eSPC, an Online Data Analysis Platform for Molecular Biophysics

Raynals 1.0
User Documentation

August 2023

Table of Contents

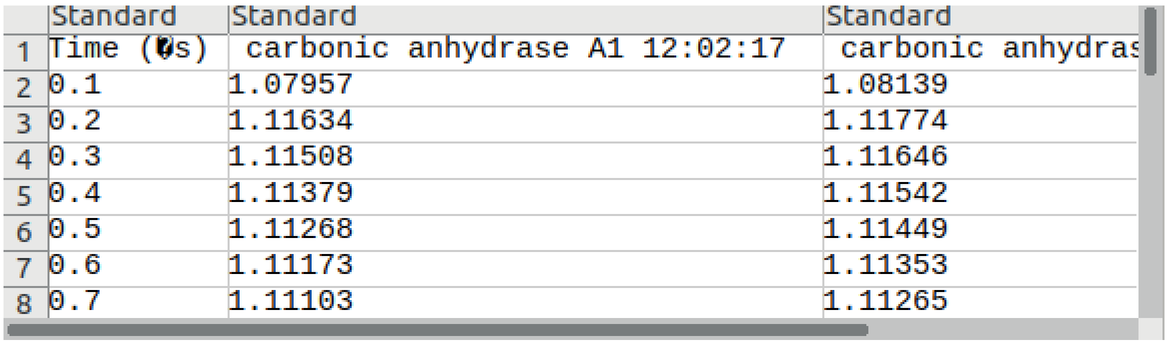
1. Load input

1.1. Input file (raw data)

Raynals accepts as input two types of files:

A) Comma-separated-values file (.csv)

The input file is a comma-separated-values (csv) file with headers. The first column contains time data, while the following columns contain the normalised second order autocorrelation ($g^{(2)}(\tau)$) data.



	Standard	Standard	Standard
1	Time (�s)	carbonic anhydrase A1 12:02:17	carbonic anhydras
2	0.1	1.07957	1.08139
3	0.2	1.11634	1.11774
4	0.3	1.11508	1.11646
5	0.4	1.11379	1.11542
6	0.5	1.11268	1.11449
7	0.6	1.11173	1.11353
8	0.7	1.11103	1.11265

Figure 1. Example of the CSV format required to load data in the Raynals app.

B) Compressed file (.zip or .7z)

The input file is a compressed file containing multiple Excel files (.xlsx). There should be one file per sample. Within each Excel file, the DLS data, comprising the time and autocorrelation values, should be present in the first two columns. The first column's header must be either 'tau' or 'time'. Sample names will be read from the Excel file names.

If you want to average acquisitions, use a directory structure similar to the one below.

```
└── pH_4_proteinX
    ├── Replicate 1
    │   ├── Acquisition 1
    │   │   ├── pH_4_proteinX - 1_acf.xlsx
    │   ├── Acquisition 2
    │   │   ├── pH_4_proteinX - 1_acf.xlsx
    │   ├── Acquisition 3
    │   │   ├── pH_4_proteinX - 1_acf.xlsx
    └── Replicate 2
        ├── Acquisition 1
```

```
│   ├── pH_4_proteinX - 2_acf.xlsx
├── Acquisition 2
│   ├── pH_4_proteinX - 2_acf.xlsx
├── Acquisition 3
│   ├── pH_4_proteinX - 2_acf.xlsx
```

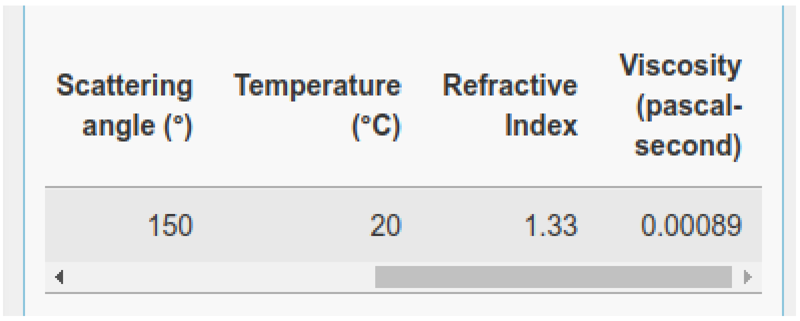
That directory structure will produce two final autocorrelation curves. Each of them consists of averaging three acquisitions.

General details

It’s important that the final autocorrelation data is one. Otherwise, it will be assumed that the DLS instrument exported $g^{(2)}(\tau) - 1$ and we’ll add one. The time units can be either microseconds or seconds.

1.2. Input parameters

To fit the autocorrelation data we need to use information about the DLS instrument setup and the experimental conditions.



Scattering angle (°)	Temperature (°C)	Refractive Index	Viscosity (pascal-second)
150	20	1.33	0.00089

Figure 2. Table with experimental and instrument parameters.

Instrument parameters: Laser wavelength (nm) and detection angle (degrees)
Experimental parameters: Temperature (ºC), refractive index and viscosity (pascal-second).

If you use a Wyatt Dynapro Plate Reader, the laser wavelength (nm) and detection angle (degrees) are 817 nm and 150°. For a Wyatt DynaPro NanoStar, use 658 nm and 90°. For a Nanotemper Prometheus Panta, use 405 nm and 145°. Regarding the temperature, refractive index and viscosity, we'll use as default 20°C, 1.33 (pure water) and 8.9e-4 pascal-second (pure water).

1.3. Filtering

Raw curves can be filtered by removing those with a lower intercept or a "bumpy" baseline to exclude samples with aggregates and/or buffers. Mathematically, the two filters can be expressed as:

Filter 1: select curves if $g^{(2)}(0) > P_1$ Equation 1

Filter 2: select curves if $g^{(2)}(P_2) < P_3$ Equation 2

where $g^{(2)}$ is the second-order autocorrelation data (raw curves), $g^{(2)}(0)$ is the value of $g^{(2)}$ at time zero, and $g^{(2)}(P_2)$ is the value of $g^{(2)}$ at a certain time 'P_2'. The parameter value P_1 from the first filter can be modified using the 'Filter by initial value' input field. The values of P_2 and P_3 are respectively modified using the 'Time limit' and 'Tolerance' input fields. Examples of these filters are presented in the User Guide from the Raynals app.

2. Fitting procedure

Raynals fitting is based on a non-parametric distribution of decay rates. The data we actually fit is the first order correlation function $g^{(1)}(\tau)$ which is related to the normalised second order correlation function through the Siegert Equation[1]:

$$g^{(2)}(\tau) = 1 + \beta|g^{(1)}(\tau)|^2 \quad \text{Equation 3}$$

where β is the coherence factor that depends on the instrument and the scattering properties of macromolecules.

2.1. First order autocorrelation

To approximate β, we fit a polynomial of degree two to the DLS data at times shorter than five μs. This approach has been proven to work with both simulated and experimental data. Then, we calculate $g_1(\tau)$ by applying the Siegert Equation until the first occurrence of $g^{(2)}(\tau) < 1$.

2.2. Distribution of decay rates

$g^{(1)}(\tau)$ can be represented by an intensity-weighted integral over a distribution of decay rates $G(\Gamma)$[2]:

[1] Siegert, A. J. F. (1943). On the fluctuations in signals returned by many independently moving scatterers Report: Radiation laboratory, Massachusetts Institute of Technology.
[2] Xu, R. (2001). Particle characterization: light scattering methods (Vol. 13). Springer Science & Business Media.

$$g^{(1)}(\tau) = \int_0^{\infty} G(\Gamma)e^{(-\Gamma_{\tau})}d\Gamma \quad \text{Equation 4}$$

where G(Γ) is normalised such that

$$\int_0^{\infty} G(\Gamma)d\Gamma = 1 \qquad \text{Equation 5}$$

Each decay rate is associated to a certain diffusion coefficient according to the following Equation:

$$D(s, q) = 1 / (sq^2) \qquad \text{Equation 6}$$

where s is the inverse of the decay rate and q is the Bragg wave vector defined as:

$$q(\lambda, \eta, \theta) = 4\eta\pi\lambda^{-1}sin(\frac{\theta}{2}) \text{ Equation 7}$$

where λ, η, θ are respectively the wavelength of the incident light, the solvent refractive index and the detection angle. Finally, the diffusion factors (D) can be transformed to hydrodynamic radius (Rh):

$$Rh(D, T, \mu) = \frac{k_b T}{6\pi\mu D} \qquad \text{Equation 8}$$

where T and μ are respectively the temperature and viscosity, and k_b is the Boltzmann constant.

2.3. Tikhonov-Philips regularised inversion

Raynals fits the first-order autocorrelation data based on the Tinkohov regularised inversion[3,4,5]. For this purpose, we first obtain β by fitting a polynomial of degree two to the DLS data at times shorter than five μs. Then, we apply Equation 3 to calculate $g_1(\tau)$. Due to the square power in this Equation, $g_1(\tau)$ can be computed only when

[3] Phillips, David L. "A technique for the numerical solution of certain integral equations of the first kind." *Journal of the ACM (JACM)* 9.1 (1962): 84-97.

[4] Provencher, Stephen W. "CONTIN: a general purpose constrained regularization program for inverting noisy linear algebraic and integral equations." *Computer Physics Communications* 27.3 (1982): 229-242.

[5] Brown, Patrick H., Andrea Balbo, and Peter Schuck. "Using prior knowledge in the determination of macromolecular size-distributions by analytical ultracentrifugation." *Biomacromolecules* 8.6 (2007): 2011-2024.

$g^{(2)}(\tau) \geq 1$. Therefore, we only evaluate the data before the first occurrence of $g^{(2)}(\tau) < 1$.

After calculating $g^{(1)}(\tau)$, we discretize the decay rate space by using *n* (*n* = 200) points between 0.1 and 1e6 nm log spaced in the hydrodynamic radius scale.

The equation we need to fit becomes

$$g^{(1)}(\tau) = \sum_{i=1}^{200} c_i e^{(-\tau / s_i)} \qquad \text{Equation 9}$$

subject to the constraints

$$\forall i;\ c_i \geq 0, \sum_{i=1}^{200} c_i = 1 \qquad \text{Equation 10}$$

$$c_1 = c_{200} = 0 \qquad \text{Equation 11}$$

where c_i is the i-th contribution of the i-th inverse decay rate (s_i). Due to the ill-condition nature of the problem (infinite possible solutions), we need to add a regularisation term, so we solve simultaneously the following equations

$$\alpha \sum_{i=2}^{199} 2c_i - c_{i-1} - c_{i+1} = 0 \quad \text{Equation 12}$$

where α is a regularisation parameter controlling how close the relative contribution of each (inverse) decay rate should be to its neighbouring (inverse) decay rates. The whole set of linear equations is solved together using scipy non-negative least squares solver (docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.nnls).

An alternative way of describing the set of linear equations from Equation 9 and Equation 12 is that we need to find the vector of relative contributions (x) such that

$$x_\alpha = argmin\, ||Ax_\alpha - g^{(1)}(\tau)||_2^2 + \alpha||Lx_\alpha||_2^2 \qquad \text{Equation 13}$$

where A is the kernel matrix with values

$$a_{i,j} = exp(-\tau_i \gamma_j) \quad \text{Equation 14}$$

where *i* and *j* iterate over the lag time vector and decay rate vector, respectively. L is the second-order derivative matrix.

2.4. The L-curve criteria

One way of finding an adequate value of α is to apply the L-curve criteria[6]. This heuristic rule consists of plotting the logarithm of the residuals (fidelity term, $||Ax_{\alpha} - g^{(1)}(\tau)||_2^2$) against the logarithm of the norm of the regularised solution (penalty term, $\alpha||Lx_{\alpha}||_2^2$) for different values of α and selecting the value of α corresponding to the corner point of the L-shaped curve. This method balances the size of the regularised solution and the accuracy of the fit to the given data.

2.5. Selecting the values of α

To generate the L-curve we need to test a wide range of different values of α. In Raynals, a sequence of regularisation parameters (α) evenly spaced in a log scale are evaluated. This sequence is generated using the following formula:

$$\alpha_n = (5^{f(n)})^2 \qquad \text{Equation 15}$$

where *f(n)* depends on three parameters called 'start', 'stop', 'step' and is defined as

$$f(n) = start + n * step \,/\, n = \{0, 1, 2, \dots, \lfloor (stop - start)/step \rfloor\} \quad \text{Equation 16}$$

The 'start', 'stop' and 'step' values can be changed in the 'Analysis' box from the 'Analysis' Tab. The tested α values together with the corresponding fidelity and penalty terms can be downloaded in the 'Export' tab.

2.6. Finding the corner of the L-curve

To automatise the detection of the corner of the L-curve we used the triangle method proposed by Castellanos *et al.*, 2002[7]. Briefly, for each point (labelled 'A') in the log-log curve, we compute the angles created by all previous points (labelled 'B'), the point 'A', and the last point (labelled 'C') (angle BAC). Then, we select the point 'A' that has the smallest angle.

3. Fitting results

3.1. Fitted autocorrelation

[6] Hansen, Per Christian. "The L-curve and its use in the numerical treatment of inverse problems." (1999): 119-142.

[7] Castellanos, J. Longina, Susana Gómez, and Valia Guerra. "The triangle method for finding the corner of the L-curve." *Applied Numerical Mathematics* 43.4 (2002): 359-373.

The experimental second-order autocorrelation $g^{(2)}(\tau)$ is displayed as coloured dots, while the predicted second-order autocorrelation (based on the fitting of the first-order autocorrelation $g^{(1)}(\tau)$) is shown with black lines. The residuals of $g^{(2)}(\tau)$ can be used to filter the data (parameter ‘Residuals filter’).

3.2. Hydrodynamic radii distribution

The estimated distribution of hydrodynamic radii represents the intensity weighted contributions and can be visualised as an histogram, density plot, or grayscale coloured bar plot. The visualisation style can be changed using the ‘Plot type’ option in the ‘Analysis’ Tab. With the exception of the ‘Collapsed’ plot, the height of the y-axis is relative and can be determined by the ‘Relative’ height’ parameter.

3.3. Peak parameters

To retrieve the characteristic Rh the user must select certain intervals to analyse the Rh distribution (‘Peak selection’ box). Then, for each range of interest, a peak searching algorithm is used to find the highest peak. From this peak, we calculate the weighted harmonic mean[8] (or peak maximum), the total relative contributions to the intensity, the standard deviation and, assuming perfect non-interacting spheres in a non-absorbing medium, the mass (or volume) weighted contribution.

3.4. Regularisation terms

If the L-curve criteria was used to fit the data, the ‘optimal’ automatically selected regularisation parameters are available in the ‘Regularisation terms’ Table.

4. Simulation

All generated data is based on non-interacting spherical particles surrounded by a non-absorbing medium. The minimum and maximum allowed hydrodynamic radius are 0.09 and 10^6 nm.

4.1. Number distribution

This distribution is directly determined by the parameters ‘Population mean’, ‘Population sd’ and ‘Number of particles’. For each normally distributed population that we want to simulate (of a certain hydrodynamic radius), we should input its mean, standard deviation and the number of particles.

[8] Farkas, Natalia, and John A. Kramar. "Dynamic light scattering distributions by any means." *Journal of Nanoparticle Research* 23.5 (2021): 120.

4.2. Volume distribution

The volume distribution is derived from the number distribution by assuming that each particle is a sphere. Therefore, the volume depends on the cube of the Rh.

4.3. Intensity distribution

The intensity distribution is derived from the number distribution by assuming that each particle is a sphere and that the intensity scattered by each particle follows the Mie theory[9]. Also, particles do not interact with each other and the medium doesn't absorb any of the scattered light. The parameters 'angle of detection (°)', 'wavelength (nanometers)' and 'refractive index (unitless)' are required at this step. For small particles (size < wavelength/10), the amount of scattered light follows the Rayleigh Theory and scales to the sixth power of the radius[10]. The intensity values are calculated with the Mie python package[11].

4.4. Autocorrelation curves

To generate the autocorrelation curves we first transform each Rh into a diffusion coefficient by using the selected 'Temperature' and 'Viscosity', and then into a decay rate according to the Bragg wave vector (which depends on the detection angle). Finally, based on the relative contributions of each decay rate to the intensity distribution, we apply Eq. 9 (first order autocorrelation) and subsequently Eq. 3 to obtain the second order autocorrelation function. The value at which this curve crosses the y-axis is determined by the 'Intercept' parameter which can only go from 0 to 1. Gaussian error can be added based on the 'Gaussian error' parameter.

Contact details

For further assistance, please contact us:

📧 spc@embl-hamburg.de
📌EMBL (c/o DESY), Notkestrasse 85, Build. 25a, 22607 Hamburg, Germany

[9] Wiscombe, W. J. (1979). *Mie Scattering Calculations: Advances in Technique and Fast, Vector-speed Computer Codes* (No. NCAR/TN-140+STR). University Corporation for Atmospheric Research. doi:10.5065/D6ZP4414
[10] Seinfeld, John H., and Spyros N. Pandis. *Atmospheric chemistry and physics: from air pollution to climate change*. John Wiley & Sons, 2016.
[11] https://miepython.readthedocs.io

Packages

Raynals is possible thanks to:

R language: R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

R package shiny: Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie and Jonathan McPherson (2020). shiny: Web Application Framework for R. R package version 1.4.0.2. https://CRAN.R-project.org/package=shiny

R package viridis: Simon Garnier (2018). viridis: Default Color Maps from 'matplotlib'. R package version 0.5.1. https://CRAN.R-project.org/package=viridis

R package tidyverse: Wickham et al., (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686, https://doi.org/10.21105/joss.01686

R package pracma: Hans W. Borchers (2019). pracma: Practical Numerical Math Functions. R package version 2.2.9. https://CRAN.R-project.org/package=pracma

R package shinydashboard: Winston Chang and Barbara Borges Ribeiro (2018). shinydashboard: Create Dashboards with 'Shiny'. R package version 0.7.1. https://CRAN.R-project.org/package=shinydashboard

R package ggplot2: H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

R package reshape2: Hadley Wickham (2007). Reshaping Data with the reshape Package. Journal of Statistical Software, 21(12), 1-20. URL http://www.jstatsoft.org/v21/i12/.

R package tippy: John Coene (2018). tippy: Add Tooltips to 'R markdown' Documents or 'Shiny' Apps. R package version 0.0.1. https://CRAN.R-project.org/package=tippy

R package shinyalert: Pretty Popup Messages (Modals) in 'Shiny'. R package version 1.1. https://CRAN.R-project.org/package=shinyalert

R package plotly: C. Sievert. Interactive Web-Based Data Visualization with R, plotly, and shiny. Chapman and Hall/CRC Florida, 2020.

R package rhandsontable: Jonathan Owen (2018). rhandsontable: Interface to the 'Handsontable.js' Library. R package version 0.3.7. https://CRAN.R-project.org/package=rhandsontable

R package shinyjs: Dean Attali (2020). shinyjs: Easily Improve the User Experience of Your Shiny Apps in Seconds. R package version 1.1. https://CRAN.R-project.org/package=shinyjs

R package reticulate: Kevin Ushey, JJ Allaire and Yuan Tang (2020). reticulate: Interface to 'Python'. R package version 1.16. https://CRAN.R-project.org/package=reticulate

R package shinycssloaders: Andras Sali and Dean Attali (2020). shinycssloaders: Add CSS Loading Animations to 'shiny' Outputs. R package version 0.3. https://CRAN.R-project.org/package=shinycssloaders

R package stringr: Wickham H (2022). stringr: Simple, Consistent Wrappers for Common String Operations. R package version 1.4.1, https://CRAN.R-project.org/package=stringr.

R package Ciaro: Urbanek S, Horner J (2022). Cairo: R Graphics Device using Cairo Graphics Library for Creating High-Quality Bitmap (PNG, JPEG, TIFF), Vector (PDF, SVG, PostScript) and Display (X11 and Win32) Output. R package version 1.6-0, https://CRAN.R-project.org/package=Cairo.

R package svglite: Wickham H, Henry L, Pedersen T, Luciani T, Decorde M, Lise V (2022). svglite: An 'SVG' Graphics Device. R package version 2.1.0, https://CRAN.R-project.org/package=svglite.

R package RColorBrewer: Neuwirth E (2022). RColorBrewer: ColorBrewer Palettes. R package version 1.1-3, https://CRAN.R-project.org/package=RColorBrewer.

R package DT: Xie Y, Cheng J, Tan X (2022). DT: A Wrapper of the JavaScript Library 'DataTables'. R package version 0.25, https://CRAN.R-project.org/package=DT.

R package scales: Wickham H, Seidel D (2022). scales: Scale Functions for Visualization. R package version 1.2.1, https://CRAN.R-project.org/package=scales.

Python3.7 language: Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.

Python package numpy: Travis E, Oliphant. A guide to NumPy, USA: Trelgol Publishing, (2006). Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37

Python package pandas: Wes McKinney. Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56 (2010)

Python package scipy: Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17(3), 261-272.

Python package miepython: Prahl, S. "miepython v1. 3.0." (2017).