener's method stands out due to the following characteristics:

- the use of hypergeometric functions (series),
- the computation of q-independent factors,
- the use of `Cuda`.

Run Wagener's code[1] on our machines was challenging as we couldn't fulfill all requirements to do so. In addition, the models were hard to excerpt from the `.cu` files. As a consequence, our supervisor Miguel GONZALEZ proposed to code from scratch a couple of models.

1.2. What did we code ?

We coded scattering patterns for the four following models:

1. sphere,
2. biaxial ellipsoid,
3. cube,
4. (small axial ratios) cylinder.

The scattering patterns were coded in `C` (we didn't use `Cuda`) and are based on Mathematica (Wolfram langage) code provided by Wagener in its Supporting Information. For each model, we coded the computation of:

- $P(q)$ with mono and polydispersity,
- $F^2(q)$ with mono and polydispersity.

[1]Originally coded in `C++` and `Cuda`

To sum up what we coded, here is a recap chart of the different pages of the Supporting Information, in which one can found the Mathematica code used to write our own `C` code:

Table 1: Pages of coded models

	$P(q)$	$F^2(q)$
Sphere	28	29-30
Biaxial Ellipsoid	33-34	36
Cube	46-47	49
Cylinder	60	63-64

To be sure that the comparison is fair, we have to compare the scattering patterns computed by Wagener's method with the scattering patterns computed with SasView (`sasmodels`).

Important note 1

All the following benchmarks, time comparisons, etc... will **always** compare:

- Wagener's **Regime I**[a] computations

with

- `sasmodels` computations

[a]The differences between the several regimes are explained in the SI, page 20.

Note 1

In the Supporting Information, Wagener only provided code for samples with polydispersity. Some following plots show $F^2(q)$ and $P(q)$ with **mono**dispersity. The code is the same, but has been simplified. We just had to compute some mathematical equivalences for $\sigma \to 0$ (or $\frac{1-\sigma^2}{\sigma^2} =: z \to +\infty$).
For instance, for $z = +\infty$, the line of code:

```c
double example = pow(z+1, 4)*tgamma(z-3)/tgamma(z+1);
```

becomes:

```c
double example = 1.;
```

because, for $z > 3$:

$$\frac{(z+1)^4\Gamma(z-3)}{\Gamma(z+1)} = \frac{(z+1)^4\Gamma(z-3)}{z(z-1)(z-2)(z-3)\Gamma(z-3)} = \frac{(z+1)^4}{z(z-1)(z-2)(z-3)} \underset{z\to+\infty}{\sim} 1.$$

However, it is still possible to choose a low value of σ, for instance 10^{-10}, to simulate monodispersity with Wagener's code, but the computations can be slower because of the use of large numbers for z.

2. Comparison of data

The aim of this section is to assess whether Wagener's method delivers similar results as `sasmodels`.

For mono and polydispersity, we choose the following models for the Wagener/`sasmodels` comparison:

Table 2: Models used for Wagener/SasView comparison

$P(q)$ & $F^2(q)$ with Wagener	$I(q)$ & $F^2(q)$ with `sasmodels`	Parameters (Sasview[2])
Sphere	Sphere	$R = 1$
Biaxial Ellipsoid	Ellipsoid	$R_{\text{pol}} = 0.5, R_{\text{eq}} = 1$[3]
Cube	Parallelepiped and Rectangular Prism[4]	$a = b = c = 5$
SAR[5] Cylinder	Cylinder	$R = 1, L = 2$

We also have to precise the number of terms kept in the series for Wagener's models.

Table 3: Values of $n_{\max}$

$P(q)$ & $F^2(q)$ with Wagener	$n_{\max}$
Sphere	70
Biaxial Ellipsoid	80
Cube	70
SAR Cylinder	70

Important note 2

In the plots of this section, you may notice that all curves start from zero. That is normal: Wagener didn't take into account the computation of the additive constant. The "`(transl)`" in the different captions means "translation".

Important note 3

For `sasmodels`, we also used the `background = 0` parameter in the `pars` dictionnary for the computations of $P(q)$.

[2]Wagener has several differences with SasView for the definitions of parameters, for instance a a variable in SasView often becomes a $a/2$ variable in Wagener's method. Also see the following note.

[3]For Wagener: $R_{\text{pol}} =: c, R_{\text{eq}} =: a$.

[4]We will show two benchmarks only when necessary. When one single plot/comparison/benchmark is shown for the "Cube" model, the results are the same for Cube VS Rectangular Prism and Cube VS Parallelepiped.

[5]Small Axial Ratios (`length/radius` < 3).

2.1. $F^2(q)$ comparison

Without polydispersity

For $F^2(q)$ without polydispersity, the results are convincing.

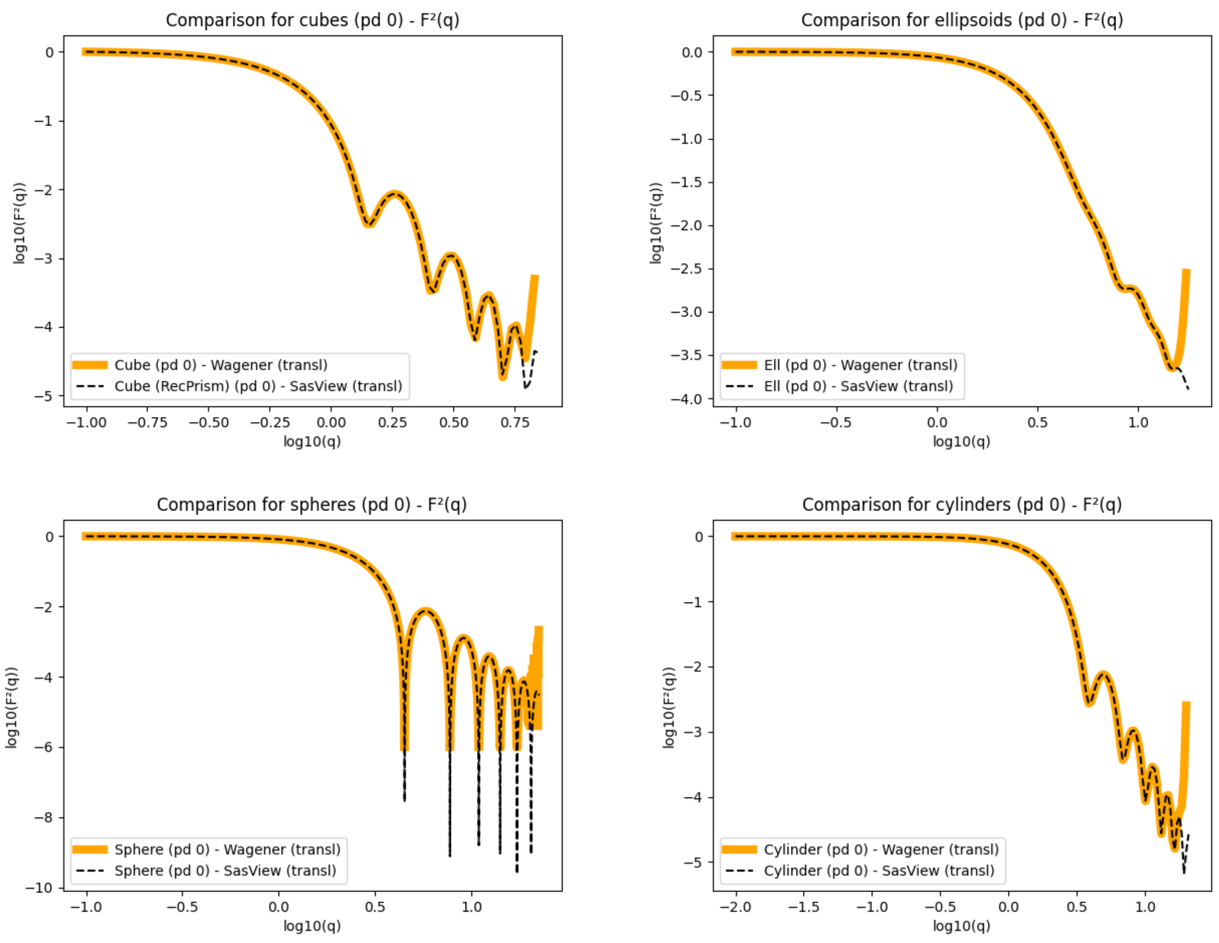


Figure 1: Wagener VS `sasmodels` comparison - $F^2(q)$ without polydispersity

Note that one can see the end of the first regime with $q_1 \cong 1$.

Possible improvement 1

According to the Supporting Information (page 65), it is possible to consider another even faster computation for Large Axial Ratios (LAR) cylinders.
The idea is to change a double `for` loop into a simple `for` loop. The code provided in the SI is however incomplete, but we do think that a `C[nmax]` series of coefficients exists such that this code for SAR:

```
// For each q:
long double Fq1 = 0; // it is F(q)^2
long double pown = 1;
// Rz2 and Lz2 constants defined previously
long double powR = -q*q*Rz2;
long double powL = -q*q*Lz2;
// nmax defined before, number of terms in the series
for (int n = 0; n <= nmax; n++){
    long double term_n = coeffn[n]*pown;
    long double sum_m = 0;
    long double powm = 1;
    for (int m = 0; m <= nmax; m++){
        sum_m += coeffmn[m][n+m]*powm; // uses n
        powm *= powR; // just a faster way to compute a pow
    }
    Fq1 += term_n * sum_m;
    pown *= powL;
}
```

becomes this code without the double `for` loop for LAR:

```
// For each q:
long double Fq1r = 0;
long double Fq1l = 0;
long double pown = 1;
// Rz2 and Lz2 constants defined previously
long double powR = -q*q*Rz2;
long double powL = -q*q*Lz2;
// ffl[] and ffr[] computed before
for (int n = 0; n <= nmax; n++){
    Fq1r += C[n]*pown*ffr[n]; // UNKNOWN C[n] <=============
    pown *= powR;
}
pown = 1;
for (int n = 0; n <= nmax; n++){
    Fq1l += (1/(2.*n+1))*pown*ffl[n];
    pown *= powL;
}
// The result used is Fq1 = Fq1r*Fq1l
```

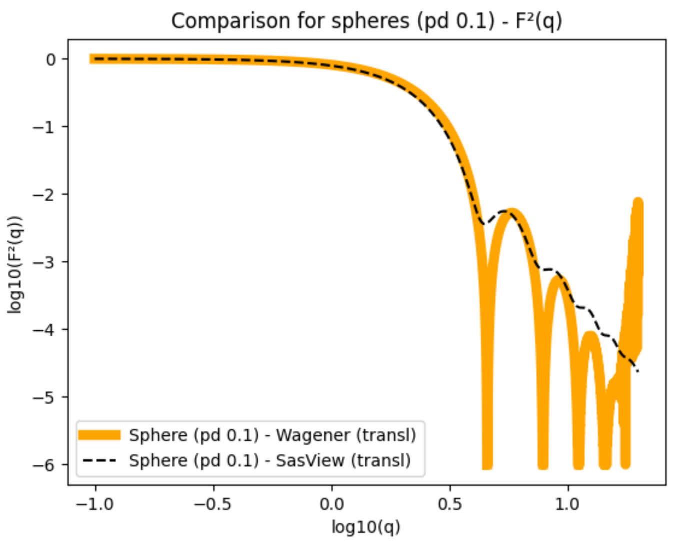
Later in the text, we will provide some time comparisons with this second model of cylinders[a], even if we don't know the `C[n]` coefficients.

[a] where $L/R \geq 3$, L for lenght et R for radius.

With polydispersity

For $F^2(q)$ with polydispersity, the distribution chosen for Wagener's computation is supposed to be the **Schulz-Zimm** distribution[6].

The results are not convincing, but this is not simple to show for all models.
The definition of polydispersities in Wagener's SI isn't always clear. If we take a model with only one parameter (the sphere), we have the following benchmark with a σ_{sphere} of 0.1:



For Wagener's plot, we notice that the curve is going down to zero for several points. Note that this defect is not generated by our `C` code. The supposed curve in Wagener's SI is the following[7], with in black Regime I and in blue Regime II:

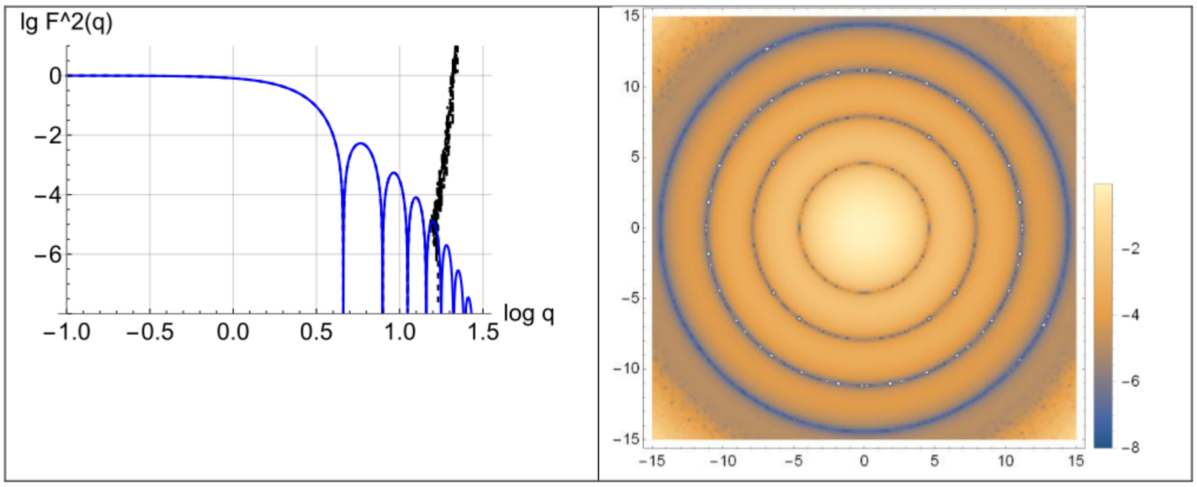


Fig. S4: Scattering amplitude of polydisperse spheres.

The comparison becomes difficult for other models if we do not exactly know the definitions of polydispersities. For instance and with `sasmodels`, for the cube, are we supposed to use the

[6]See SI, page 19.
[7]See SI, page 30.

Parallelepiped with three different and independent polydispersities for a, b, c, or the Rectangular Prism with $b/a = c/a = 1$ and one single polydispersity for a ? We tried with the two cases, the results are not convincing in both.

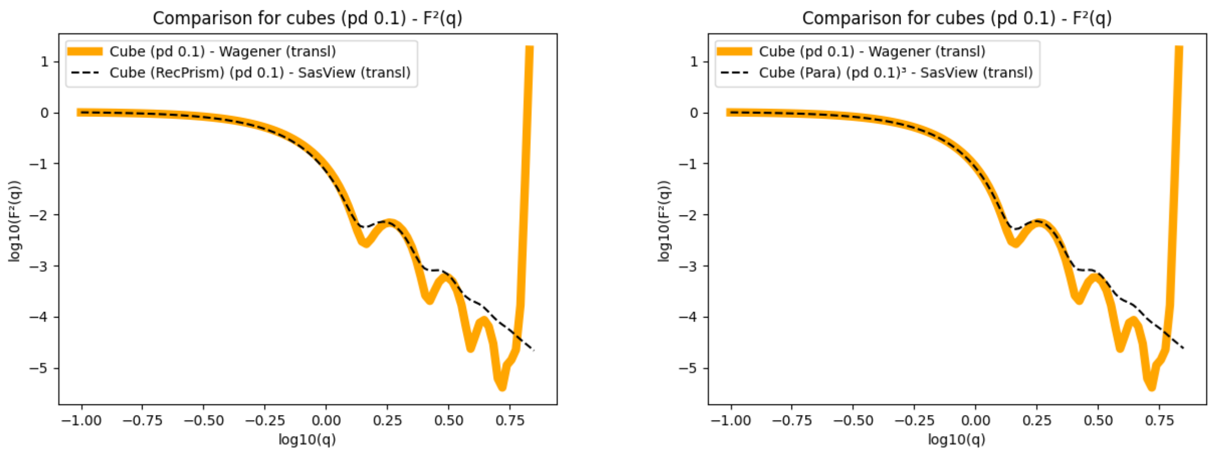


Figure 2: Wagener VS `sasmodels` Rectangular Prism (left) and Parallelepiped (Right)

Idem, for cylinders (with independent L and R) and for ellipsoids (with independent R_{pol} and R_{eq}), we have:

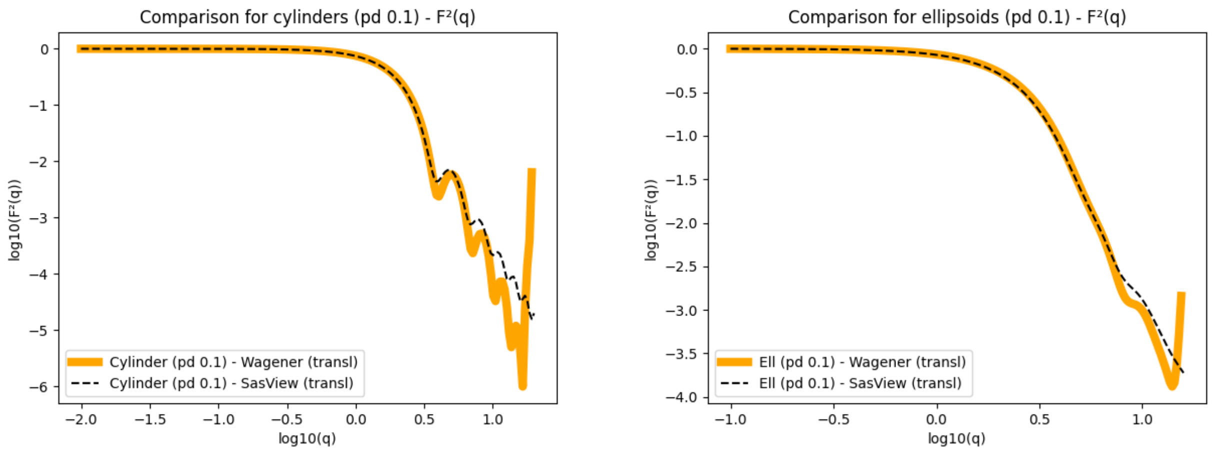


Figure 3: Wagener VS `sasmodels` Cylinder (left) and Ellipsoid (Right)

The use of polydispersity for $F^2(q)$ within Wagener's method is ambiguous.

Important note 4

For the time comparisons made later with computations of $F^2(q)$, we decided to only compare Wagener and SasView **monodisperse** models.

2.2. $I(q)$ VS $P(q)$ comparison

Important note 5

The following results compare:

- Wagener's computations of $P(q)$ with **Regime I**

with

- `sasmodels` computations of $I(q)$.

Without polydispersity

Without polydispersity, the results are convincing.

Note that after the end of Regime I, some q points have an **undefined** $P(q)$. In fact for these points, $P(q)$ is defined but **negative** and cannot be plotted on a log-log plot.

Due to the manipulation of `long double` values in `C`, it was difficult to have a $\sigma = 0$ value for $P(q)$ for cubes with good results. We decided to present a result where $\sigma = 10^{-10}$.

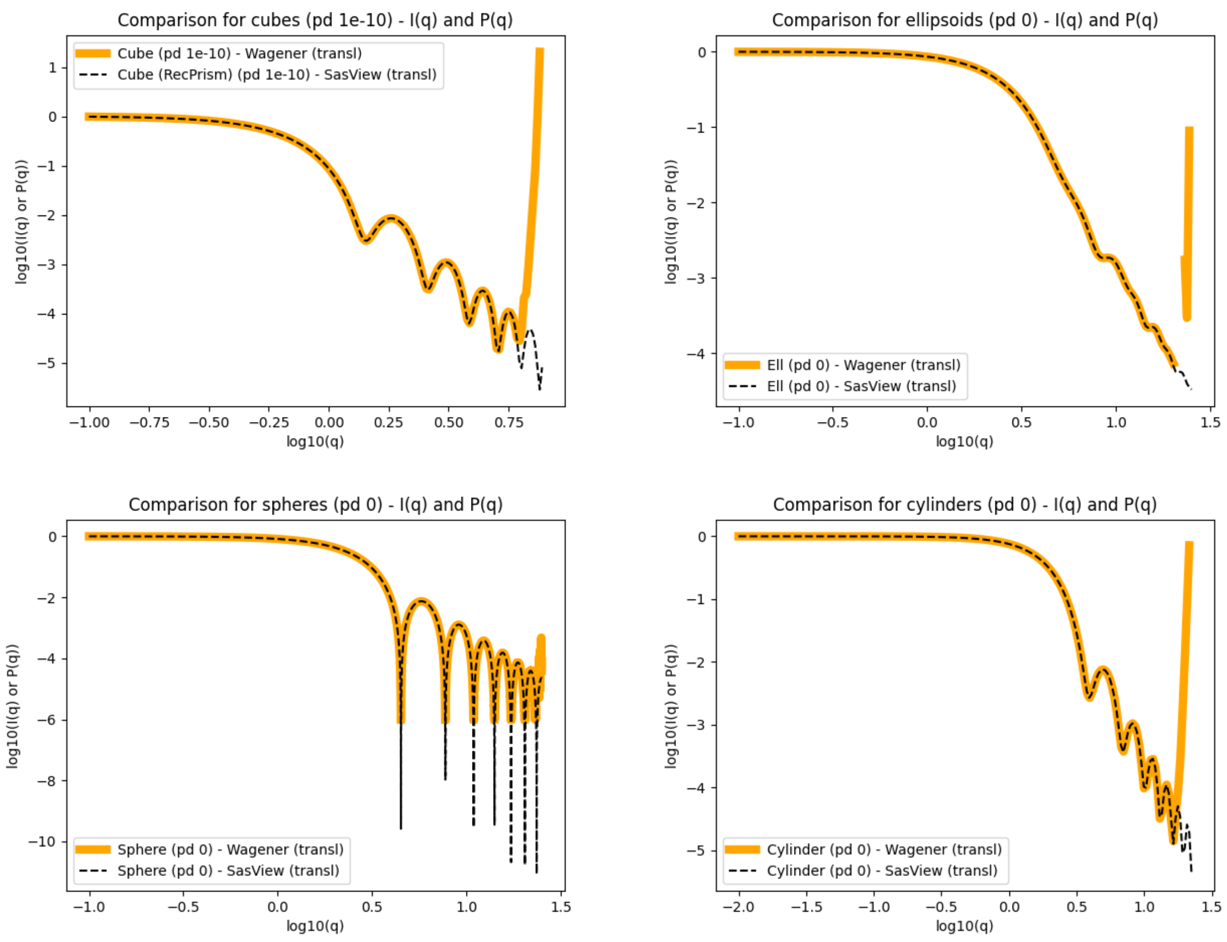


Figure 4: Wagener VS `sasmodels` comparison - $P(q)$ (Wagener) & $I(q)$ (`sasmodels`) without polydispersity

With polydispersity

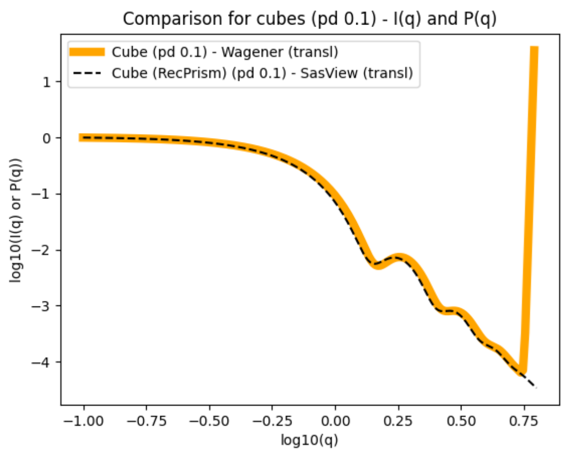
For $P(q)$ with polydispersity, the distribution chosen for Wagener's computation is supposed to be the **Schulz-Zimm** distribution.

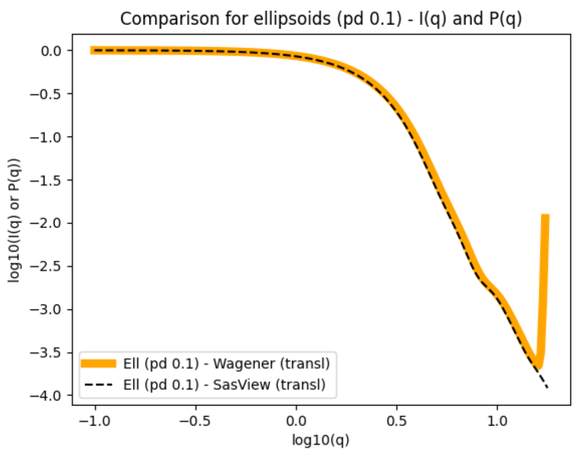
Important note 6

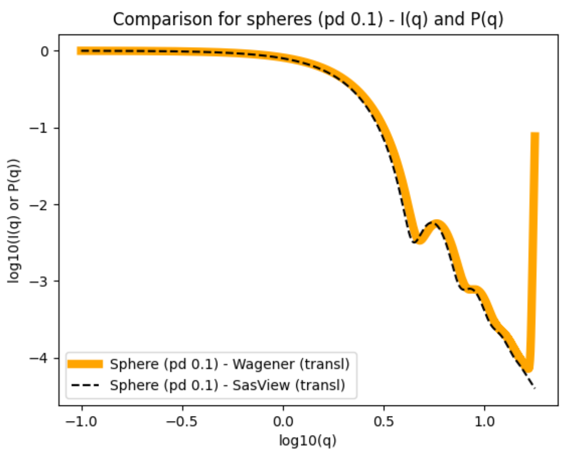
The parameters kept for the polydispersity in SasView for the comparison are:

- polydispersed values of R for spheres,
- polydispersed independent values of R_{pol} and R_{eq} for biaxial ellipsoids (in other words, $\frac{R_{\text{pol}}}{R_{\text{eq}}}$ is **not** constant for all ellipsoids),
- polydispersed cubes ($a = b = c$ with polydispersed a),
- polydispersed independent values of L and R for small axial ratios cylinder.

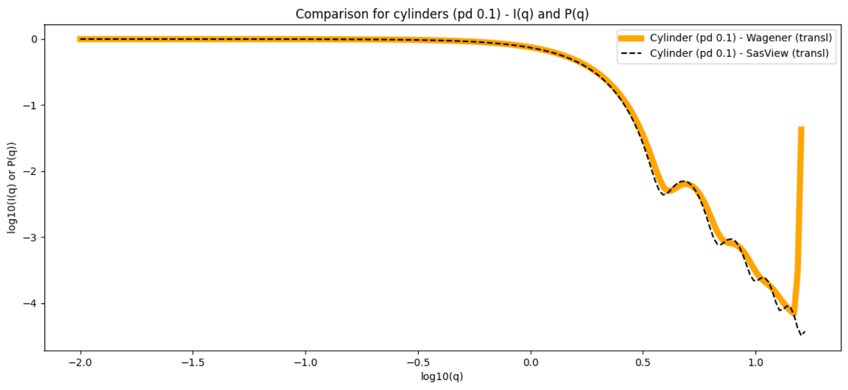
With polydispersity, the results are interesting: they include an unexpected shift.







The cylinder is the only model with a remarkable difference.



Important note 7

For the time comparisons made later with computations of $P(q)$ and $I(q)$, we decided to compare Wagener and SasView **monodisperse and polydisperse** models.

3. Comparison of time

This section presents some results about time comparisons, and proposes two benchmarks, the first one is naive, and the second one is less naive.

Important note 8

A short reminder. We are only benchmarking in this paper:

- $P(q)$ with mono and polydispersity,
- $F^2(q)$ with monodispersity.

Also note that `sasmodels` can use $F^2(q)$ to compute $I(q)$. In the less naive benchmark, we **won't compare** the time of computations of $I(q)$ and $P(q)$, we will only compare the times of computations of $F^2(q)$ ($F^2(q)$ Wagener VS $F^2(q)$ `sasmodels`).

3.1. Naive benchmark

This first benchmark is called naive because Wagener and `sasmodels` are **NOT** called in the same way.

☞ For Wagener and for each model `modelname.c`, we decided in this first benchmark to:

1. create a `modelname.so` library by compiling `modelname.c`,
2. trigger the chronometer,
3. use the `ctypes` library to run the compiled code and to compute the pattern,
4. stop the chronometer.

☞ For `sasmodels`, we decided for each model to:

1. create the `kernel` and `pars` objects
2. trigger the chronometer,
3. call `sasmodels call_Fq(kernel, pars)` or `sasmodels call_kernel(kernel, pars)`,
4. stop the chronometer.

Note 2

In `sasmodels`, the `C` code is called for **every** q. For Wagener, the `C` code is called only one time.
To make a fair comparison, the idea would be to call Wagener's code for each q or to inject Wagener's code into `sasmodels`.
In the following, the less naive benchmark section will use the second option.

3.1.1. $F^2(q)$

For $F^2(q)$, we have the following results[8]:

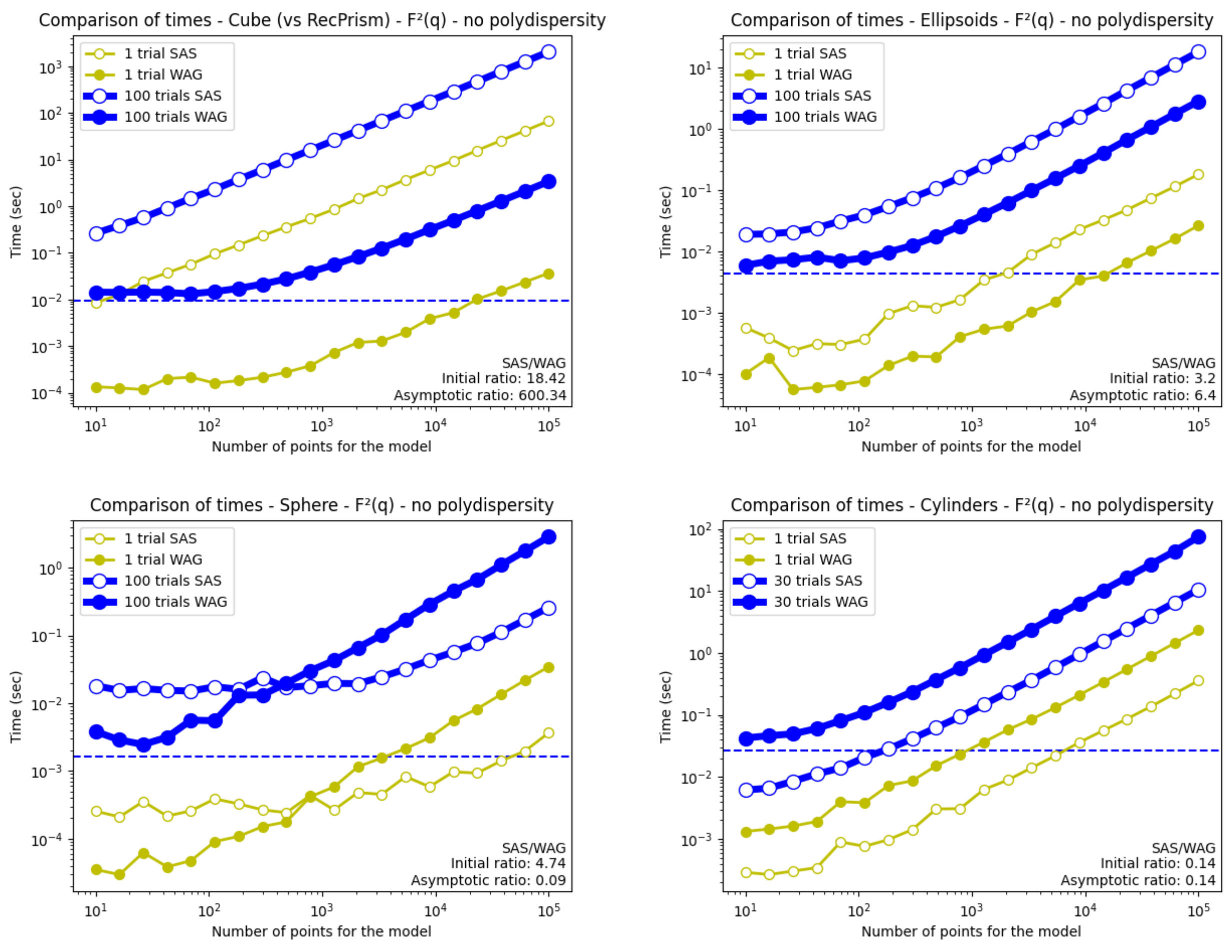


Figure 5: Naive benchmark for $F^2(q)$ with monodispersity. The straight lines in blue represent the initial costs for the computations of q-independent coefficients with the $n_{\max}$ values given in the Table 3, for 100 (or 30) trials.

The "100 trials" curves times are simply computed by making 100 classical calculations. In other words, the two pseudo-algorithms presented here become:

1. create what is useful,
2. **trigger** the chronometer,
3. for $i = 1, 2, ..., 100$, do what you did for the point 3. of the "1 trial" case,
4. **stop** the chronometer.

The `SAS/WAG` ratio is simply $\frac{\text{Time taken by SasView}}{\text{Time taken by Wagener}}$. The `Initial ratio` is for the first two blue points, the `Asymptotic ratio` is for the last two blue points.

Important note 9

Our Wagener code creates and prints data files in the `data` folder. The time benchmarks **always** disable this printing, with `printing = false` in the code.

[8]For cubes: Cube (Wagener) VS Rectangular Prism (`sasmodels`). Rectangular Prism will also be used for the $P(q)$ VS $I(q)$ time comparison.

Possible improvement 2

With LAR cylinders, we would possibly have the following result:

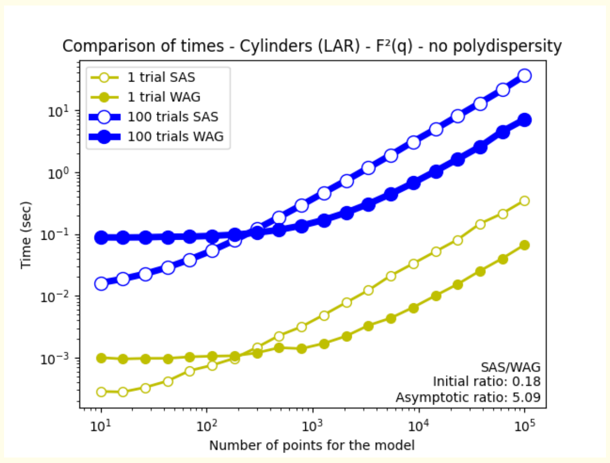


Figure 6: Potential gain of time with LAR cylinders

In the next page, we propose another possible improvement.

Possible improvement 3

In this first naive benchmark, you can note that, **for monodispersity**, it is possible to store the q-independent factors once and for all. That means that the program can avoid to compute q-independent coefficients if they are hard coded in the program.
This seems to slightly reduce the size of the Regime I, but multiplies the gain of time for small numbers of points, making the blue-Wagener curve a straight line and increasing the Initial Ratio.

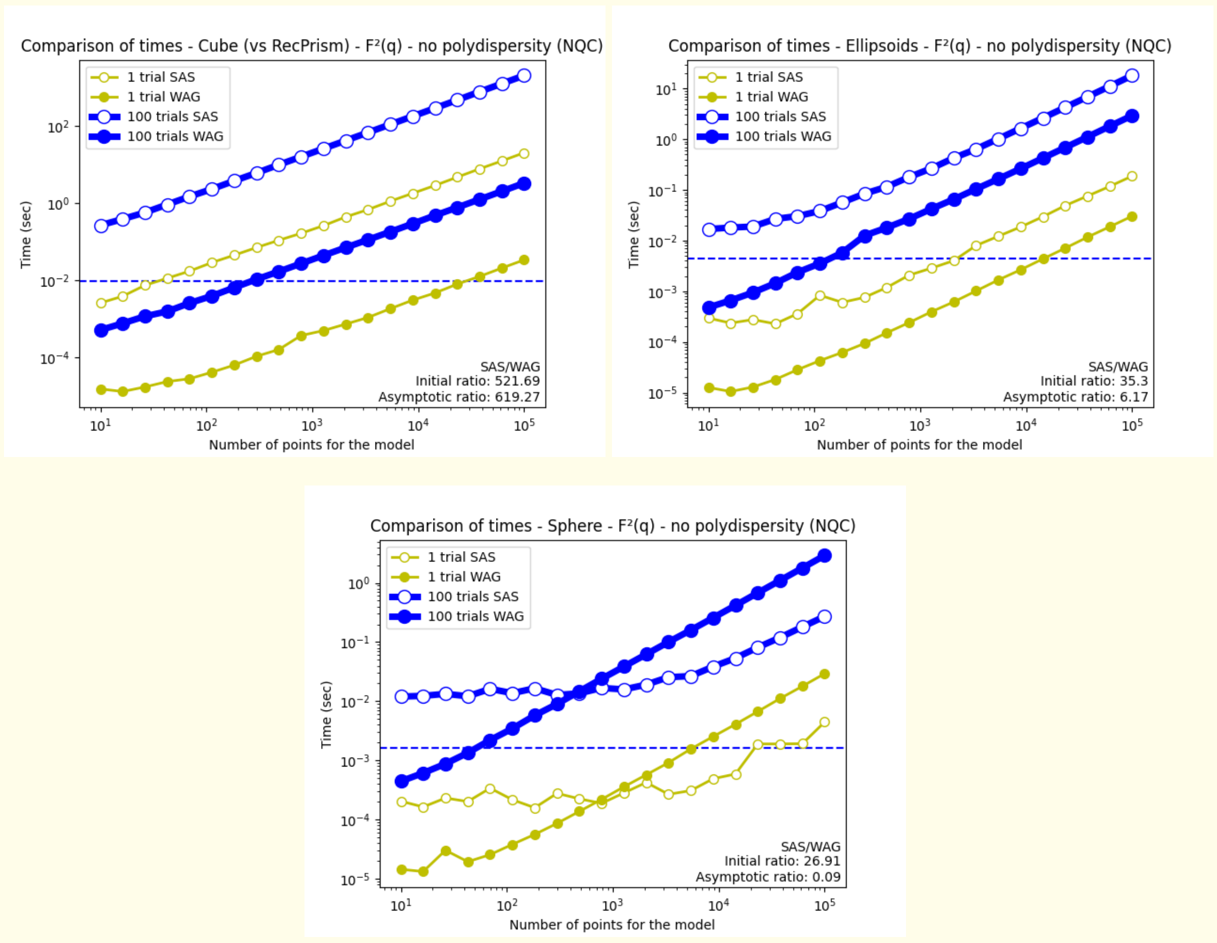


Figure 7: First benchmark for $F^2(q)$ with monodispersity, with q-independent coefficients hard coded in the program.

We didn't make this benchmark for cylinders, which need $71 \times 71 = 5041$ coefficients stored. The other models only needed $n_{\max} + 1 = 71$ or 81 coefficients (see Table 3).
It is also possible to use this idea for monodisperse $P(q)$.

3.1.2. $P(q)$

For $P(q)$ and $I(q)$ with mono and polydispersity, we have the results on the next page.

Note 3

Even if the data benchmark for cubes with **mono**dispersity was unsatisfying, we eventually decided to chronometer the "`pd = 0` way" for Cubes in the Wagener's method.

Here again, one can see the cost of the q-independent factors for several models.

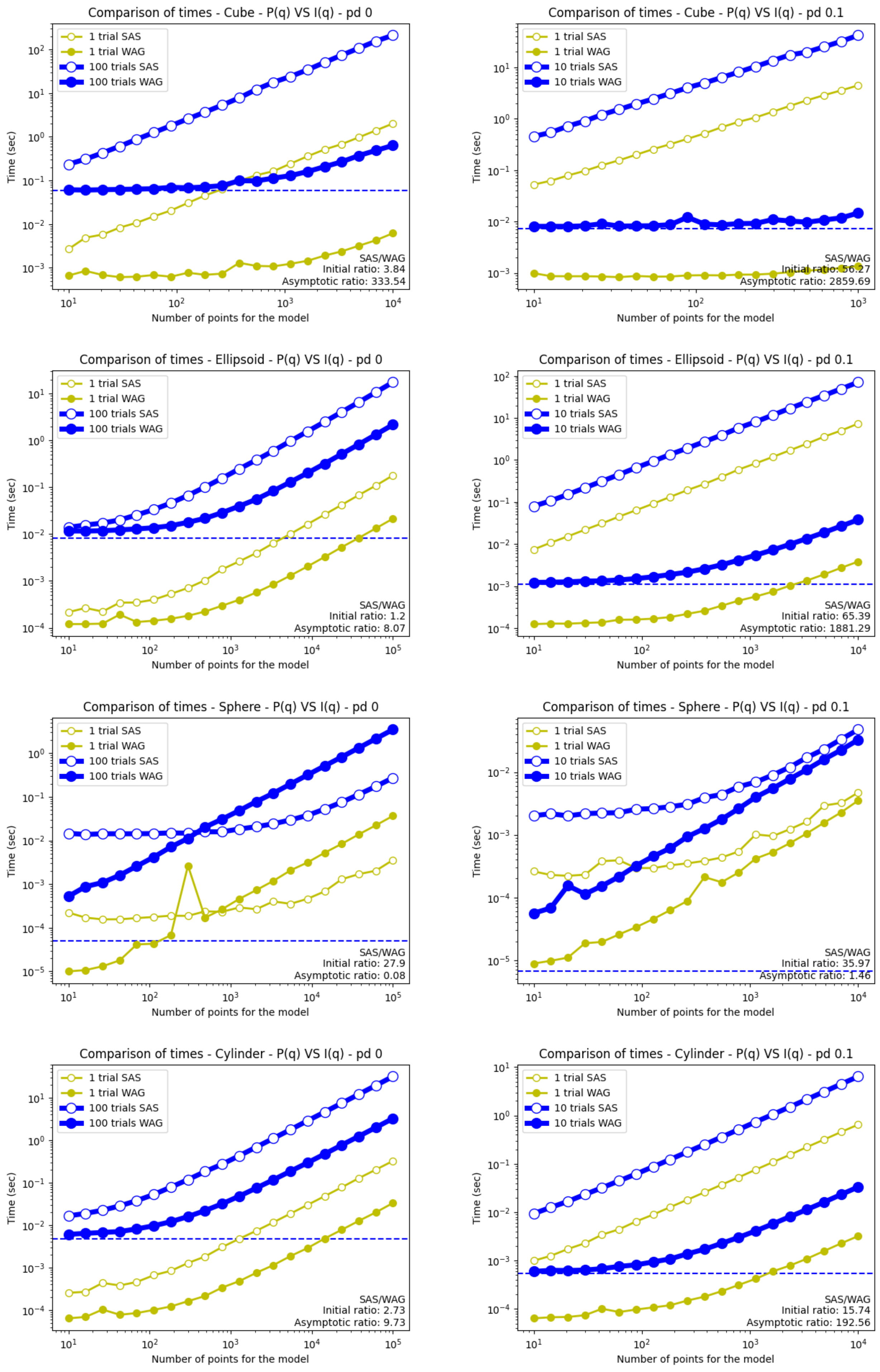


Figure 8: First benchmark which compares the time of computation of $P(q)$ (Wagener) and the time of computation of $I(q)$ (`sasmodels`), with and without polydispersity.

3.2. Less naive benchmark

This less naive benchmark was proposed by Paul Butler.
☞ For Wagener and for each model `modelname.c`, we decided in this second benchmark to:

1. trigger chronometer C_1,
2. compute q-independent factors a huge number of times,
3. stop chronometer C_1, we have (an average) Δt_1, time to compute q-independent factors,
4. trigger chronometer C_2,
5. call `sasmodels call_Fq(kernel, pars)` with Wagener's code in `call_Fq`,
6. stop chronometer C_2, we have Δt_2 (total theoretical computation time: $\Delta t_1 + \Delta t_2$).

Note 4

We noted in this remark an important point: we have to hard code q-independent coefficients in `call_Fq` for the comparison to be fair. We computed these coefficients and added by hand in each `modelname.c` of `sasmodels`:

```
long double coeffs[] = {..., ..., ...};
```

to run Wagener's code in `sasmodels`. This `coeffs[]` array is different for each model and each parameter of polydispersity σ[a].

[a]In this less naive benchmark, $\sigma = 0$.

Important note 10

Actually, we tried two slightly different approaches:

- a first one, where we used:
  `long double coeffs[nmax + 1]; coeffs[0] = ...; coeffs[1] = ...; ...`
- a second one, where we used: `long double coeffs[] = {..., ..., ...};`

The second approach is faster for high q, as one can see it on the following plots.

☞ For `sasmodels`, we still:

1. create the `kernel` and `pars` objects,
2. trigger the chronometer,
3. call `sasmodels call_Fq(kernel, pars)`,
4. stop the chronometer.

Important note 11

We are only benchmarking monodisperse $F^2(q)$ in this **Less naive benchmark** section. We won't benchmark times for the cylinder in this section (5041 coefficients to store).

Important note 12

Please note! In all the following benchmarks, the “`WAG in SAS`” curves plot what we called Δt_2 in the precedent pseudo-algorithm. $100 \times \Delta t_1$ is represented by the dotted blue line and is one hundred times the average cost of computing q-independent factors.

3.2.1. Wagener with `cytypes` VS Wagener with `sasmodels`

A first interesting thing to do is to compare Wagener's method with `ctypes` and Wagener's method with `sasmodels`.

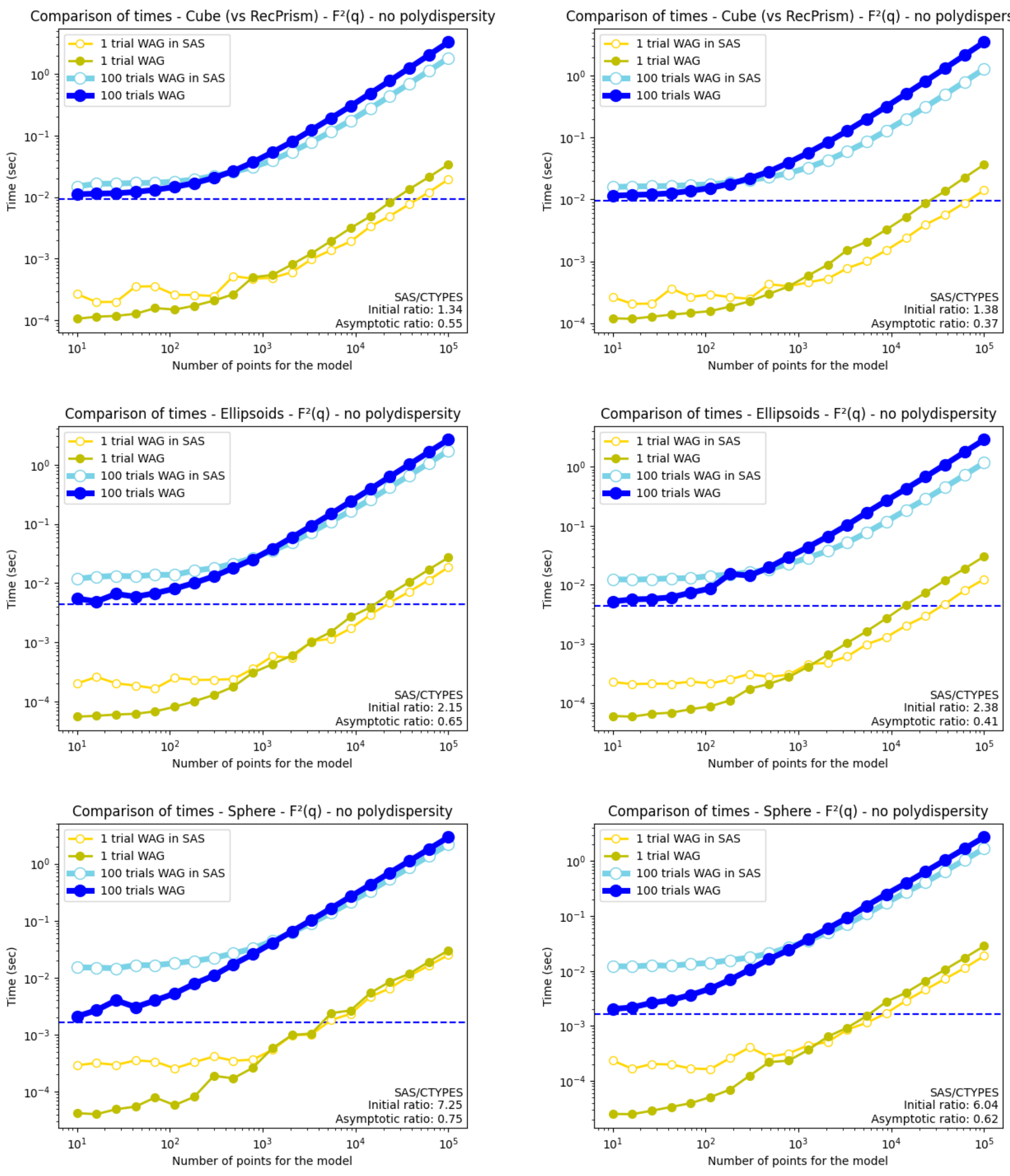


Figure 9: Comparison of times of computation of $F^2(q)$ (Wagener with `ctypes` VS Wagener launched with `sasmodels`). On the left, the first approach, on the right, the second one. `SAS/CTYPES` is $\frac{\text{Time taken by Wagener (}\texttt{sasmodels}\text{)}}{\text{Time taken by Wagener (}\texttt{ctypes}\text{)}} = \frac{\texttt{100 trials WAG in SAS}}{\texttt{100 trials WAG}}$.

Note 5

We also provide the time comparisons of Wagener `ctypes` VS Wagener `sasmodels` using the idea of this note (q-independent factors stored once and for all).

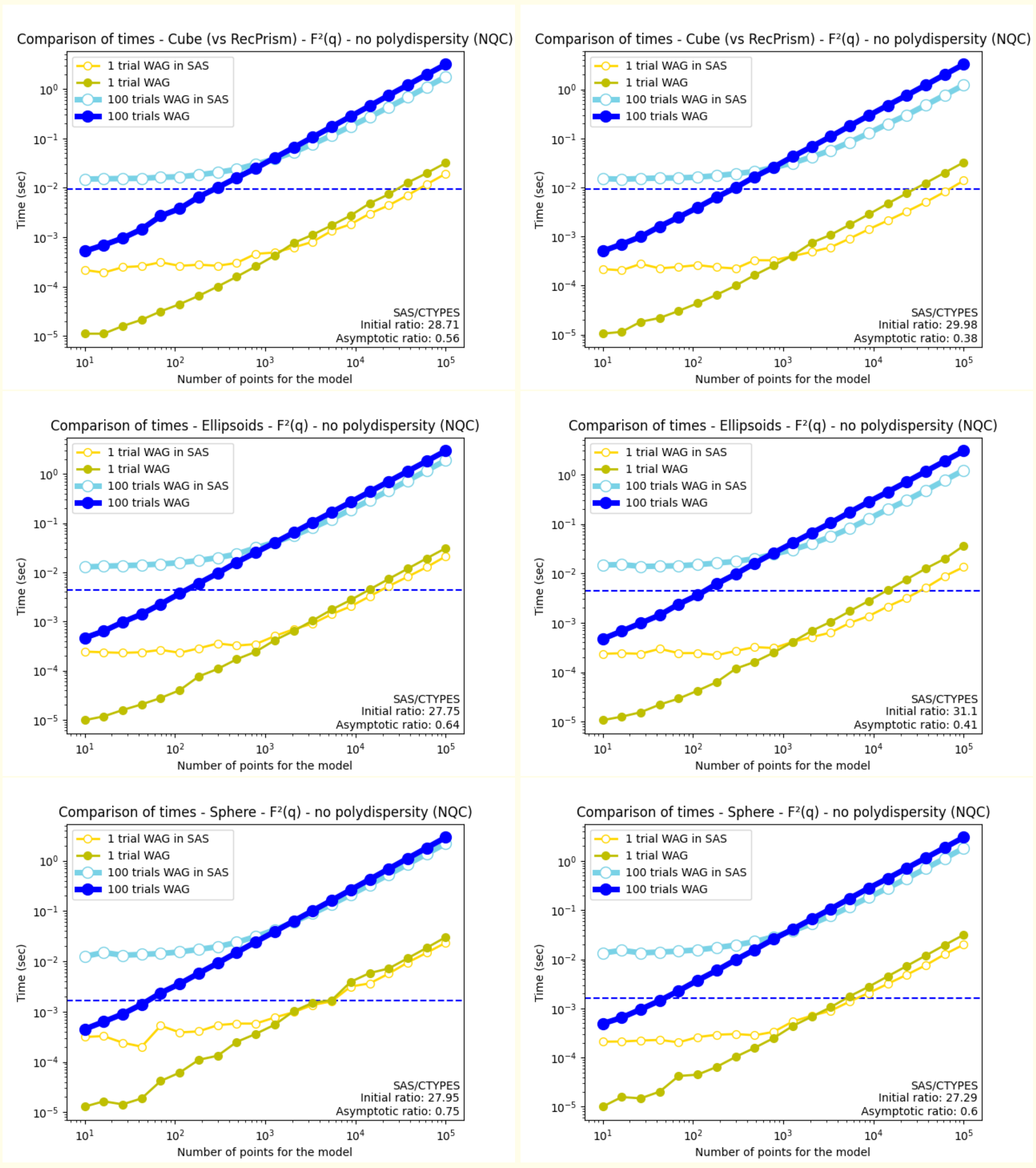


Figure 10: Comparison of times of computations of $F^2(q)$ (Wagener with `ctypes` VS Wagener launched with `sasmodels`). On the left, the first approach, on the right, the second one. `SAS/CTYPES` is $\frac{\text{Time taken by Wagener (\texttt{sasmodels})}}{\text{Time taken by Wagener (\texttt{ctypes})}} = \frac{\texttt{100 trials WAG in SAS}}{\texttt{100 trials WAG}}$.

One can see that launching `sasmodels` has a cost of $\cong 1.3 \times 10^{-2}$ s for 100 trials.
Nevertheless, `sasmodels` has a better use of `C` for high values of q.
In the following, we will keep the second approach.

3.2.2. Wagener with `sasmodels` VS (SasView with) `sasmodels`

We keep the second approach and we have the following results:

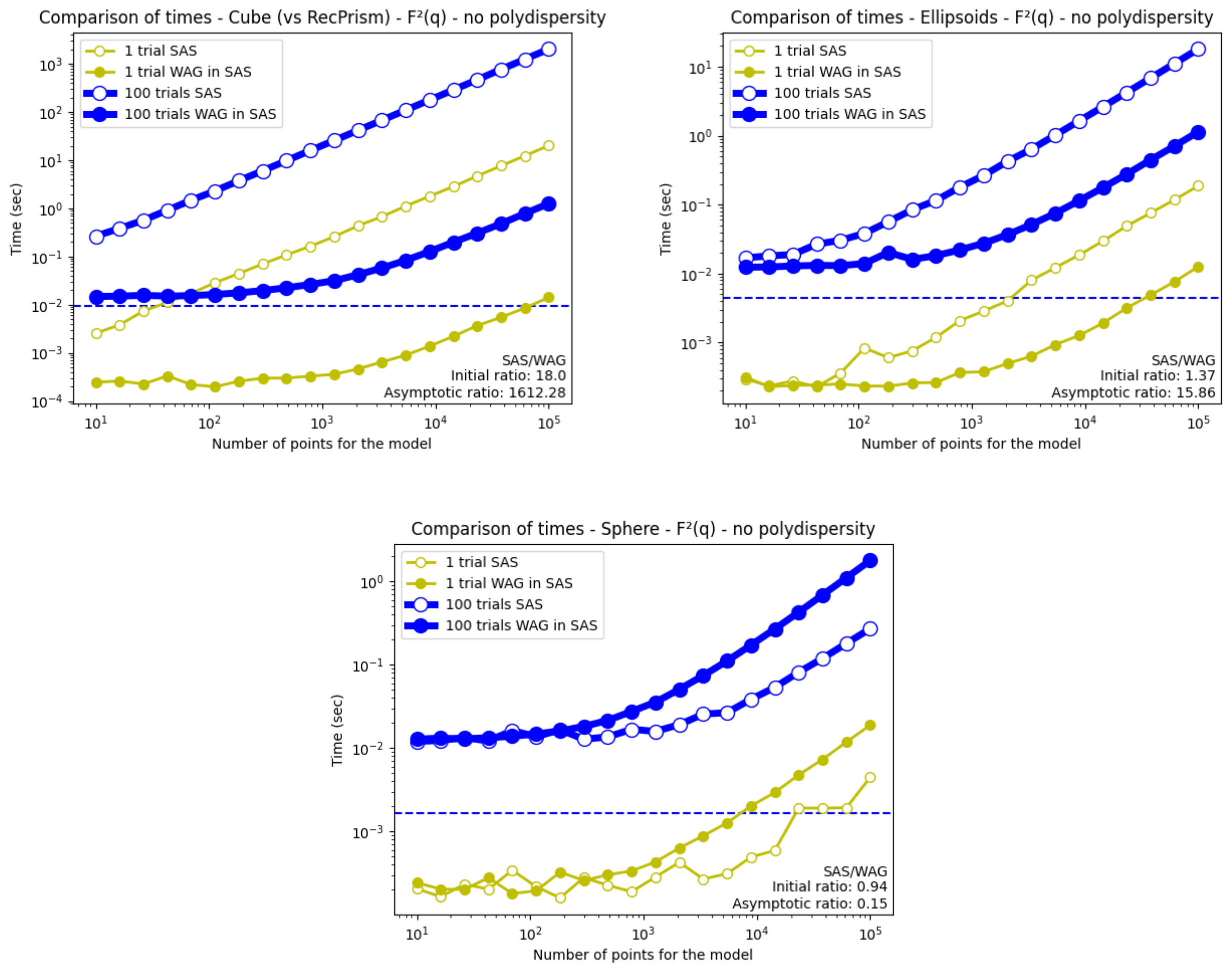


Figure 11: Comparison of times of computations of $F^2(q)$ (Wagener with `sasmodels` VS SasView with `sasmodels`). The dotted blue line still indicates the cost of computation of q-independent factors.

A. How to use the code provided

The code provided is written in `C` (for the models) and in `Python` (for the benchmarks). Keep in mind that this is a working code.

All the following instructions **have** to be typed in the main directory.

Note 6

The following instructions are given for `Linux`.

A.1. Test a model

To test a model, one can simply type :

```
make model; ./model_test # Pseudo-command!
```

The models available are:

```
make cube; ./cube_test
make ell; ./ell_test
make sphere; ./sphere_test
make cyl; ./cylinder_test
make lcyl; ./lcylinder_test # LAR cylinders
```

The different programs will launch GNUPlot to plot $P(q)$ and $F^2(q)$ with the parameters hardcoded in the correspondent `main`.

An important part of the code is commented (for time benchmarks) and can compute Regime III (and sometimes for simple cases Regime II).

Important note 13

Some compilation commands can include a special option: `-lprofiler`, in order to use `GPerfTools`. This option can simply be removed if the developer doesn't want to use profiling. A tutorial explaining profiling with `GPerfTools` can however be found here.

Executables and datasets generated can be erased with :

```
make clean;
```

A.2. Launch a data benchmark

Note 7

In the two following subsections, you have to install `sasmodels` to run the different programs.

To launch the data benchmark, one has to compile the different `C` libraries used by `Python` to launch Wagener's code. The libraries are stored in the `lib` directory and can be created with:

```
make xamodel; # Pseudo-command!
```

The five models available are:

```
make xacube
make xaell
make xasphere
make xacyl
make xalcyl # LAR cylinders
```

A general command creates the five libraries:

```
make xaxa
```

Data benchmark is launched with:

```
python ./versus/compare_data.py option pd
```

Where `option` is one of the following options:

Table 4: Options for `compare_data.py`

Option	Result
`pspherepd`	$P(q)$ (Wagener) VS $I(q)$ (`sasmodels`) for Spheres
`fspherepd`	$F^2(q)$ (Wagener) VS $F^2(q)$ (`sasmodels`) for Spheres
`pcubepd`	$P(q)$ (Wagener) VS $I(q)$ (`sasmodels`) for Cubes
`fcubepd`	$F^2(q)$ (Wagener) VS $F^2(q)$ (`sasmodels`) for Cubes
`pcylpd`	$P(q)$ (Wagener) VS $I(q)$ (`sasmodels`) for Cylinders
`fcylpd`	$F^2(q)$ (Wagener) VS $F^2(q)$ (`sasmodels`) for Cylinders
`pellpd`	$P(q)$ (Wagener) VS $I(q)$ (`sasmodels`) for Ellipsoids
`fellpd`	$F^2(q)$ (Wagener) VS $F^2(q)$ (`sasmodels`) for Ellipsoids
`flcylpd`	$F^2(q)$ (Wagener) VS $F^2(q)$ (`sasmodels`) for LAR cylinders.

The `pd` value is the wanted polydispersity value.
The parameters used are referenced in the Tables 2 and 3.
For instance, to plot $F^2(q)$ (Wagener) VS $F^2(q)$ (`sasmodels`) for Cubes with a polydispersity of 0.2, one can type:

```
make xaxa; python versus/compare_date.py fcubepd 0.2
```

A.3. Launch a time benchmark

To launch a time benchmark, one has to type:

```
make xaxa; python versus/compare_time.py
```

The benchmarks launched are determined by the uncommented lines after

```
if __name__=='__main__':
```

in `compare_time.py`.
Attention! Computations can take some time.

B. List of Figures

1. Wagener VS `sasmodels` comparison - $F^2(q)$ without polydispersity 5
2. Wagener VS `sasmodels` Rectangular Prism (left) and Parallelepiped (Right) . . 8
3. Wagener VS `sasmodels` Cylinder (left) and Ellipsoid (Right) 8
4. Wagener VS `sasmodels` comparison - $P(q)$ (Wagener) & $I(q)$ (`sasmodels`) without polydispersity 9
5. Naive benchmark for $F^2(q)$ with monodispersity. The straight lines in blue represent the initial costs for the computations of q-independent coefficients with the $n_{\max}$ values given in the Table 3, for 100 (or 30) trials. 13
6. Potential gain of time with LAR cylinders 14
7. First benchmark for $F^2(q)$ with monodispersity, with q-independent coefficients hard coded in the program. 15
8. First benchmark which compares the time of computation of $P(q)$ (Wagener) and the time of computation of $I(q)$ (`sasmodels`), with and without polydispersity. . 16
9. Comparison of times of computation of $F^2(q)$ (Wagener with `ctypes` VS Wagener launched with `sasmodels`). On the left, the first approach, on the right, the second one. `SAS/CTYPES` is $\frac{\text{Time taken by Wagener (sasmodels)}}{\text{Time taken by Wagener (ctypes)}} = \frac{\texttt{100 trials WAG in SAS}}{\texttt{100 trials WAG}}$. . . 18
10. Comparison of times of computations of $F^2(q)$ (Wagener with `ctypes` VS Wagener launched with `sasmodels`). On the left, the first approach, on the right, the second one. `SAS/CTYPES` is $\frac{\text{Time taken by Wagener (sasmodels)}}{\text{Time taken by Wagener (ctypes)}} = \frac{\texttt{100 trials WAG in SAS}}{\texttt{100 trials WAG}}$. 19
11. Comparison of times of computations of $F^2(q)$ (Wagener with `sasmodels` VS SasView with `sasmodels`). The dotted blue line still indicates the cost of computation of q-independent factors. 20