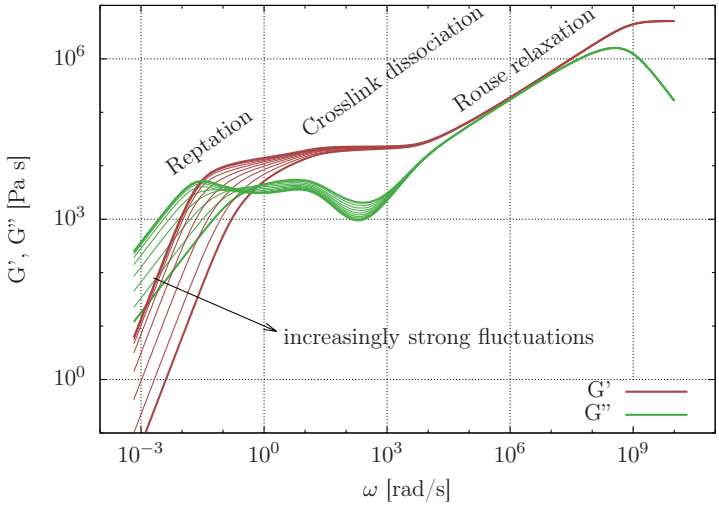
User Manual for Bombyx 1.0

Modelling of linear viscoelasticity

(Dated: March 18, 2019)



Contents

I. INTRODUCTION

Bombyx uses the sticky-reptation model (see Appendix VII) to calculate the elastic, $G'(\omega)$, and viscous, $G''(\omega)$, modulus as a function of the angular frequency, ω, and fit that to experimental data. From this fit, values and standard deviations for the following physical properties are extracted:

- τ_{S}: The effective sticker-dissociation time.
- Z_{S}: The effective number of stickers per chain.
- Z_{E}: The effective number of entanglements per chain.
- G_{E}: The elasticity modulus.
- α: Magnitude of contour-length fluctuations.

The physical meaning of these parameters are briefly discussed in Appendix VII; the emphasis of this manual is on the algorithm of curve fitting, and the workflow of running Bombyx to analyse experimental datasets.

For successful fitting and error analysis, numerical input setting have to be adjusted by the user of the software. Therefore, first some background on the fitting algorithm is given to show where these settings come in play. Next this user manual provides an installation guide, and refers to a demo (runtime: couple of minutes on a regular desktop computer) and the expected output of that demo. Finally, a demo of a typical workflow is provided to find the appropriate numerical settings for custom experimental data.

II. BACKGROUND: MARKOV CHAIN MONTE CARLO ALGORITHM

Extracting the physical parameter values is done using a conventional χ^2 definition of the fit quality, and their standard deviation is determined from the χ^2 landscape as detailed below.

The χ^2 definition is

$$\chi^2 \equiv \frac{1}{N_{\mathrm{dof}}} \sum_{i=1}^{N_{\mathrm{data}}} \frac{(\Delta G_i')^2}{G_{\mathrm{fit},i}'} + \frac{(\Delta G_i'')^2}{G_{\mathrm{fit},i}''}, \tag{1}$$

with the ΔG_i's the residual between model and experiment for data point $i = 1, 2, \ldots, N_{\mathrm{data}}$ with N_{data} the number of data points. Further, $N_{\mathrm{dof}} \equiv N_{\mathrm{data}} - N_{\mathrm{par}}$ are the degrees of freedom, with N_{par} the number of free fit parameters. The parameter values are optimised by minimising χ^2, and their standard deviations are obtained from a sample distribution of the parameter space near this optimum.

Close to the optimum, a quadratic dependence of χ^2 on the fit parameters p_i is assumed, with $i = 1, 2, \ldots, N_{\mathrm{par}}$, given by

$$\chi^2 = \chi^2_{\mathrm{bestfit}} + \frac{\sigma^2}{\sigma_i^2} \sum_{j=1}^{N_{\mathrm{par}}} (p_j - p_{j,\mathrm{bestfit}})^2. \tag{2}$$

Given a parameter set $\{p\}_j$ with fit quality χ^2_j, a new set $\{p\}_{j+1}$ with fit quality χ^2_{j+1} is proposed to add to the sample distribution. This proposition is accepted if $\chi^2_{j+1} < \chi^2_j$, but only accepted with probability $\exp(-(\chi^2_{j+1} - \chi^2_j)/2\sigma^2_\alpha)$ if otherwise. If σ_α equals σ, this Markov chain generates the distribution

$$P(\chi^2) \propto (\sigma^2)^{-N_{\mathrm{par}}/2} \exp\left(-\chi^2/2\sigma^2\right), \tag{3}$$

from which the mean parameter values and their standard deviations are straight-forwardly extracted.

While it is possible to adapt σ_α during the sampling algorithm such that it matches σ, the present sampling method, which resembles umbrella sampling, does not do this. Here, σ_α is chosen larger than σ so that the χ^2 landscape is oversampled. From the resulting distribution, data is then removed with a probability $\exp(-(\chi^2_{j+1} - \chi^2_{\mathrm{bestfit}})/2\sigma^2_\beta)$, such that the σ^2 of the new distribution obeys $\sigma^{-2} = \sigma_\alpha^{-2} + \sigma_\beta^{-2}$.

This algorithm generates a sample distribution by performing the steps:

1. Set $j = 1$ and initialise parameter values $\{p\}_j$ and calculate χ^2_j, with $\{p\}_j = \{\tau_{\mathrm{S}}, Z_{\mathrm{S}}, Z_{\mathrm{E}}, G_{\mathrm{E}}, \alpha\}_{\mathrm{j}}$.
2. Propose new parameter set $\{p\}_{j+1}$ and calculate χ^2_{j+1}.

3. Accept the new parameter set $\{p\}_{j+1}$ if $\chi^2_{j+1} < \chi^2_j$ or if a uniform deviate on the unit interval, u is sufficiently small $u < exp(-(\chi^2_{j+1} - \chi^2_j)/2\sigma^2_\alpha)$; otherwise, the new parameter set is rejected and $\{p_{j+1}\}$ and χ^2_{j+1} are set to the old values $\{p_j\}$ and χ^2_j.

4. j is increased to $j + 1$.

5. Iterate steps (2-4) until j reaches $N_{\rm iter}$.

6. Calculate σ^2 of the distribution: if $\sigma_\alpha < \sigma$ sampling has been too local, and sampling has to be repeated with a larger value for σ_α. If $\sigma_\alpha > \sigma$, samples will be removed using the parameter σ_β such that $\sigma^{-2} = \sigma_\alpha^{-2} + \sigma_\beta^{-2}$. The algorithm is optimised if only few samples are removed, i.e., σ_α is close to σ; this can be verified in the *bbx_errordata.out* output file, see Section V.

Even if sampling of the χ^2 landscape near the minimum is succesful (which should be verified by i) the mean $\langle p_i \rangle$ drawn from the distribution being sufficiently close to $p_{i,\rm bestfit}$, that is, $(\langle p_i \rangle - p_{i,\rm bestfit})^2 < \sigma_i^2$, and ii) the width of the χ^2 distribution, represented by σ^2 agreeing with the sample distribution), the sample distribution may describe the χ^2 landscape near a local minimum rather than near the global minimum.

In practice, the software performs this algorithm for different initial conditions, which are obtained by global sampling. In this case, initial conditions are randomly sampled from a uniform distribution between lower and upper boundaries that can be estimated from physical arguments.

Summarised, the algorithm is controlled using the lower and upper boundary values of the physical parameters and the numerical settings:

- $N_{\rm seeds}$: The number of initial conditions or seeds.
- $N_{\rm iter}$: The number of Monte Carlo steps per seed
- σ_α: Parameter that controls the Monte Carlo acceptance probability.
- δ: Parameter that controls the Monte Carlo step size.

III. FILE STRUCTURE AND INSTALLATION

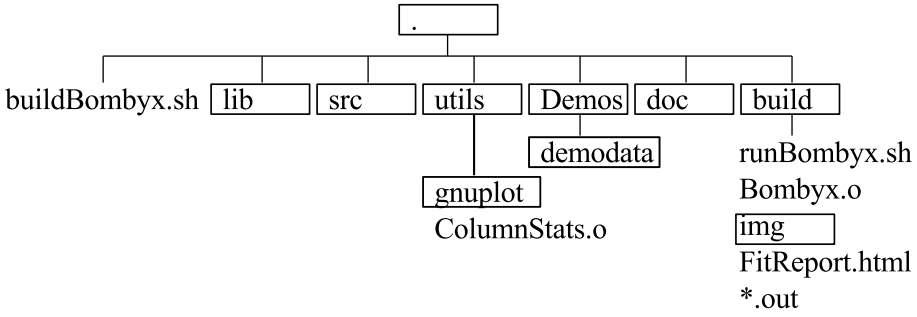


FIG. 1: File structure. ∗ Prior to installation, the *build* directory and its contents is not present.

Installation is done by running

```
./buildBombyx.sh
```

which created the *build* directory, compiles the libraries in *lib*, compiles postprocessing tools (*.o*) in *utils*, and compiles *Bombyx.o* in *build*. (**NOTE:** software was tested for Linux/Debian (see readme.md); it was not tested on windows, where it is likely required to change the .o extensions to .exe). Further, this script copies the *runBombyx.sh* script from *utils* to *build*.

Testing the software can be done by navigating to the build directory and running the shell script:

```
cd build
./runBombyx.sh -r -p
```

Default, this runs a demo curve fit of the data in *Demos/demodata* (in **.bbx* format), and produces data (in ∗*.out*, html, img/∗.png format). Running the demo requires a couple of minutes on a regular desktop computer. The expected output should be similar to the demo output provided in *Demos/demodata/demodata_expected_output*. Fast visual verification can be achieved by viewing the FitReport.html files (e.g., using a browser such as firefox or chrome).

IV. INPUT

The curve-fitting algorithm is performed by the executable *Bombyx.o* that can be manually run by the user as
`./Bombyx.o <datafile.bbx> [arguments]`
or (as recommended) using the shell script *runBombyx.sh*. The input parameters to *Bombyx* are defined in the "USER INPUT" section of the *runBombyx.sh* file:

```
#####################################################################
# USER INPUT
inputfile=‘‘../Demos/demodata/demodata.bbx’’
outfile=‘‘bbx_errordata.out’’
rseed=-1 # random number seed.  -1:  time stamp; >0:  user-defined integer
MCseeds=50 # Number of Monte Carlo seeds
MCiter=1000 # Number of Monte Carlo iterations per seed
MCdelta=0.2 # small value -> small step size
MCsigma=0.2 # small value -> larger acceptance probability
ZeL=4 # Lower value for number of entanglements
ZeU=30 # Upper value for number of entanglements
ZsL=1 # Lower value for number of stickers
ZsU=10 # Upper value for number of stickers
tauSL=3.5e-3 # Lower value for sticker lifetime
tauSU=0.25 # Upper value for sticker lifetime
fL=0.2 # Lower value for prefactor f:  Ge = f * Ge_estimate
fU=2.0 # Upper value for prefactor f:  Ge = f * Ge_estimate
alpha=10 # material parameter
# END USER INPUT
#####################################################################
```

By running
`./runBombyx.sh -c -r -p`
the software is compiled (`-c`), run (`-r`), and the results (consisting of samples of the χ^2 landscape) are postprocessed (`-p`). All possible arguments to the executable are displayed by running `./Bombyx.o --help` and all possible arguments to the shell script are displayed by running `./runBombyx.sh --help`. Below, the *.bbx* data file and its most important input parameters are discussed.

A. Data file

Example input in a *.bbx* file (taken from *Demos/demodata/demodata.bbx*):

```
#DegreeOfPolymerisation:  5525 # number of monomers per polymer chain
#MonomerVolume:  0.13e-27 # in cubic meters
#VolumeFraction:  0.228 # occupied by the polymer (in solution)
#Temperature:  288 # in Kelvin
#Frequency:  1 # 0=‘‘Angular’’ or 1=‘‘Hertz’’
#FirstDataLine:  13 # Start reading data from this line
#LastDataLine:  27 # Last line with data
#-----------------------
#omega Gvisc omega Gelas
#-----------------------
12.537 2636.0 12.537 9748.8
8.878 2875.4 8.878 8933.7
6.287 3131.0 6.287 8287.0
```

The physical information 'DegreeOfPolymerisation' (N), 'MonomerVolume' b, 'VolumeFraction' (ϕ), 'Temperature' (T) is used in the program to estimate the elastic modulus

$$G_{\mathrm{e,estimate}} = \frac{4}{5}\frac{\phi Z_{\mathrm{e}}}{b^3 N} k_{\mathrm{B}} T, \tag{4}$$

with Z_{e} the number of entanglements per chain (which is varies in the fitting algorithm). The actual fit value is $G_{\mathrm{e,fit}} = f G_{\mathrm{e,estimate}}$, where f is set by the user, see Section IV B.

B. Setting boundaries to physical parameters

TABLE I: Lower (L) and Upper (U) values of physical parameters.

parameter	function	*Bombyx* argument
τ_s	(L) sticker dissociation time	`--tauSL <value>`
	(U)	`--tauSU <value>`
Z_s	(L) number of stickers	`--ZsL <value>`
	(U)	`--ZsU <value>`
Z_e	(L) number of entanglements	`--ZeL <value>`
	(U)	`--ZeU <value>`
f	(L) elasticity modulus prefactor	`--GeL <value>`
	(U)	`--GeU <value>`
	(fixed)	`--Ge-fac <value>`
α	(L) contour-length fluctuations	`--alphaL <value>`
	(U)	`--alphaU <value>`
	(fixed)	`--alpha <value>`

Bombyx estimates upper and lower boundary values for τ_S, Z_E, Z_S from the data. These can be overridden by the user by parsing the arguments `--tauSL <value>`, `--tauSU <value>`, `--ZsL <value>`, `--ZsU <value>`, `--ZeL <value>`, `--ZeU <value>`, `--alphaL <value>`, `--alphaU <value>` to the program, for example:
`./Bombyx.o datafile.bbx --tauSL 1e-4 --tauSU 1e-2`
Parameter values can be fixed by setting the lower and upper boundaries to the same value. In case of the parameter α, the value can also be fixed using the argument `--alpha <value>`.

The parameter G_e is estimated by Bombyx using the relation between the number density of entanglements (ρ) and the thermal energy via $G_e \propto \rho k_B T$ (see previous section). The prefactor is estimated using the degree of polymerisation, monomer volume, volume fraction and the temperature, which are defined in the data file (see previous section). Variations from the estimated modulus ($G_{e,fit} = f G_{e,estimated}$) can be allowed in the fit using the arguments `--GeL <value>`, `--GeU <value>`. These values give the upper and lower bounds of the factor f and *not* of the modulus itself.

C. Setting numerical parameters

TABLE II: Numerical program settings.

parameter	function	*Bombyx* argument
-	random number seed	`--rseed <value>`
N_{seeds}	number of initial conditions / seeds	`--MCseeds <value>`
N_{iter}	number of Monte Carlo steps per seed	`--MCiter <value>`
δ	Monte Carlo step size	`--MCdelta <value>`
σ_α	Controls the acceptance probability	`--MCsigma <value>`

The algorithm makes use of a random-number generator, which is default seeded using the same integer value. This value may be overridden using `--rseed` and the number of seeds using `--MCseeds`

```
./Bombyx.o <file> --rseed <val> --MCseeds <val>
```

The value for `--rseed` should be a positive integer to set to a fixed value, or -1 to use the computers clock / time stamp to initialise the random number. The value for `--MCseeds` should be a positive integer, typically with a value $5 - 100$.

New parameters in step $i + 1$ of the algorithm are proposed (step 2 of the MC algorithm) randomly uniform from the interval

$$\max\{p_i(1-\delta), p_{i,L}\} \leq p_{i+1} \leq \min\{p_i(1+\delta), p_{i,U}\} \tag{5}$$

The step size, or the width of the interval, is controlled by the parameter $\delta < 1$, For large step sizes ($\delta \approx 1$), the proposed parameter values are independent of the previous values, and the process is χ^2 landscape is sampled randomly. For small step sizes ($\delta \approx 0$), the parameter values are likely to get trapped in a local minimum.

The value at which efficient sampling is achieved depends on the acceptance probability of the new parameter values. This probability is

$$P = \min\left\{1, \exp\left(-\frac{\chi^2_{\text{new}} - \chi^2_{\text{old}}}{\sigma_\alpha}\right)\right\}, \tag{6}$$

which is controlled by $\sigma_\alpha > \sigma$, with σ the standard deviation of the χ^2 distribution (see algorithm section). For large σ_α, the acceptance probability is high, and a large portion of the parameter space can be explored. For a small value, algorithm may get stuck in local minima of the parameter space.

V. OUTPUT

After running the program, output is generated in

- *bbx_convergence.out*: contains all accepted data generated using σ_α (see algorithm section).
- *bbx_convergence_filtered.out*: contains all data after filtering the data using σ_β (see algorithm section). This data is used to calculate the mean and standard deviation of the physical parameters.
- *bbx_errordata.out*: contains the parameter values and their error analysis, including the mean and standard deviation, but also the σ, σ_α, and σ_β ("`chi_std_a`", "`chi_std_b`", "`chi_std`") values that control the efficiency of the algorithm. If *runBombyx.sh* is run in postprocessing mode, *bbx_errordata.out* will contain the Pearson correlation matrix, and minimum, first quartile, median, third quartile and maximum parameter values for box-and-whisker plots.
- *bbx_model.out*: contains the fitted G', G'' model.
- *img*: contains graphs in png format, generated by postprocessing setting of *runBombyx.sh* using the gnuplot scripts in *utils*. These include the curve fit of the model to experiment, convergence plots, histograms, and box-and-whisker plots.
- *FitReport.html*: displays the figures in *img* with commentary.

VI. DEMO

A. Introduction

To extract the values of Z_s and Z_e, τ_s, G_e, and α from experimental data (symbols in Figure 2), it is necessary to set upper and lower values and/or fix parameter values, set the number of iterations and set values for the numerical parameters δ and σ_α. As demonstrated below, in practice one starts fitting globally with broad parameter ranges to find the appropriate numerical parameters and to identify local minima. Then, the local minima can be inspected individually (using the fitting algorithm with different numerical settings). Running the program to reproduce the data in this demo can be done in less than 10 minutes on a regular desktop computer. Before discussing this procedure, correct installation and presence of "demo files" should be checked.

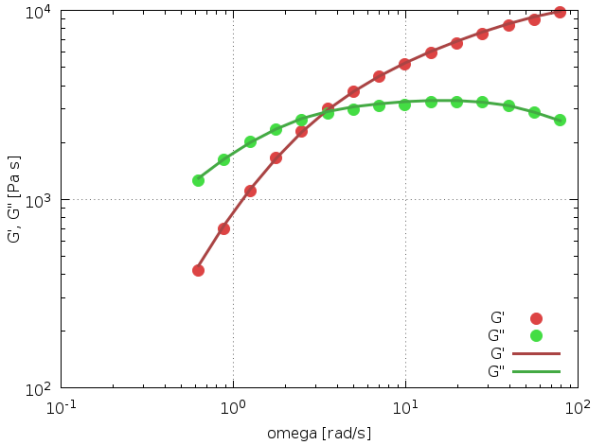


FIG. 2: Viscous and elastic moduli, G' and G'', as a function of the angular frequency ω. The experimental data (symbols) are given in *Demo/demodata*, and the lines are obtained by curve fitting using *Bombyx*.

The software should be installed manually or using the shell script *buildBombyx.sh* (see *readme.md*). After succesful installation, there should be a *build* directory that contains the script *runBombyx.sh* and the executable *Bombyx.o* for running the curve fit, and there should be a *utils* directory that contains the executable *ColumnStats.o* for postprocessing. The demo presented in this section uses the content of the *Demos/demodata* directory. The raw experimental data is provided in *demodata.bbx* in that directory (see Section IV A), and example output after fitting that data is given in the four *FitReport_x_xxx.pdf* files that will be referred to in the remainder of this section.

B. Exploring the parameter space and identifying local minima

The first step to do when fitting the SR model to experiment (see Figure 2) is to estimate the parameter values and assign lower and upper values for these. The experimental data (symbols) in Figure 2 show a roll off of the viscous modulus, G'', at an angular frequency of approximately 40 rad/s, which roughly gives a sticker dissociation time of $\tau_s \approx 0.025$ seconds. The number of stickers and entanglements may be estimated using the terminal relaxation time, which is approximately 0.3 s, as estimated from the frequency 3 rad/s where $G' = G''$. From the two time scales, it follows that the width of the non-sticky rubber plateau is $\ln\left(\alpha Z_e Z_s^2\right) \approx 10$, giving (if $Z_e = Z_s$) approximately 13 stickers and/or entanglements per chain if $\alpha = 10$. The final parameter is G_e, which is estimated by *Bombyx* from $G_e \propto k_B T Z_e$, with the proportionality factor calculated from the physical properties (molecular volume, degree of polymerisation, volume fraction) in *Demo/demodata*. Summarised, it seems reasonable to set as physical boundaries $Z_e \in [1, 30]$, $Z_s \in [1, 30]$, $\tau_s = [0.0025, 0.25]$ and $f = [0.2, 5]$, with f a prefactor to the estimate of G_e. These parameters are set in the *runBombyx.sh* file, which runs *Bombyx.o* with the arguments "`--alpha 10 --ZeL 1 --ZeU 30 --ZsL 1 --ZsU 30 --tauSL 2.5e-3 --tauSU 0.25 --GeL 0.2 --GeU 5`".

Exploration of the parameter space is controlled by the number of initial parameter sets, the number of Monte Carlo (MC) steps per initial condition, N_{iter}, the step size δ and the parameter to set the acceptance probability σ_α. These parameters are set in the *runBombyx.sh* file, which runs *Bombyx* with the arguments "`--Nseeds 50 --MCiter 100 --MCsigma 0.2 --MCdelta 0.4`". Running the algorithm takes about a minute of runtime, and is done by running `./runBombyx.sh -r -p`, which calls `./Bombyx.o ../Demos/demodata/demodata_15_2.bbx --alpha 10 --ZeL 1`

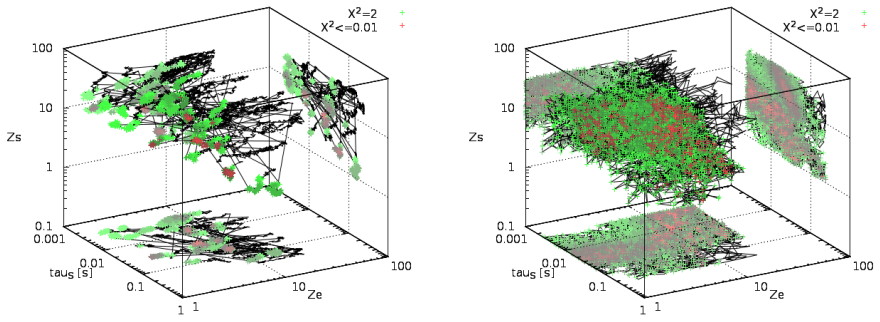


FIG. 3: Explored parameter space for 50 seeds after 100 iterations each, for $\sigma_\alpha = 0.2$ and $\delta = 0.05$ (left) and $\delta = 0.5$ (right).

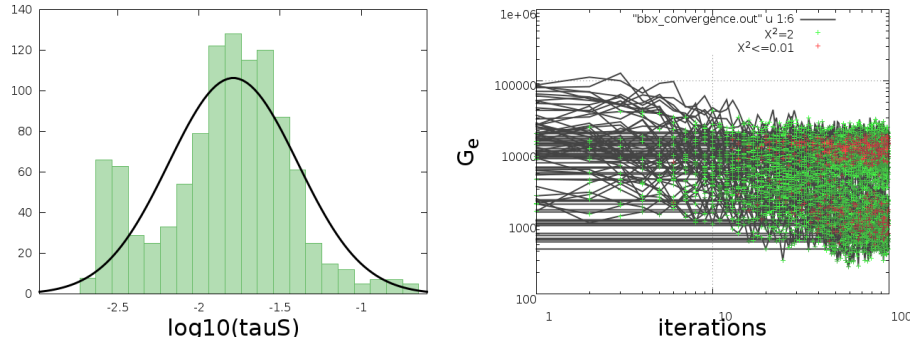


FIG. 4: Explored parameter space for 50 seeds after 100 iterations each, for $\sigma = 0.2$ and $\delta = 0.05$ (left) and $\delta = 0.5$ (right).

`--ZeU 30 --ZsL 1 --ZsU 30 --tauSL 2.5e-3 --tauSU 0.25 --GeL 0.2 --GeU 5 --Nseeds 50 --MCiter 100 --MCsigma 0.2 --MCdelta 0.4` Figure 3 shows the exploration of the parameter space (in the *img* directory and in the *FitReport.sh* file) for $\delta = 0.05$ and for for $\delta = 0.4$: For small δ, the parameters get stuck in local minima, while for large δ the parameter space is explored more efficiently. A similar result would be seen by varying the value of σ_α.

The choice for σ_α can be reviewed in the output in *bbx_errordata.out*, in lines alike

```
#Filtered data:
#Ndata = 33895
#chibest = 0.000863
#chi_std_a = 0.282843
#chi_std_b = 5.100172e-01
#chi_std = 2.540926e-01
#(1/va+1/vb)^(-1/2) = 2.473519e-01
```

Here, 'chi_std_a' represents the input value σ_α, 'chi_std_b' represents σ_β which is found by the program to match, within 10%, 'chi_std' (representing σ) to (1/va+1/vb)^(-1/2) (representing $sqrt\left(\sigma_\alpha^{-2} + \sigma_\beta^{-2}\right)$) (see algorithm section of this manual). The value 'chi_std_a' should be larger than that of 'chi_std'. If it is much larger, then σ_α may be lowered to improve the efficiency of sampling the parameter space.

For $\delta = 0.4$, Figure 4 shows a bimodal distribution for the sticker dissociation time (left). The first peak is at times smaller than actually assessed experimentally in Figure 2, and has a numerical origin: To exclude this local minimum, the fitting procedure should be rerun with as lower boundary of τ_S set to 0.0065 (which is still smaller than the smallest time experimentally available, guaranteeing a sufficiently broad interval). The right panel of Figure 4 shows the convergence of the elasticity modulus. The converged values are typically lower than the initial value. For the rerun of the fit the upper value of the prefactor is therefore set from 5 to 2.

C. Finding and characterising the global minimum

After rerunning the algorithm with the adapted (see *Demos/demodata/FitReport_1_global.pdf*), the elasticity modulus G_e remains to have a bimodal distribution, while Z_s and Z_e have scewed distributions. The value of G_e is correlated

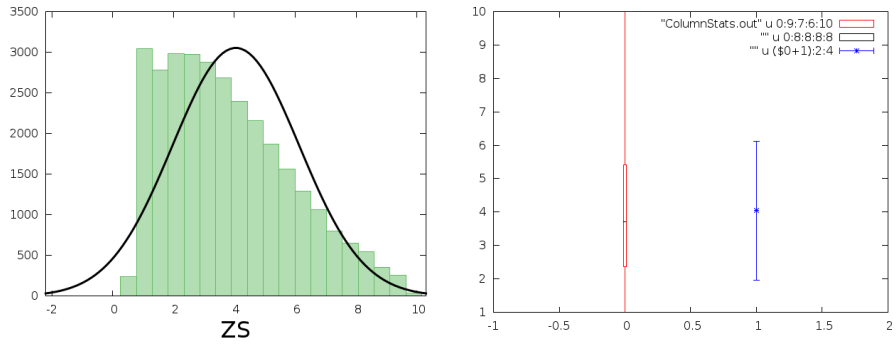


FIG. 5: Analysis of number of stickers, Z_S. Left: skewed distribution from sampling (bars) compared to Gaussian distribution (line). Right: Box-and-whisker plot (red) with median (black) compared to mean and standard deviation (blue).

with Z_e and anticorrelated with Z_s (can be concluded from the Pearson correlation coefficients in *bbx_errordata.out*, which is produced after running *runBombyx.sh* in postprocessing mode). Indeed, the right panel of Figure 4 displays two local minima with high Z_s and low $Z_\mathrm{e}, G_\mathrm{e}$ and with low Z_s and high $Z_\mathrm{e}, G_\mathrm{e}$. The separation of these minima is at $Z_\mathrm{e} \approx 4$ and $Z_\mathrm{s} \approx 10$, which can now be inspected separately by setting the boundaries appropriately.

The fit reports *Demos/demodata/FitReport_2_local_minimum_1.pdf* and *Demos/demodata/FitReport_3_local_minimum_2.pdf* are produced by rerunning the software with as boundary conditions $\{Z_\mathrm{e} \in [1,\,4], Z_\mathrm{s} \in [10,\,30]\}$ and $\{Z_\mathrm{e} \in [4,\,30], Z_\mathrm{s} \in [1,\,10]\}$, respectively. Further, for both runs the prefactor to G_e was set $f \in [0.2,\,2]$ and $\tau_\mathrm{s} \in [0.0065, 0.25]$, and all other settings were kept the same as in the previous section. From the fit quality in both data sets, it was concluded that the first data set (for $\{Z_\mathrm{e} \in [1,\,4], Z_\mathrm{s} \in [10,\,30]\}$) represented a local minimum and the second dataset (for $\{Z_\mathrm{e} \in [4,\,30], Z_\mathrm{s} \in [1,\,10]\}$) represented a global minimum.

This curve fit was now rerun, but to improve the statistics it was run with 1000 Monte Carlo steps per seed instead of 100. The results are summarised in *Demos/demodata/FitReport_4_final.pdf*. The χ^2 landscape plot suggests only the vicinity of a single (global) minimum is sampled. This parameter space is not ellipsoidal due to the cross correlations between the parameters. While the G_e distribution is close to Gaussian, the distributions of the other parameters are skewed. Despite the skewness, comparison of the box-and-whisker plots with the mean and standard deviation show acceptable agreement between the median and mean well within the standard deviation, see Figure 5.

VII. APPENDIX: STICKY-REPTATION MODEL

To calculate the relaxation modulus $G(t)$, the SR model [1] extends the usual description of linear entangled polymers [2] with a description for stickers. That is, $G(t)$ is composed of Rouse relaxation of monomers along the backbone of a bead-spring chain, and of relaxation by the constraint release of Z_e topological entanglements via polymer reptation in a tube-like confinement. The stickers are represented by sticky monomers of which a number Z_s per chain actually forms a bridge. These stickers hinder Rouse relaxation for wavelengths longer than the strand between stickers. For those long-wavelength modes, the relaxation modulus is

$$G_\mathrm{SR}(t) = \frac{G_\mathrm{e}}{Z_\mathrm{e}} \sum_{q=1}^{Z_\mathrm{s}} \kappa \exp\left(-\frac{tq^2}{\tau_\mathrm{s} Z_\mathrm{s}^2}\right), \tag{7}$$

where the coefficient κ equals unity for modes with a wavelength shorter than the entanglement strand (i.e., for wavenumbers $q \geq Z_\mathrm{e}$) and equals $1/5$ for longer wavelengths (i.e., for wavenumbers $q < Z_\mathrm{e}$) [3].

Under the assumption that the Rouse relaxation of short wavelengths is much faster than the dissociation of stickers (i.e., $\tau_\mathrm{s} \gg \tau_0 (N/Z_\mathrm{S})^2$, (i) τ_s is discernible from early-stage relaxations as a maximum in G'', and (ii) the time scale of 'sticky-reptation' at which the chain escapes the tube-like confinement is $\tau_\mathrm{rep} = \tau_\mathrm{s} Z_\mathrm{e} Z_\mathrm{s}^2$ [1]. Following Chen et al. [4, 5], the curvilinear motion of the sticky polymer in the tube is described using the double-reptation model [6]

$$G_\mathrm{rep}(t) = G_\mathrm{e} \left(\frac{8}{\pi^2} \sum_{\mathrm{odd}\ q} \frac{1}{q^2} \exp(-q^2 U(t)) \right)^2, \tag{8}$$

with

$$U(t) = \frac{t}{\tau_{\rm rep}} + \frac{\alpha}{Z_{\rm e}} g\left(\frac{Z_{\rm e}}{\alpha}\frac{t}{\tau_{\rm rep}}\right), \tag{9}$$

and with $g(x) = \sum_{m=1}^{\infty} m^{-2}[1 - \exp(-m^2 x)]$ a function that captures approximately the modification to reptation entanglement loss arising from contour-length fluctuations. In this model, α should in principle equal a universal constant value, but is in practice found to increase with a decreasing number of entanglements per chain [7], which may potentially be corrected for using a more sophisticated description for contour-length fluctuations [3]. The parameter α predominantly affects the terminal relaxation time as $\tau_{\rm d} = \tau_{\rm rep}/\alpha$ for $\alpha \gg 1$. Transformation of $G(t)$ from the time to the frequency domain is numerically achieved using the Finite Element Analysis detailed in Ref. [8].

[1] L. Leibler, M. Rubinstein, and R. H. Colby, Macromolecules **24**, 4701 (1991).
[2] T. C. B. McLeish, Adv. Phys. **6**, 1379 (2002).
[3] A. E. Likhtman and T. C. B. McLeish, Macromolecules **35**, 6332 (2002).
[4] Q. Chen, Z. Zhang, and R. H. Colby, J. Rheol. **60**, 1031 (2016).
[5] Z. Zhang, Q. Chen, and R. H. Colby, Soft Matter **14**, 2961 (2018).
[6] J. des Cloizeaux, Europhys. Lett. **5**, 437 (1988).
[7] E. van Ruymbeke, R. Keunings, V. Stéphenne, A. Hagenaars, and C. Bailly, Macromolecules **35**, 2689 (2002).
[8] M. R. Nobile and F. Cocchini, Rheol. Acta **40**, 111 (2001).