Characterising Different Molecular Landscapes in Dynamic Covalent Networks

Filip Van Lijsebetten,[a] Kevin De Bruycker,[a] Evelyne Van Ruymbeke,[b] Johan M. Winne*[a] and Filip E. Du Prez*[a]*

[a] *Polymer Chemistry Research group, Centre of Macromolecular Chemistry (CMaC) and Laboratory of Organic Synthesis, Department of Organic and Macromolecular Chemistry, Faculty of Sciences, Ghent University, Krijgslaan 281-S4, Ghent, 9000, Belgium.*

E-mail: Filip.DuPrez@UGent.be; Johan.Winne@UGent.be

[b] *Bio and Soft Matter, Institute of Condensed Matter and Nanosciences, Université Catholique de Louvain, Croix du Sud 1, Louvain-la-Neuve, 1348, Belgium.*

E-mail: Evelyne.VanRuymbeke@uclouvain.be

Experimental section

Materials. Butylamine (> 99 %, TCI Europe), butyl glycidyl ether (95 %, Sigma Aldrich), 1,6-hexanediol (99 %, Sigma Aldrich), magnesium sulfate ($MgSO_4$, anhydrous, Boom), pyridine-dry (99.5 %, extra dry over molecular sieves, Fisher Chemical), trimellitic anhydride chloride (98 %, Sigma Aldrich), toluene (≥ 99.5 %, Fisher Chemical), trimethylolpropane (97 %, Sigma Aldrich), tert-butyl acetoacetate (TBAA, > 98 %, TCI Europe), tetrahydrofuran (THF, > 99.8 %, Acros Organics), $CDCl_3$ (Euriso-top), Pripol 2033 and Priamine 1074 were kindly provided by Croda. All reagents were used without further purification unless stated otherwise.

Instrumentation. Nuclear magnetic resonance (NMR) analyses were conducted on a Bruker Avance 300 (300 MHz) to measure proton spectra at 25 °C. The NMR spectra were measured in $CDCl_3$, and chemical shifts (δ) are presented in parts per million (ppm), relative to $CDCl_3$ as the internal standard. Attenuated total reflection - Fourier-transform infrared spectroscopy (ATR-FTIR) spectra were measured using a Perkin–Elmer Spectrum1000 FTIR infrared spectrometer with a diamond ATR probe. Thermogravimetric analyses (TGA) were performed with a Mettler Toledo TGA/ SDTA851e instrument under nitrogen atmosphere at a heating rate of 10 $K.min^{-1}$ from 25 °C to 800 °C for the dynamic mode. Isothermal measurements were conducted under air at 150 °C for 120 min. Differential scanning calorimetry (DSC) analyses were performed with a Mettler Toledo instrument 1/700 under nitrogen atmosphere at a heating and cooling rate of 10 $K.min^{-1}$. Measurements were performed from -50 to 150 °C. Temperature-modulated DSC (mDSC) experiments were performed on disc-shaped samples of 20 to 30 mg by recording the required energy to raise the temperature over the range of 80 °C to 160 °C with a heating rate of 0.5 $K.min^{-1}$. Heat capacity (C_p) values were determined by performing a TOPEM evaluation using the STARe software. The signal was adjusted using a sapphire reference curve. To do this, the measured curve was corrected with the sapphire reference curve obtained from a reference measurement with sapphire. The sapphire method measures the C_p of a sample in comparison to the C_p of a sapphire standard. A SpeedMixer DAC 150.1 FVZ was used to homogenise the samples before curing. Rheology experiments were performed on an Anton Paar MCR 302. The experiments were performed in parallel plate geometry using 8 mm sample disks. Each sample was weighed before measurement, yielding a mass between 72 and 78 mg. Amplitude sweep experiments were performed using a frequency of 1 Hz, a constant force of 1 N, and a variable shear strain that was ramped up logarithmically from 0.01% to 10%. Stress-relaxation experiments were performed at different temperatures (160 to 110 °C, with intervals of -10 °C) using a constant shear strain of 0.5%, within the linear viscoelastic region of the samples, and a constant force of 0.2 N. Each measuring step was preceded by a force normalisation step of 0.2 N, while monitoring the gap between each plate. Eyring analysis. Ln ($1/T\ \tau^*$), whereby the relaxation time was obtained from each fitting model, was plotted as a function of 1000/T resulting in an Eyring plot for sample N-1% to N-20%. The activation enthalpy ($\Delta H^{\ddagger}$) was calculated from the slope $\frac{-\Delta H^{\ddagger}}{R}$ with R equal to the universal gas constant. The activation entropy ($\Delta S^{\ddagger}$) was calculated from the intercept $\ln\left(\frac{\kappa k_B}{h}\right) + \frac{\Delta S^{\ddagger}}{R}$ with the transmission coefficient (κ) assumed to be unity, k_B the Boltzmann constant and h the Planck's constant. Van 't Hoff analysis. -Ln (K_{diss}) (see SI for calculations) was plotted as a function of 1000/T resulting in a Van 't Hoff plot for sample N-1% to N-20%. The enthalpy (ΔH) was calculated from the slope ($\frac{\Delta H}{R}$) with R equal to the universal gas constant. The entropy (ΔS) was calculated from the intercept ($-\frac{\Delta S}{R}$).

Reprocessing. To reprocess the network, the material was broken into pieces of 1 mm in size and placed into a rectangular mould (A: 70 mm x 40 mm x 2 mm; B: 30 mm x 15 mm x 2 mm) for compression moulding. This assembly was placed in a 150 °C preheated compression press for 1 min

under 0.5 metric tons of pressure. Then the pressure was increased to 2 tons and kept constant for an additional 4 to 19 min. After 5 to 20 min of pressing in total, the sample was carefully removed from the mould while still heated and in its elastic state. The temperature and pressing time were adjusted according to the amount of aminodiol present in the sample.

Solubility and hydrolysis tests. were carried out with samples of 4 mm diameter and 2 mm of thickness with a weight of around 20 mg and 20 mL of THF. Those tests were performed for 24 h at 25 °C in THF. The solvent was then removed, and the samples were dried under vacuum overnight at 40 °C. The soluble fraction was calculated using **eq. 4**, while the swelling ratio was calculated using **eq. 5**.

$$\text{soluble fraction } (\%) = \frac{m_i - m_d}{m_i} \quad (4)$$

$$\text{swelling ratio } (\%) = \frac{m_s - m_i}{m_i} \quad (5)$$

with m_i, m_s, and m_d stand for initial, swollen, and dry mass, respectively.

Synthetic procedures.

Pripol dianhydride (**1**). Pripol dianhydride was synthesised according to a previously described procedure. In a two-neck round bottom flask, trimellitic anhydride chloride (39.214 g, 186.2 mmol, 1 eq) was dissolved in 300 mL of a mixture of toluene containing a small amount of $MgSO_4$. The mixture was cooled to 0 °C and placed under nitrogen. Pripol 2033, a C36 dimer fatty acid-derived alcohol, (50 g, 96.1 mmol, 0.5 eq) was dissolved in 100 mL of toluene together with dry pyridine (15.0 mL, 186.2 mmol, 1 eq). This alcohol solution was added dropwise to the cooled acid chloride. The mixture was slowly heated to room temperature and stirred for another 16 hours. The mixture was filtered to remove the formed pyridine salts and concentrated in vacuo to obtain the product as a yellowish viscous oil. The anhydride was used without further purification Yield: 96 % ^{1}H NMR (400 MHz, $CDCl_3$, see **Figure S22**): δ (ppm) = 8.64 (s, 2H, 2xAr-**H**), 8.57 (dd, J = 7.9 Hz; 1.4 Hz, 2H, 2xAr-**H**), 8.10 (m, 2H, 2xAr-**H**), 4.41 (t, J = 6.7 Hz ,4H, 2xC**H₂**-O,) 1.77-1.86 (m, 4H, 4xC**H**), 1.02- 1.59 (m, 62H, 31xC**H₂**), 0.74-0.97 (m, 6H, 2xC**H₃**).

Aminodiol (**2**). Equimolar amounts of butylamine (4.21 g, 57.61 mmol, 1 eq) and butyl glycidyl ether (15 g, 115.22 mmol, 2 eq) were added to a glass vial with screw cap. Next, the mixture was heated for 16 h at 70 °C until full conversion of the epoxide. The obtained aminodiol was used without further purification. Yield: 98 % ^{1}H NMR (400 MHz, $CDCl_3$, see **Figure S23**): δ (ppm) = 3.76-3.82 (m, 2H, 2xC**H**-OH), 3.46 – 3.32 (m, 8H, 4xC**H₂**-O), 3.12-3.18 (m, CH-O**H**, 1H), 2.44 – 2.59 (m, 3xC**H₂**-N, 6H), 1.50 – 1.57 (m, 4H, 2xC**H₂**CH_2-O), 1.21 – 1.44 (m, 8H, 2x(O)C**H₂**CH_3; (N)C**H₂**C**H₂**CH_3), 0.86-0.90 (m, 9H, 3xC**H₃**). ^{13}C NMR (100 MHz, $CDCl_3$, see **Figure S24**): δ (ppm) = 73.14 (O-**C**-C-OH), 73.12 (O-**C**-C-OH), 71.41 (**C**-O-C), 71.39 (**C**-O-C), 68.13 (**C**-OH), 67,79 (**C**-OH), 58.05 (N-**C**-C-OH), 57.71 (N-**C**-C-OH), 55.40 (N-**C**-C), 31.70 (2x**C**-C-O-C), 29.28 (N-C-**C**), 20.48 (N-C-C-**C**), 19.28 (2x**C**-C-C-O-C), 14.02 (N-C-C-C-**C**), 13.90 (**C**-C-C-C-O).

Network synthesis (N-1%, N-5%, N-10% and N-20%). Trimethylolpropane (0.24 g, 1.8 mmol, 0.4 eq.), hexanediol (see values below) and aminodiol **2** (see values below) were added to a 20 mL polypropylene cup and the cup was heated in an oven to melt the alcohol mixture. Pripol dianhydride **1** (4 g, 4.5 mmol, 1 eq.) was added and mixing was done using a DAC 150.1 FVZ speed mixer (typical conditions of mixing: 2 min with a speed of 2500 rpm) to obtain a homogeneous mixture. Then, the cup was placed in an oven at 80 °C for up to 4h to initiate the network formation. Hereafter, the network was further cured for 16 h at 100 °C under vacuum. The following diol mixtures were used for N-1%: hexanediol (0.21 g, 1.8 mmol, 0.396 eq.) and aminodiol **2** (0.01 g, 0.02 mmol, 0.004 eq.); N-5%: hexanediol (0.20 g, 1.7 mmol, 0.38 eq.) and aminodiol **2** (0.03 g, 0.09 mmol, 0.02 eq.); N-10%:

hexanediol (0.19 g, 1.6 mmol, 0.36 eq.) and aminodiol **2** (0.06 g, 0.18 mmol, 0.04 eq.); N-20%: hexanediol (0.17 g, 1.4 mmol, 0.32 eq.) and aminodiol **2** (0.12 g, 0.36 mmol, 0.09 eq.).

1,1,1-Trimethylpropane trisacetoacetate (**5**, TMP-AA). Trimethylolpropane (10 g, 74.5 mmol, 1 eq.) and tert-butyl acetoacetate (42.44 g, 268.2 mmol, 3.6 eq.) were added in a 250 mL flask equipped with a still-head, a thermometer and a cooler. The viscous mixture was heated to 135 °C until the temperature of the vapour dropped to 40 °C. The unreacted tert-butyl acetoacetate was removed by vacuum distillation at 130 °C under 2 mbar pressure. Yield: 94 % ^{1}H NMR (300 MHz, $CDCl_3$): δ (ppm) = 0.85 (t, 3H, C**H$_3$**CH$_2$C), 1.42 (q, 2H, CH$_3$C**H$_2$**C), 2.21 (s, 9H, 3xC**H$_3$**COCH$_2$), 3.45 (s, 6H, 3xCH$_3$COC**H$_2$**COO), 4.04 (s, 6H, 3xCC**H$_2$**OCO) (See **Figure S25**).

Network synthesis (VU-ref and VU-pTsOH). TMP-AA (1.5 g, 3.9 mmol and 1.0 eq.) and Priamine 1074 (3.7 g, 6.7 mmol, 1.725 eq.) were mixed in a 20 mL polypropylene cup using a DAC 150.1 FVZ speed mixer (typical conditions of mixing: 2 min with a speed of 2500 rpm). Then, the cup was placed in an oven at 80 °C for up to 4h to initiate the network formation. Hereafter, the network was further cured for 18 h at 100 °C under vacuum. The same procedure was repeated with the addition of *p*-toluenesulfonic acid (0.1 g, 0.7 mmol and 0.17 eq.) to the curing mixture.

Material characterisation

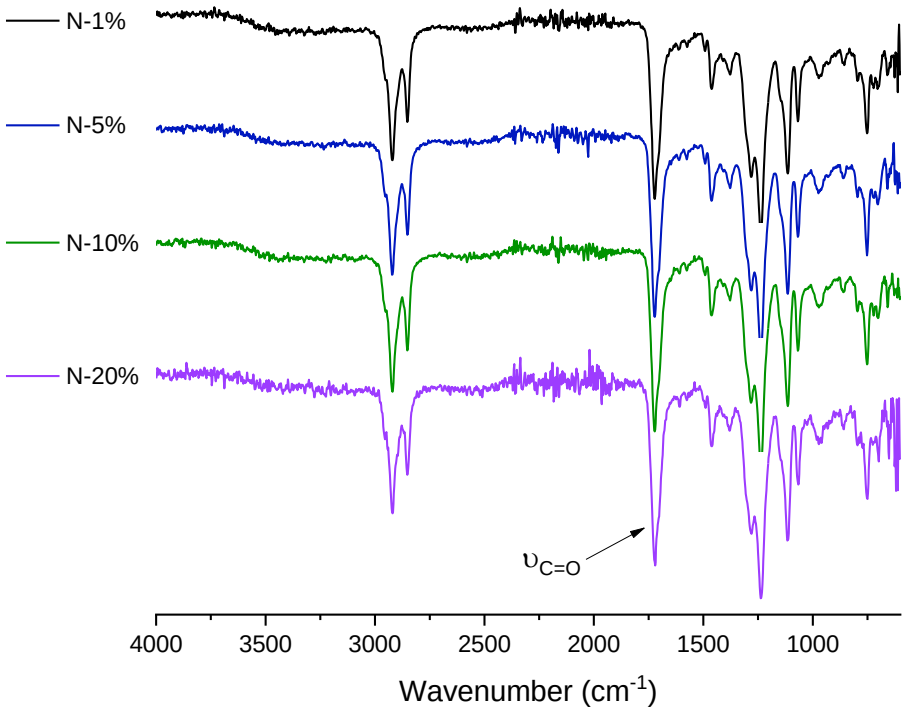


Figure S1. *FT-IR spectrum of N-1%, N-5%, N-10% and N-20%.*

Table S1. Overview of compositions and physical properties of PME networks.

CAN	T_g[a] (°C)	$T_{d5\%}$[b] (°C)	Swel. Rat.[c] (%)	Sol. Frac.[c] (%)
N-1%	14	274	299 ± 5	6.8 ± 0.9
N-5%	6	297	336 ± 7	5.8 ± 0.4
N-10%	7	300	363 ± 23	5.7 ± 0.4
N-20%	13	295	332 ± 24	6.9 ± 0.8

[a] DSC glass transition temperature (T_g), [b] TGA onset temperatures after 5% weight loss ($T_{d5\%}$), [c] obtained after swelling for 24 h in THF.

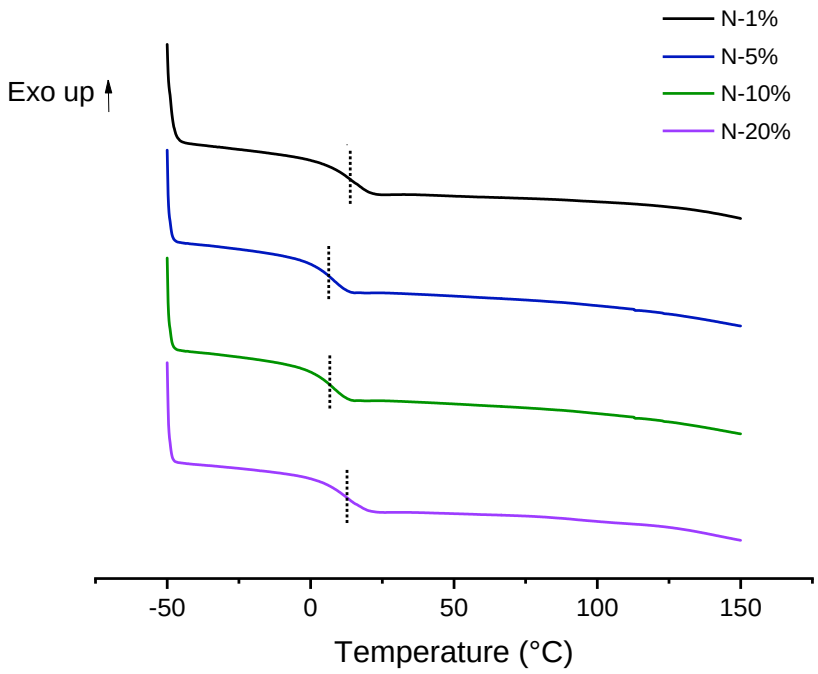


Figure S2. *DSC thermograms of the second heating step of N-1%, N-5%, N-10% and N-20%, measured at a heating rate of 10 °C.min^{-1}.*

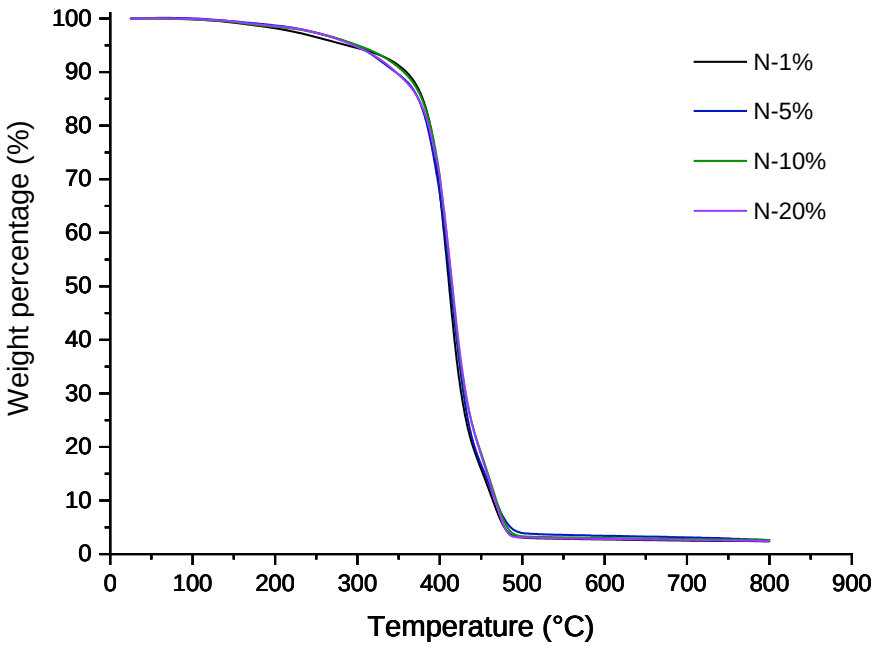


Figure S3. *TGA analysis under N_2 atmosphere of N-1%, N-5%, N-10% and N-20% with a temperature ramp from 25-800 °C and a heating rate of 10 °C.min^{-1}.*

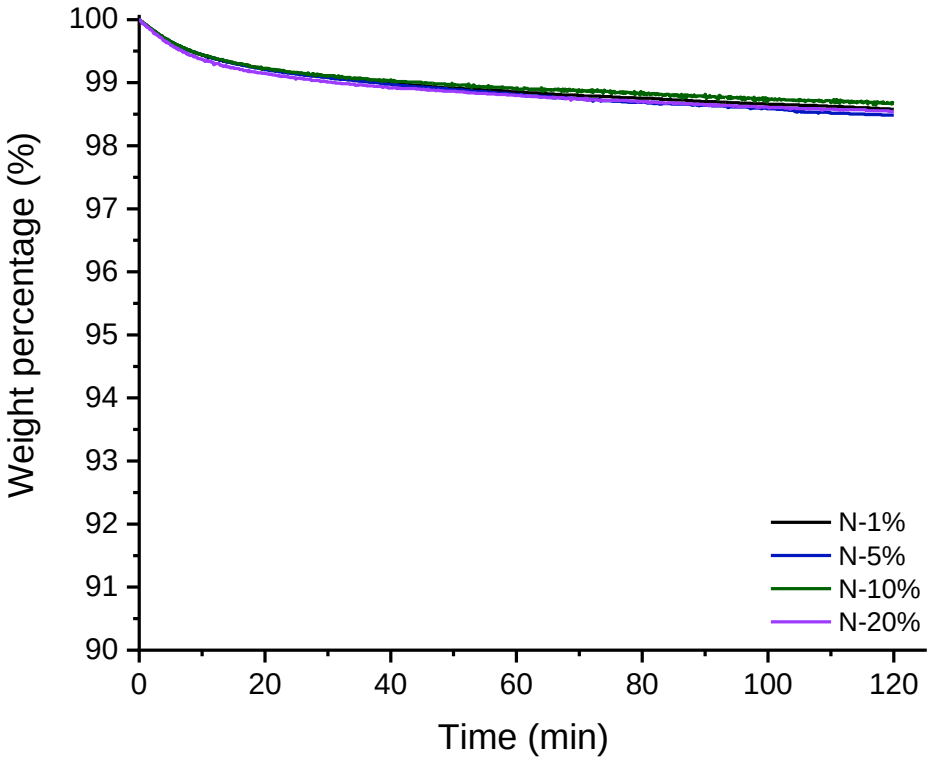


Figure S4. *Isothermal TGA measurements of N-1%, N-5%, N-10% and N-20% at 150 °C for 120 minutes under air.*

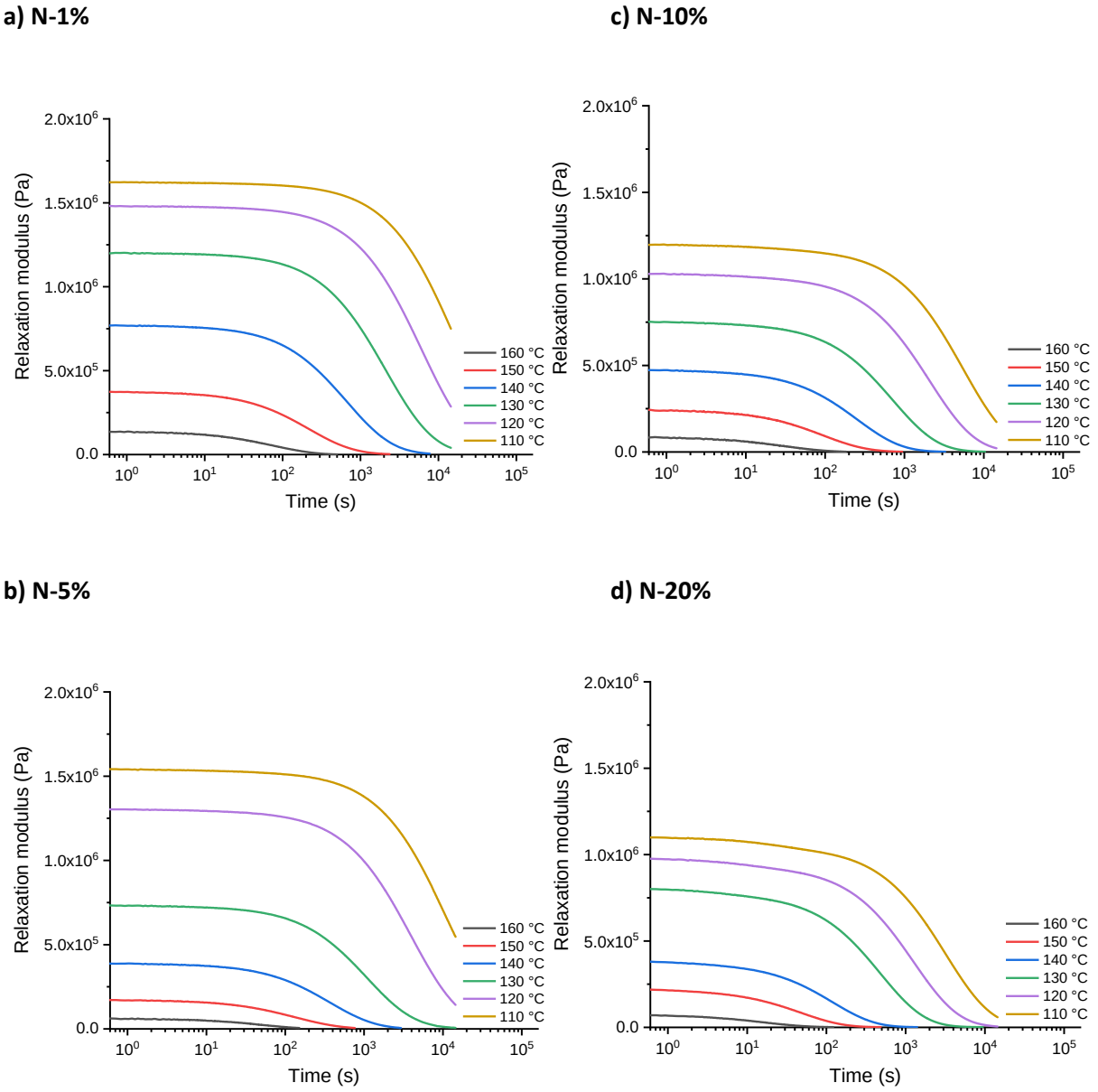


Figure S5. *Stress-relaxation graphs of a) N-1%, b) N-5%, c) N-10% and d) N-20% measured at different temperatures between 160 and 110 °C. Data were not normalised due to a large variation in initial relaxation modulus (G_0) as a function of temperature.*

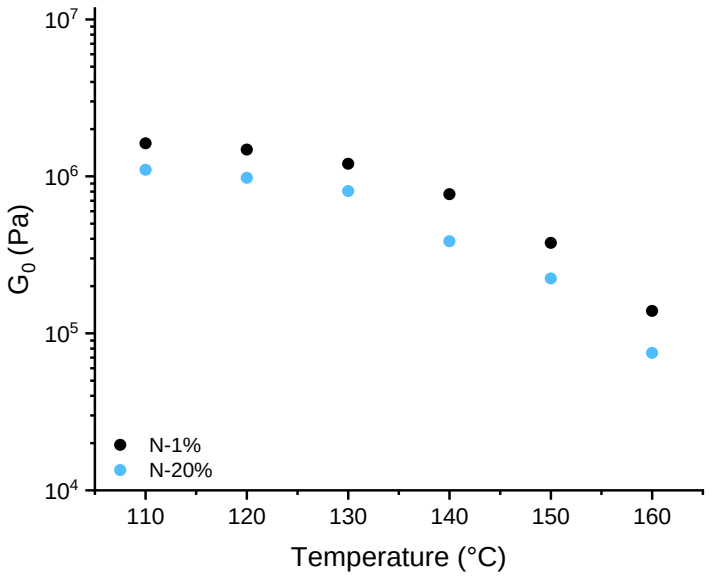


Figure S6. *Overlay of the initial relaxation modulus (G_0) of N-1% and N-20% as a function of temperature.*

$$G(t) = G_0\, e^{\frac{-t}{\tau_{single}}} \qquad (S1)$$

$$G(t) = G_0\, e^{(\frac{-t}{\tau_{stretched}})^{\beta}} \qquad (S2)$$

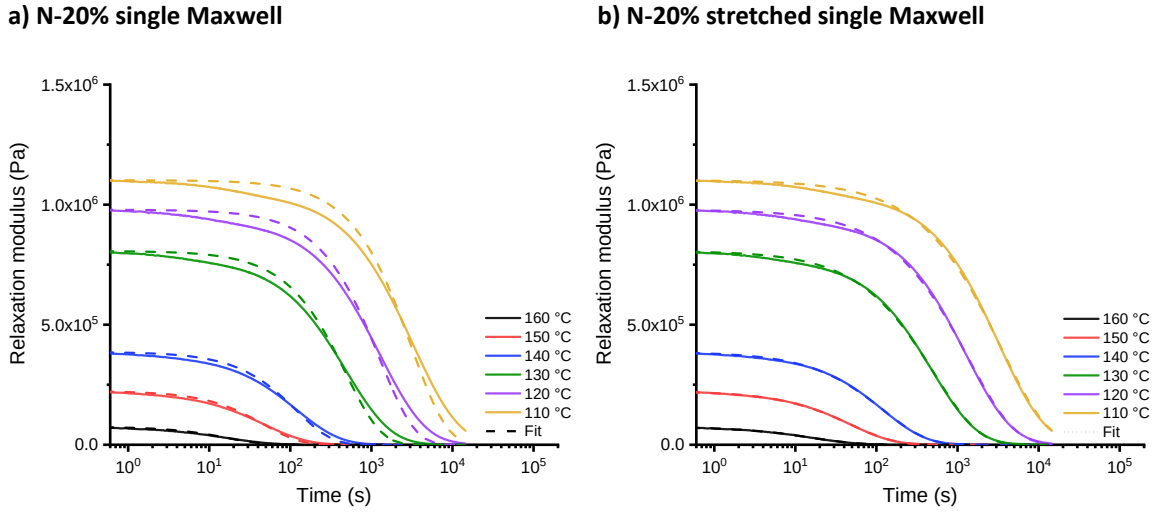


Figure S7. *Stress-relaxation graphs displaying a a) single and b) stretched exponential fit to the stress-relaxation data of N-20% from 160 °C to 110 °C.*

***Table S2**. Representative relaxation parameters obtained by fitting relaxation data of N-1% (left) and N-20% (right) to a generalised Maxwell model (2 elements).*

Temperature (°C)	$G_{0,fast}$ (kPa)	τ_{fast} (s)	$G_{0,slow}$ (kPa)	τ_{slow} (s)
160	32	29	100	105
150	171	126	196	435
140	358	376	404	1399
130	542	1142	651	4628
120	428	3249	1043	10886
110	[a]	[a]	[a]	[a]

Temperature (°C)	$G_{0,fast}$ (kPa)	τ_{fast} (s)	$G_{0,slow}$ (kPa)	τ_{slow} (s)
160	24	7	41	27
150	68	22	140	73
140	113	53	250	181
130	267	220	501	808
120	312	523	626	2142
110	[a]	[a]	[a]	[a]

[a] No complete relaxation during investigated time window.

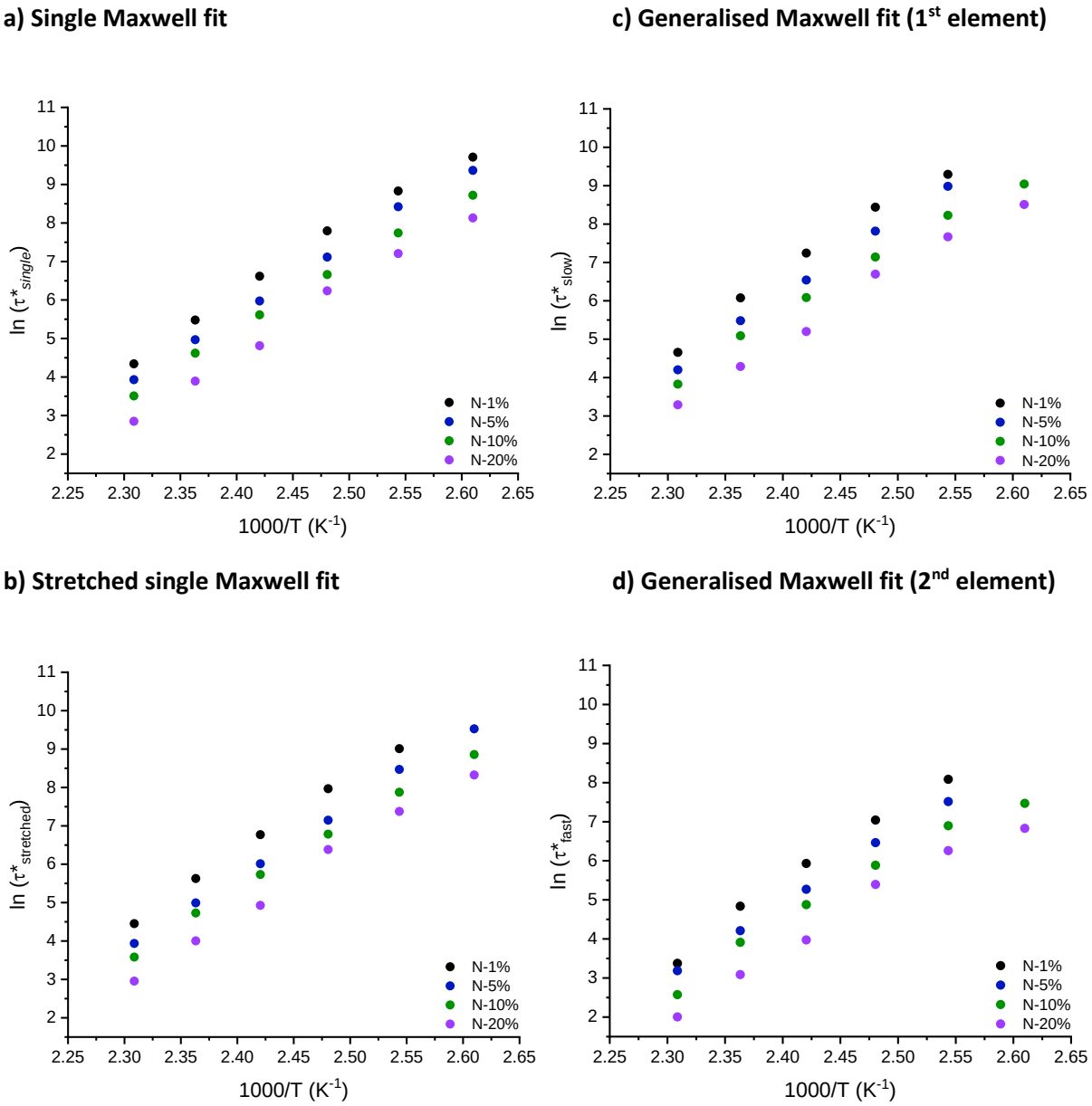


Figure S8. *Temperature dependence of calculated relaxation time (τ^*) values obtained by fitting to a) single exponential, b) stretched exponential, c) double exponential (1st exponent) and d) double exponential (2nd exponent) decay from 160 °C to 110 °C. Note a small deviation from linearity at lower temperatures, when fitting was done to non-sufficiently relaxed data.*

$$\tau^*(T) = \tau_0 \, e^{\frac{-E_a}{RT}} \qquad (S3)$$

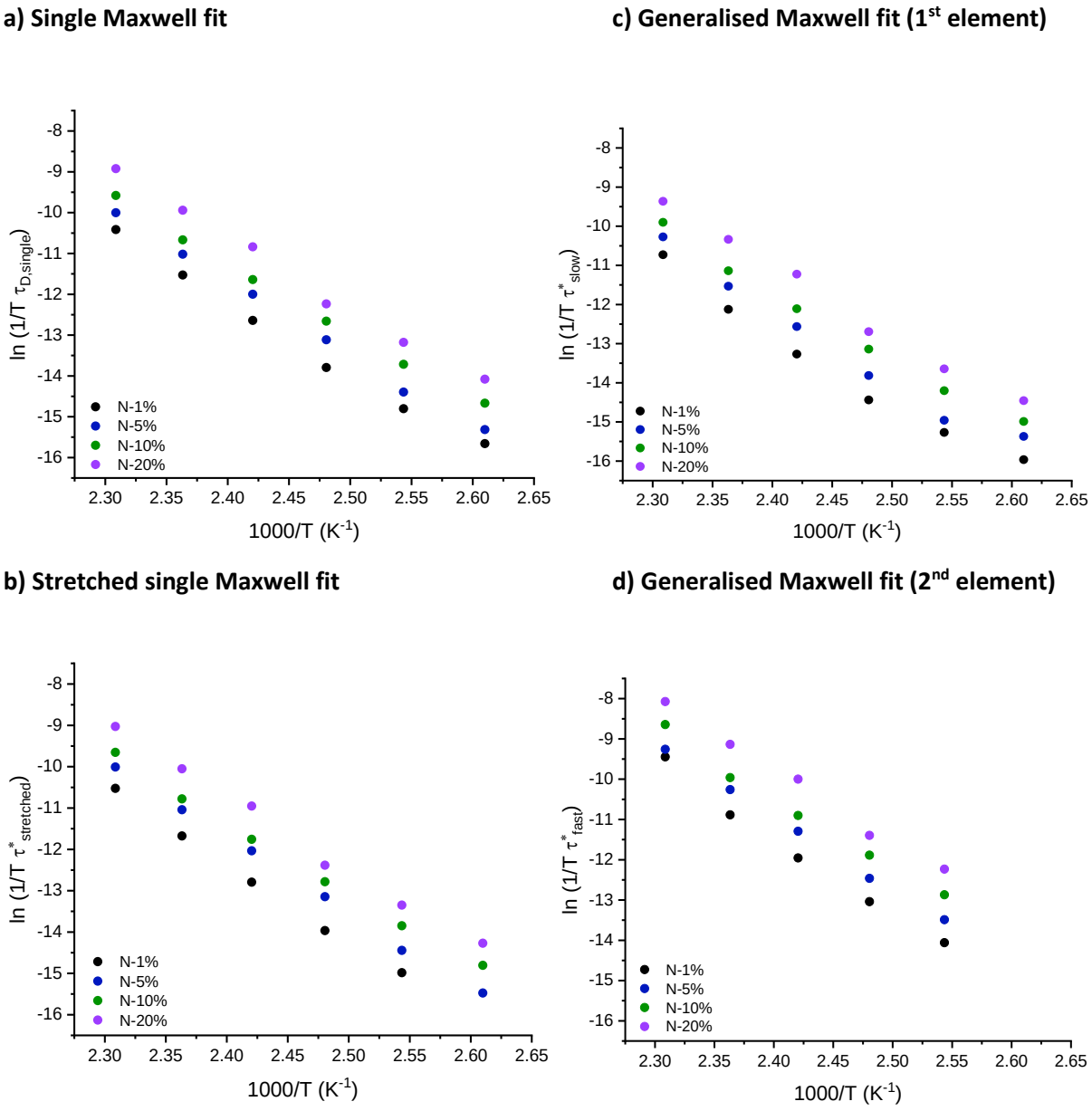
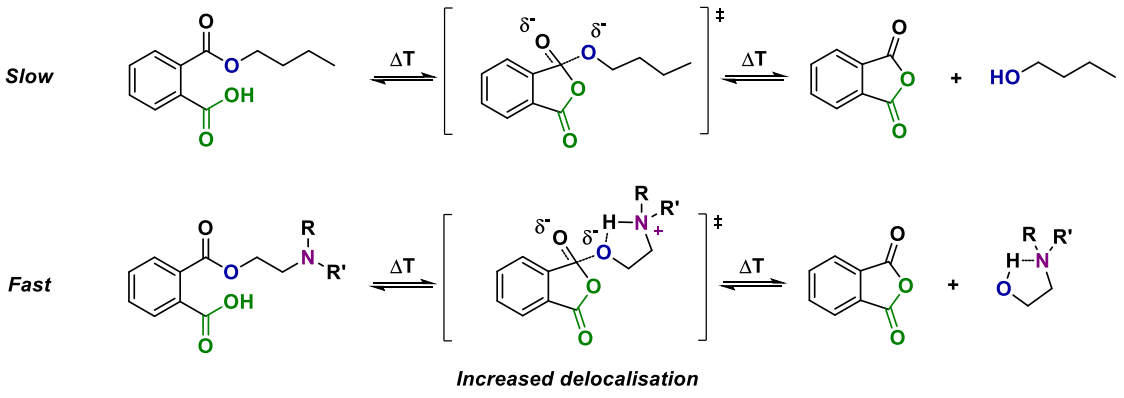


Figure S9. *Temperature dependence of exchange rate according to the adjusted Eyring equation. Values were obtained by fitting to a) single exponential, b) stretched exponential, c) double exponential (1st exponent) and d) double exponential (2nd exponent) decay from 160 °C to 110 °C. Note a small deviation from linearity at lower temperatures, when fitting was done to non-sufficiently relaxed data.*



Scheme S1. *Difference in transition state stabilisation between slow and fast exchanging bonds.*

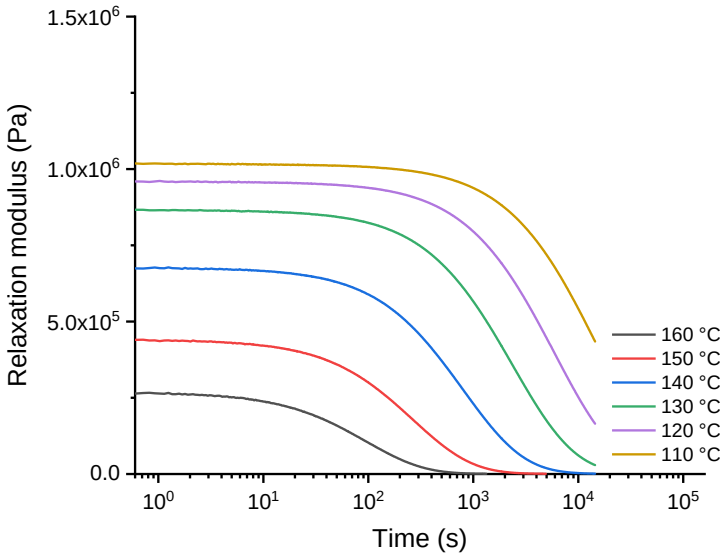


Figure S10. *Stress-relaxation graph of N-0% from 160 °C to 110 °C. Note that the absolute values of the initial relaxation modulus (G_0) cannot be compared to those of N-1% to N-20% since the building blocks for this material have been made from a different batch of pripol 2033. Nevertheless, this does not affect kinetic analysis.*

a) Activation enthalpy **b) Activation entropy**

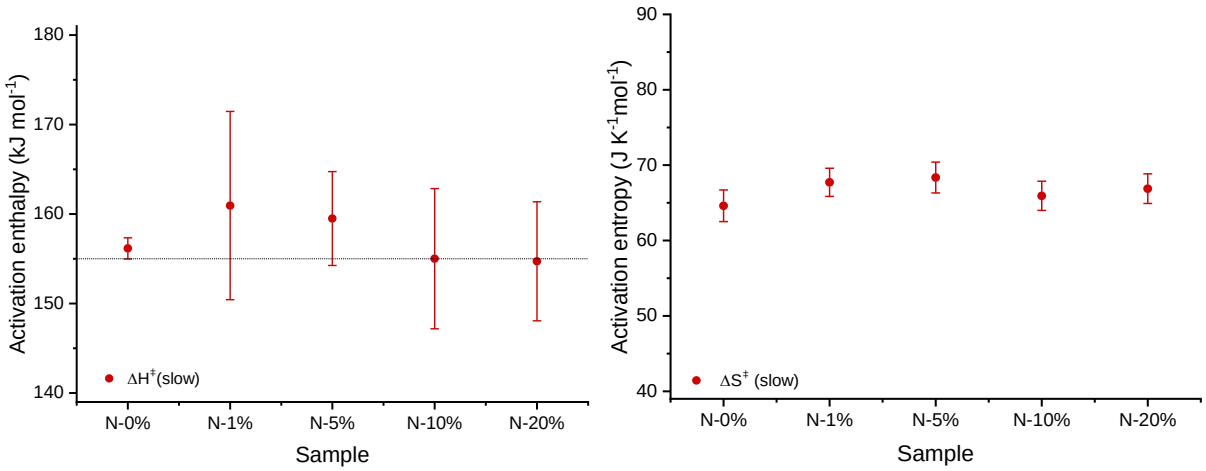


Figure S11. *Overview of a) activation enthalpy ($\Delta H^{\ddagger}$) and b) activation entropy ($\Delta S^{\ddagger}$) of N-0% and N-1% to N-20%. The respective data were obtained by fitting to a single and generalised Maxwell model respectively.*

Network composition and elastically active segments - Statistical modelling and rationale

To analyse the rheological results, we created a statistical code that allowed us to determine the average molar mass between two cross-linking points as well as the polymer fraction that is effectively trapped between two branching points based on the synthesis recipe. Following that, the corresponding value of the modelled plateau modulus was determined, as a function of the association probability of the moieties, and compared to the experimental data.

The first input data, which are directly coming from the synthesis, are the amount of the different monomers, $Eq_{dianhydride}$, Eq_{diol}, $Eq_{aminodiol}$ and Eq_{triol} as well as their corresponding molar mass $M_{dianhydride}$, M_{diol}, $M_{aminodiol}$ and M_{triol}.

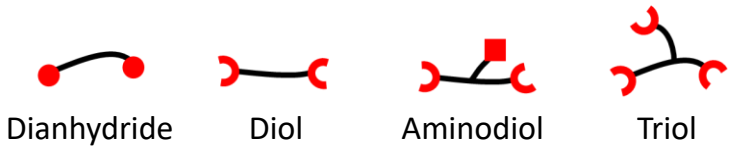


Figure S12. *Cartoon representation of each monomer.*

From these parameters, the fraction in number of end groups of each species are determined, accounting for the functionality of the branched building blocks:

$\upsilon_{dianhydride} = 1, \upsilon_{diol} = \frac{Eq_{diol}}{\Sigma}, \upsilon_{aminodiol} = \frac{Eq_{aminodiol}}{\Sigma}, \upsilon_{triol} = \frac{3}{2}\frac{Eq_{triol}}{\Sigma}$ with $\Sigma = Eq_{diol} + Eq_{aminodiol} + \frac{3}{2}Eq_{triol}$. For this system specifically, there was a stoichiometric amount of dianhydride, resulting in $\Sigma = 1$, as $\upsilon_{dianhydride}$ is fixed to 1.

We then introduce the probabilities $p_{ass,slow}$ and $p_{ass,fast}$ (or equivalently, the probabilities (1- $p_{ass,slow}$) and (1- $p_{ass,fast}$)) that at a specific moment, the end groups of the building blocks diol and triol, or of the building blocks aminodiol are associated (or equivalently, are not associated) to a dianhydride. These parameters are unknown.

It must be noted here that the probabilities $p_{ass,slow}$ and $p_{ass,fast}$ must be considered with care as they only approximate reality. For example, they do not account for the creation of inefficient small loops (containing, for example, only one dianhydride and one alcohol building block), which are not considered to contribute to elasticity.

With these parameters defined, a simulated polymer network could be generated by growing a random dianhydride from the left and the right side. The probability that at a specific moment a diol (**eq S4**), triol (**eq S5**) or aminodiol (**eq S6**) is connected at e.g. the right side of the dianhydride depends on both the fraction of the available building blocks and their reactivity (*i.e.*, their association probability).

$$p_{diol} = p_{ass,slow}\,\upsilon_{diol} \quad (S4)$$

$$p_{triol} = p_{ass,slow}\,\upsilon_{triol} \quad (S5)$$

$$p_{aminodiol} = p_{ass,fast}\,\upsilon_{aminodiol} \quad (S6)$$

Thus, the right side of the dianhydride has a probability equal to $(1 - p_{diol} - p_{triol} - p_{aminodiol})$ to be a chain end (*i.e.* an end of the polymer network assembly).

If a diol or triol is added, each of its remaining end group(s) has a probability equal to $p_{ass,slow}$ to be associated to a dianhydride, and allow the growing network to further develop. If an aminodiol is added, its remaining end group has a probability equal to $p_{ass,fast}$ to be associated to a dianhydride. If not, this would mean that the respective building block will act as a dead or free chain end.

If in a next step another dianhydride is added, the same process can be repeated over and over again. Each time an alcohol or a dianhydride is added, its molar mass is also added to the polymer network assembly under construction.

In this algorithm, the entire polymer network was not replicated but we rather focused on keeping track of the molecular segments between two branching points or chain ends: as soon as a chain end was reached or a triol was added, we stopped growing the polymer network assembly and applied the same protocol on the left side of the first, starting dianhydride, in order to determine whether the molecular segments under construction will be a free linear chain (*i.e.* with two chain ends), a dangling end (*i.e.* with a triol on one side and a chain end on the other) or a trapped segment (*i.e.* terminated by a triol on both sides). This protocol enabled the construction of thousands of molecular strands from which we could determine their average molar masses in number (*i.e.* $M_{n,free}$, $M_{n,dangling}$ and $M_{n,trapped}$), weight molar masses (*i.e.* $M_{w,free}$, $M_{w,dangling}$ and $M_{w,trapped}$), and weight proportions (*i.e.* ϕ_{free}, $\phi_{dangling}$ and $\phi_{trapped}$). Using these values, the theoretical plateau modulus $G_{0,N}$ was calculated using **eq S7**, according to which only the molecular segments trapped between two branching points contribute to the network elasticity:

$$G_{0,N} = \varphi_{trapped} \frac{\rho RT}{M_{w,trapped}} \quad (S7)$$

The molar mass distributions of the different types of segments can also be determined.

The same procedure was then used to isolate the contribution of the slower exchanging bonds to the network, which are responsible for the second shoulder observed in the relaxation curve. In this case, the $G_{0,N,slow}$ values were calculated by considering that only the molecular segments without any aminodiol are contributing to the sample elastic modulus.

Practically, in the code, the association state of the different groups are determined by generating random numbers between 0 and 1, and comparing their values to the corresponding probabilities.

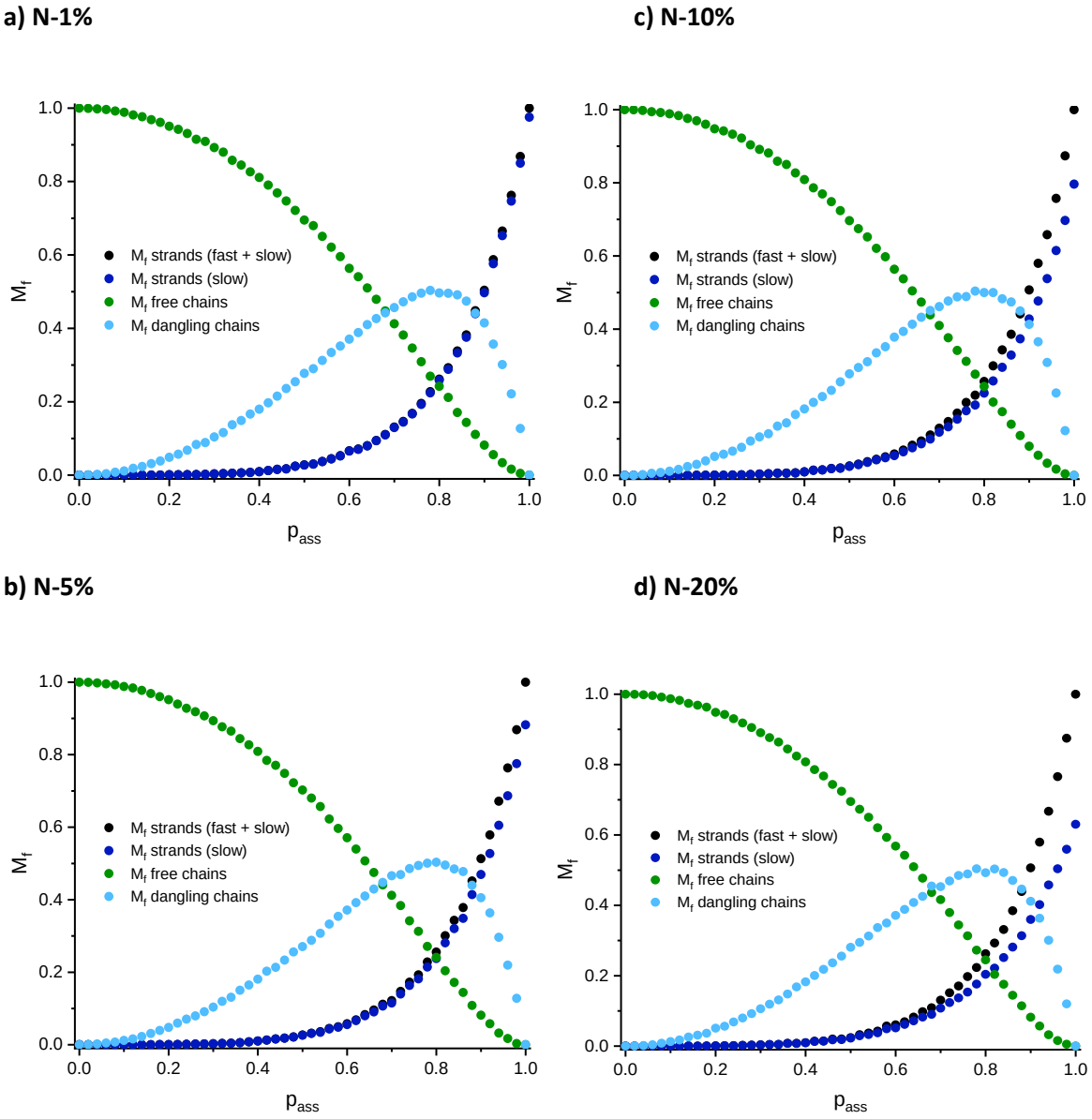


Figure S13. *Evolution of the mass fraction of free chains (green), dangling ends (light blue), trapped segments (black) and trapped segments which do not contain aminodiol (dark blue) as a function of the probability that any alcohol reacts with a dianhydride (with $p_{ass,slow} = p_{ass,fast} = p_{ass}$) for a) N-1%, b) N-5%, c) N-10% and d) N-20%.*

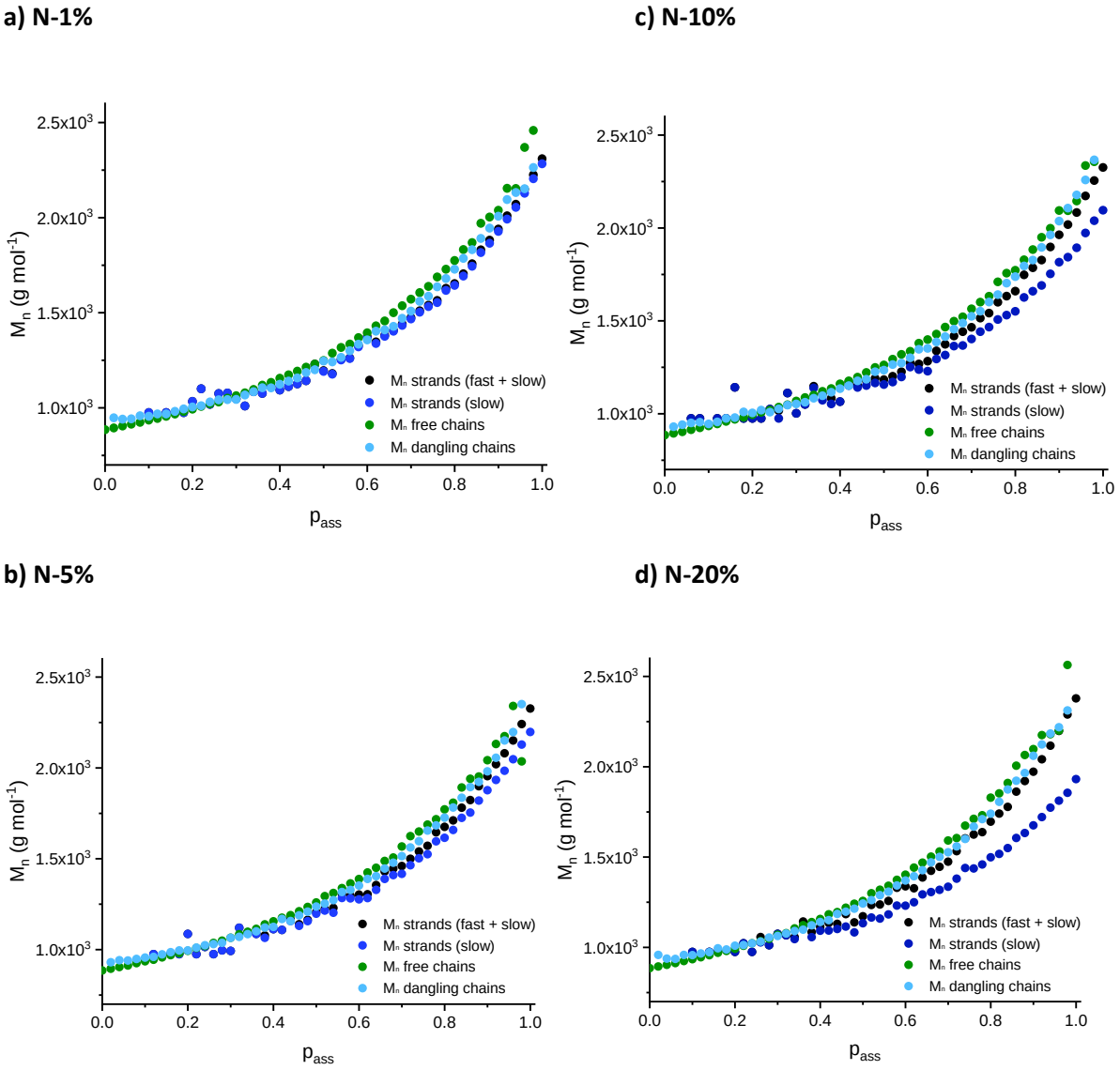


Figure S14. *Evolution of the molar mass of free chains (green), dangling ends (light blue), trapped segments (black) and trapped segments that do not contain aminodiol (dark blue) as a function of the probability that any alcohol reacts with a dianhydride (with $p_{ass,slow} = p_{ass,fast} = p_{ass}$) for a) N-1%, b) N-5%, c) N-10% and d) N-20%.*

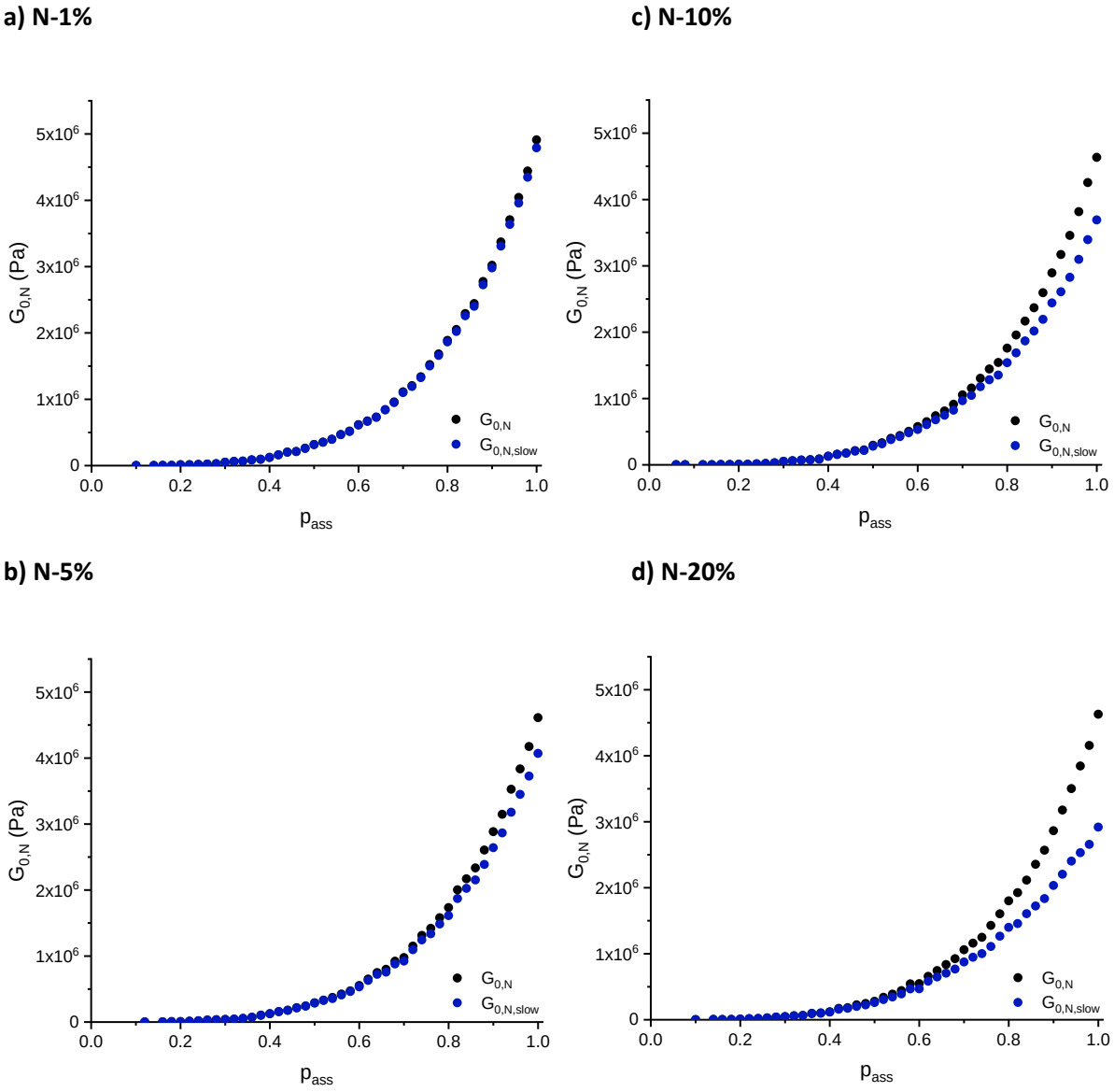


Figure S15. *Evolution of the theoretical plateau modulus $G_{0,N}$ (black) and theoretical plateau modulus $G_{0,N,slow}$ which does not consider segments with aminodiol (dark blue) as a function of the probability that any alcohol reacts with a dianhydride (with $p_{ass,slow} = p_{ass,fast} = p_{ass}$) for a) N-1%, b) N-5%, c) N-10% and d) N-20%.*

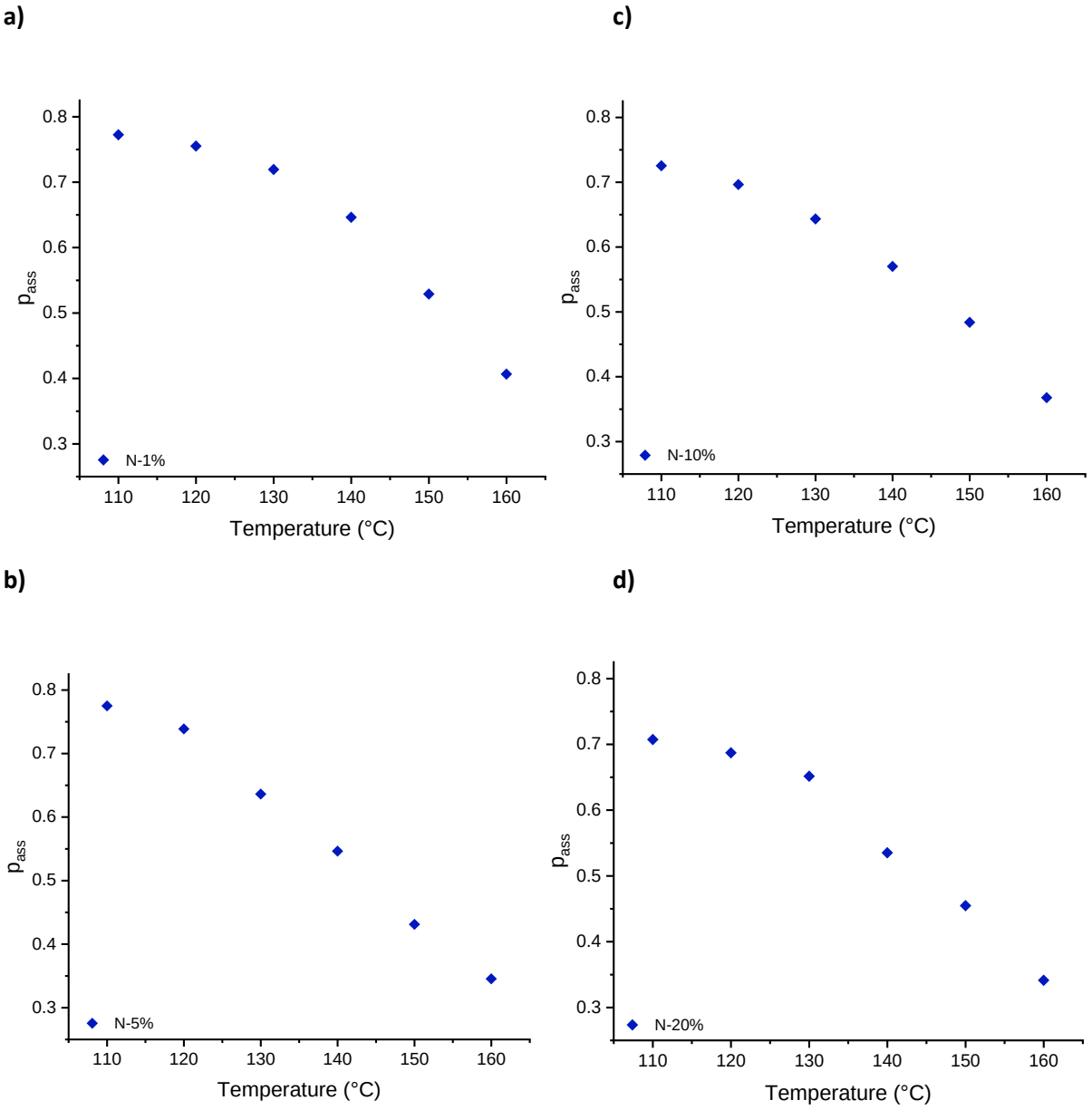


Figure S16. *Temperature dependence of association probability (p_{ass}) values for a) N-1%, b) N-5%, c) N-10% and d) N-20%.*

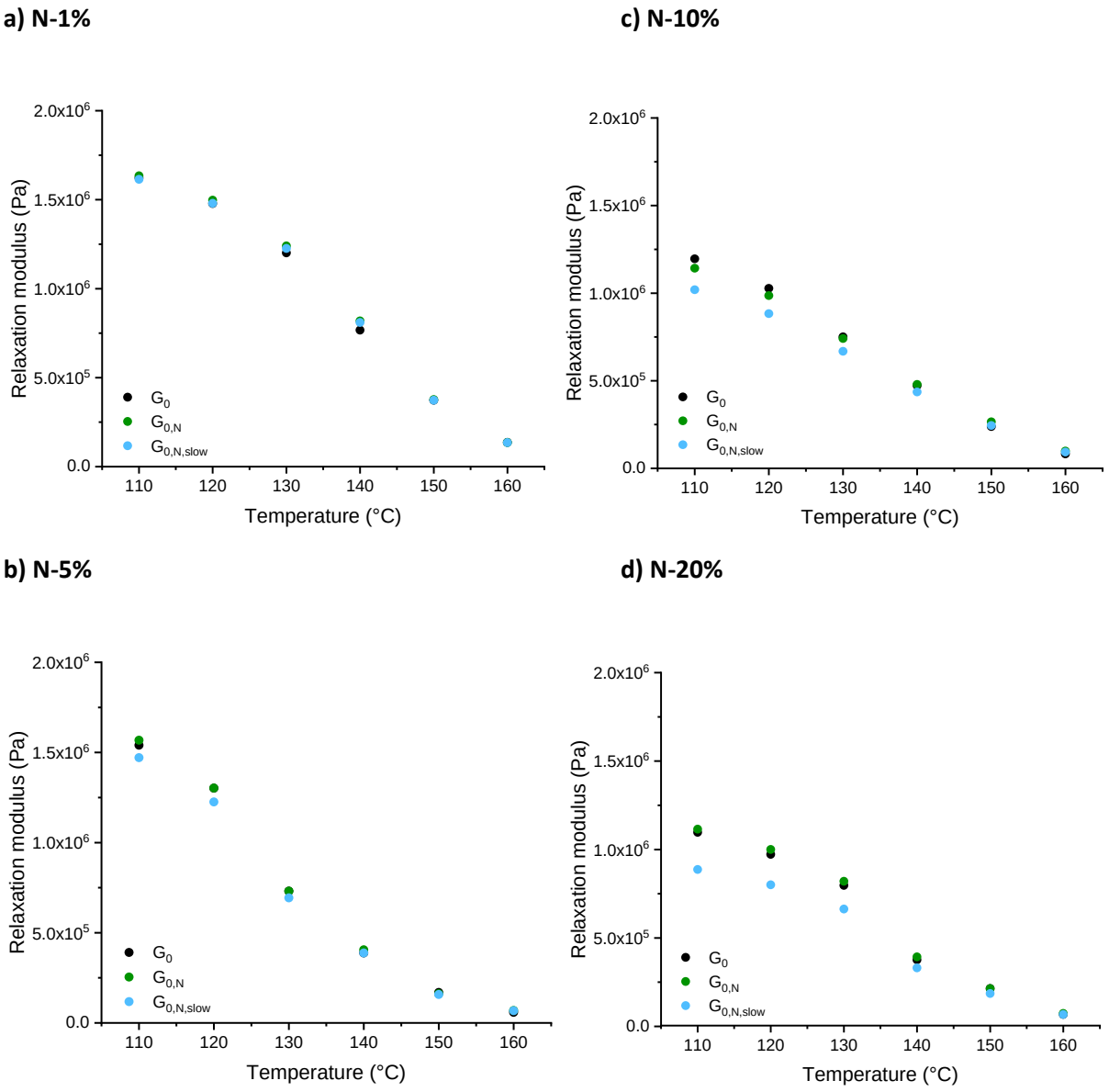


Figure S17. *Comparison of the experimental G_0 (black) and theoretical $G_{0,N}$ (green) plateau modulus and theoretical $G_{0,N,slow}$ (light blue) plateau modulus which does not consider segments with aminodiol as a function of temperature for a) N-1%, b) N-5%, c) N-10% and d) N-20%.*

Conversion of association probability to association constant

The association probability (p_{ass}) is defined as the probability that any alcohol functionality reacts with an anhydride functionality. In terms of concentration, at a fixed temperature, this value will represent the relative amounts of phthalate monoester bonds compared to anhydride and alcohol in the mixture (**eq S8**).

$$p_{ass} = \frac{[phthalate\ monoester]}{[Anhydride] + [Alcohol]} \quad (S8)$$

If for this system equimolar amounts of dianhydride and alcohol are used, **eq S8** simplifies to:

$$p_{ass} = \frac{[phthalate\ monoester]}{[Anhydride]_0} \quad (S9)$$

With $[Anhydride]_0$ being the initial concentration of anhydride added to the curing mixture.

In terms of chemical thermodynamics, the concentration of phthalate monoester bonds follows from the association constant (K_{ass}):

$$K_{ass} = \frac{[phthalate\ monoester]}{[Anhydride][Alcohol]} \quad (S10)$$

Since every association needs at least one anhydride and at least one alcohol and equimolar amounts of compounds are used, at equilibrium the concentration of anhydride is equal to the concentration of alcohol:

$$K_{ass} = \frac{[phthalate\ monoester]}{[Anhydride]^2} \quad (S11)$$

At equilibrium, the concentration of anhydride will be equal to the initial concentration of anhydride minus the concentration of anhydride needed to form a phthalate monoester bond:

$$K_{ass} = \frac{[phthalate\ monoester]}{([Anhydride]_0 - [phthalate\ monoester])^2} \quad (S12)$$

Starting from this expression, we can substitute the $[phthalate\ monoester]$ for $p_{ass}[Anhydride]_0$ in **eq S12**:

$$K_{ass} = \frac{[Anhydride]_0 p_{ass}}{([Anhydride]_0 - [Anhydride]_0 p_{ass})^2} \quad (S13)$$

Which could be further simplified to:

$$K_{ass} = \frac{1}{[Anhydride]_0} \frac{p_{ass}}{(1 - p_{ass})^2} \quad (S14)$$

For each investigated temperature, $[Anhydride]_0$ was calculated by dividing the total moles of anhydride added to the curing mixture by the volume of the disc-shaped sample ($V = \pi r^2 h$) used for each rheology experiment. The diameter of the sample was fixed at 8 mm (r = 4 mm), while the height (h) of the sample was measured by the rheometer as the gap distance at the start of each experiment (160 °C to 110 °C). The obtained values varied between 6.98x10^{-5} m^3 and 7.74x10^{-5} m^3.

From each calculated K_{ass} value, a dissociation constant (K_{diss}) could be derived by taking into account the following straightforward relationship:

$$K_{diss} = \frac{1}{K_{ass}} \quad (S15)$$

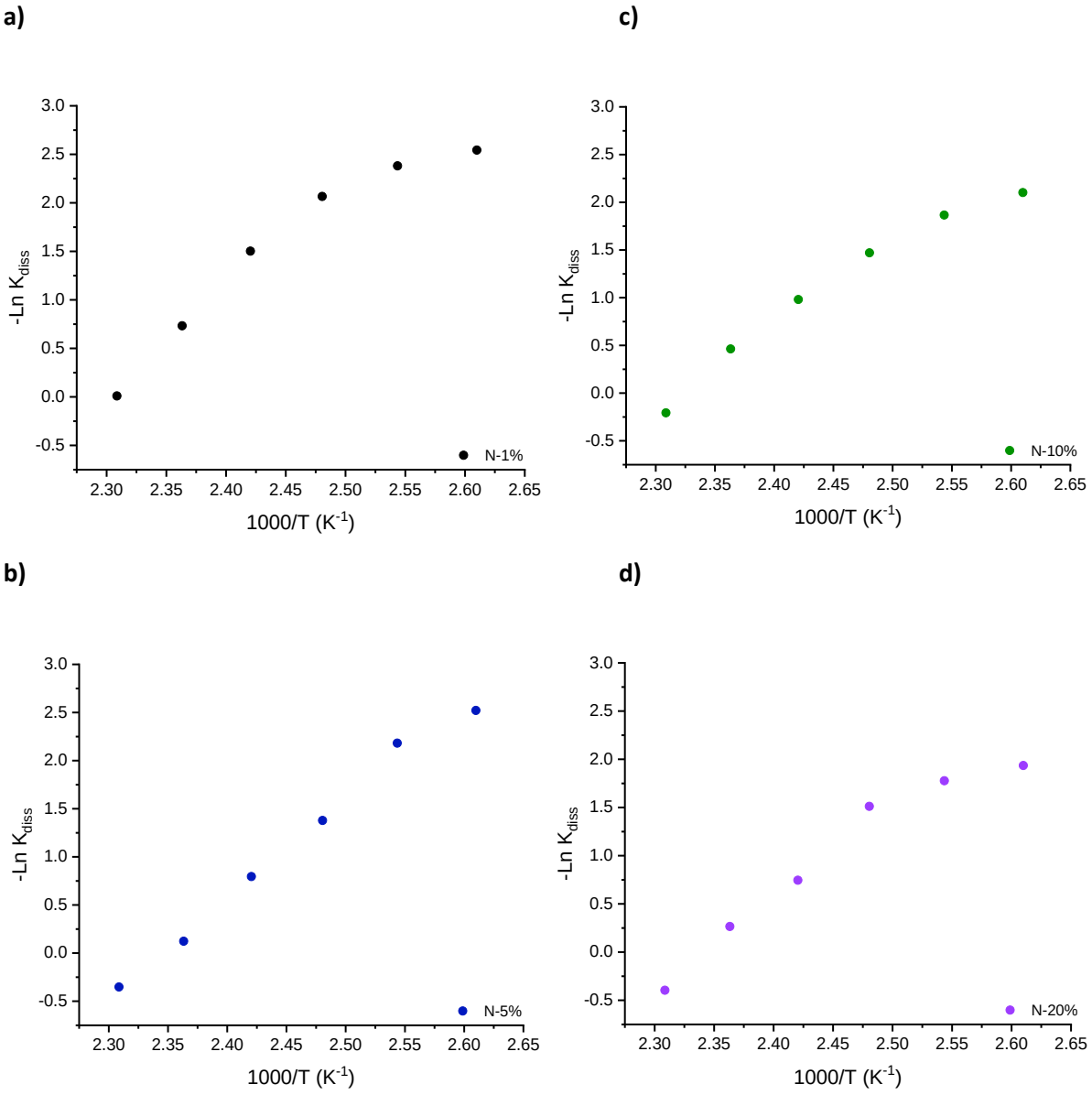


Figure S18. *Temperature dependence of dissociation constant (K_{diss}) values obtained from modelled association probability (p_{ass}) for a) N-1%, b) N-5%, c) N-10% and d) N-20%.*

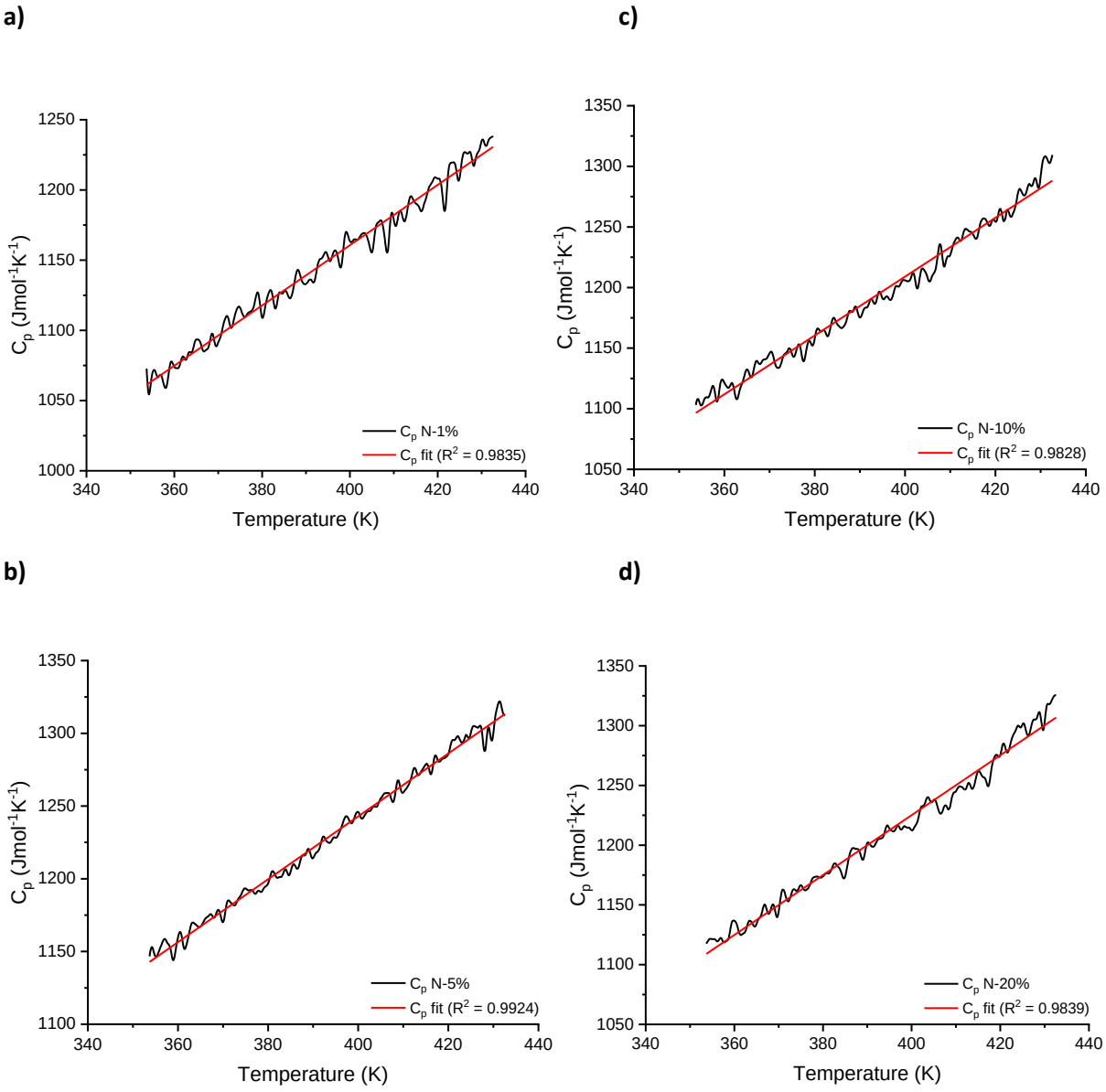


Figure S19. *Heat capacity as a function of temperature determined by modulated DSC measurements for a) N-1%, b) N-5%, c) N-10% and d) N-20%.*

Thermodynamic data from calorimetry measurements

At constant pressure, the following relationship between the relative change in enthalpy upon bond dissociation ($\Delta_r H$) and molar heat capacity (C_p) was taken from literature:[1]

$$\Delta_r H = \int_{T_1}^{T_2} C_p \; dT \qquad (S16)$$

Whereby the temperature dependent function of C_p can be written as:

$$C_p = a + bT + \frac{c}{T^2} \qquad (S17)$$

By fitting **eq S17** to the experimental data for C_p as a function of temperature (**Figure S20**), numerical values for a, b and c could be retrieved. Subsequently, combining **eq S16** and **eq S17** resulted in an expression of $\Delta_r H$ that could be numerically evaluated:

$$\Delta_r H = a(T_2 - T_1) + \frac{b}{2}\left({T_2}^2 - {T_1}^2\right) - c\left(\frac{1}{T_2} - \frac{1}{T_1}\right) \quad (S18)$$

At constant pressure, the following relationship between the relative change in entropy upon bond dissociation ($\Delta_r S$) and molar heat capacity (C_p) was taken from literature: [1]

$$\Delta_r S = \int_{T_1}^{T_2} \frac{C_p}{T}\, dT \quad (S19)$$

Subsequently combining **eq S19** and **eq S17** resulted in an expression of $\Delta_r S$ that could be numerically evaluated:

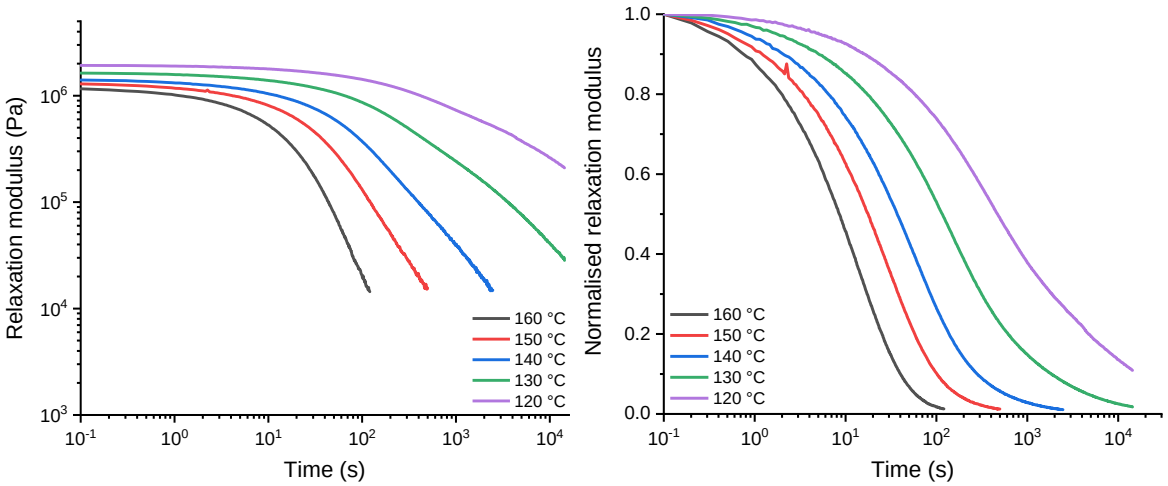
$$\Delta_r S = a\, ln\left(\frac{T_2}{T_1}\right) + b(T_2 - T_1) - \frac{c}{2}\left(\frac{1}{{T_2}^2} - \frac{1}{{T_1}^2}\right) \quad (S20)$$

***Table S3**. Overview of physical properties and relaxation data of (modified) vinylogous urethane networks.*

Vitrimer	T_g[a] (°C)	$T_{d5\%}$[b] (°C)	Swel. Rat.[c] (%)	Sol. Frac.[c] (%)	$\Delta H^{\ddagger}_{fast}$[d] (kJ.mol^{-1})	$\Delta H^{\ddagger}_{slow}$[d] (kJ.mol^{-1})	$\Delta S^{\ddagger}_{fast}$[d] (J K^{-1}mol^{-1})	$\Delta S^{\ddagger}_{slow}$[d] (J K^{-1}mol^{-1})
VU-ref	- 16	317	225 ± 2	8.9 ± 0.4	[e]	114 ± 4[e]	[e]	-22 ± 2[e]
VU-TfOH	-7	269	514 ± 22	12 ± 1.0	65 ± 0.6	80 ± 1.1	-98 ± 2	-78 ± 2

[a] Determined from the second heating in DSC analysis (10 °C.min^{-1}). [b] TGA onset temperatures after 5% weight loss ($T_{d5\%}$). [c] Obtained after swelling in THF for 24 h. [d] Obtained by fitting to a stretched single exponential decay. [d] Activation enthalpy ($\Delta H^{\ddagger}$) and entropy ($\Delta S^{\ddagger}$) values obtained by fitting the relaxation data of the double exponential decay to an adjusted Eyring equation. Errors were obtained by calculating the standard deviation on the respective values. Depending on the completeness of relaxation, a larger standard deviation was observed. [e] Fitting was done to a single exponential decay, since a double exponential decay did not lead to chemically interpretable data.

a) VU-ref



b) VU-*p*TsOH

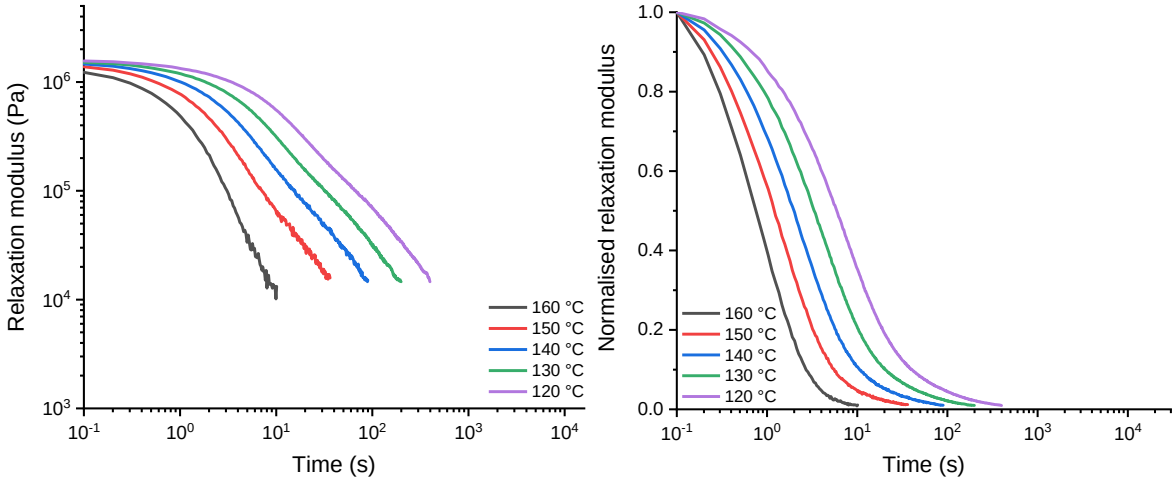


Figure S20. *Stress-relaxation graphs for a) VU-ref and b) VU-pTsOH measured at different temperatures between 160 and 120 °C with non-normalised (left) and normalised data (right). Note a slight increase in modulus with temperature due to entropic elasticity.*

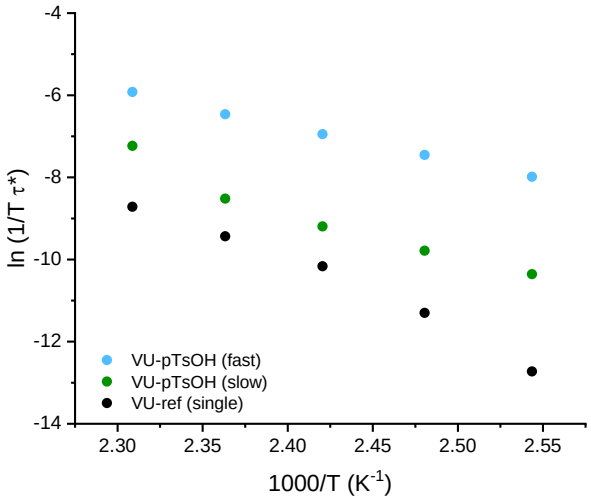


Figure S21. *Temperature dependence of exchange rate according to the adjusted Eyring equation for VU-ref and VU-pTsOH. Values were obtained by fitting to a single exponential or double exponential decay from 160 °C to 120 °C.*

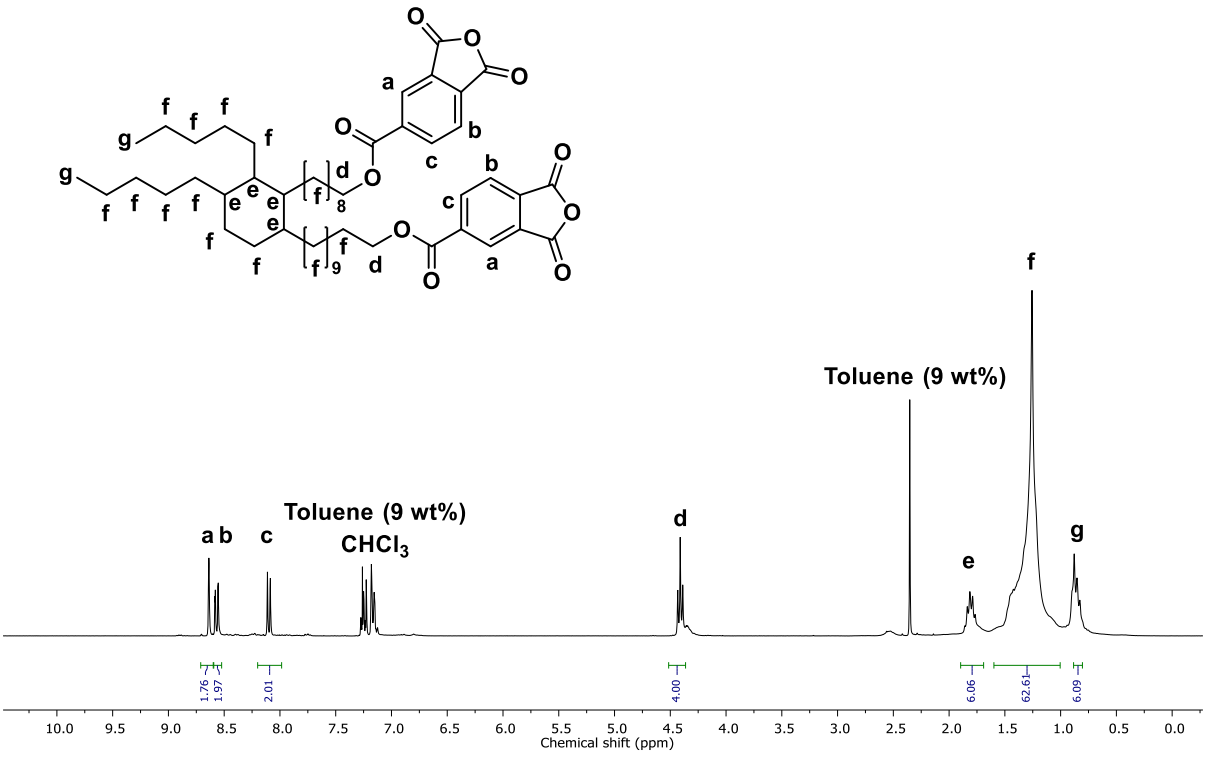


Figure S22. *^{1}H NMR of pripol dianhydride compound (**1**) in $CDCl_3$.*

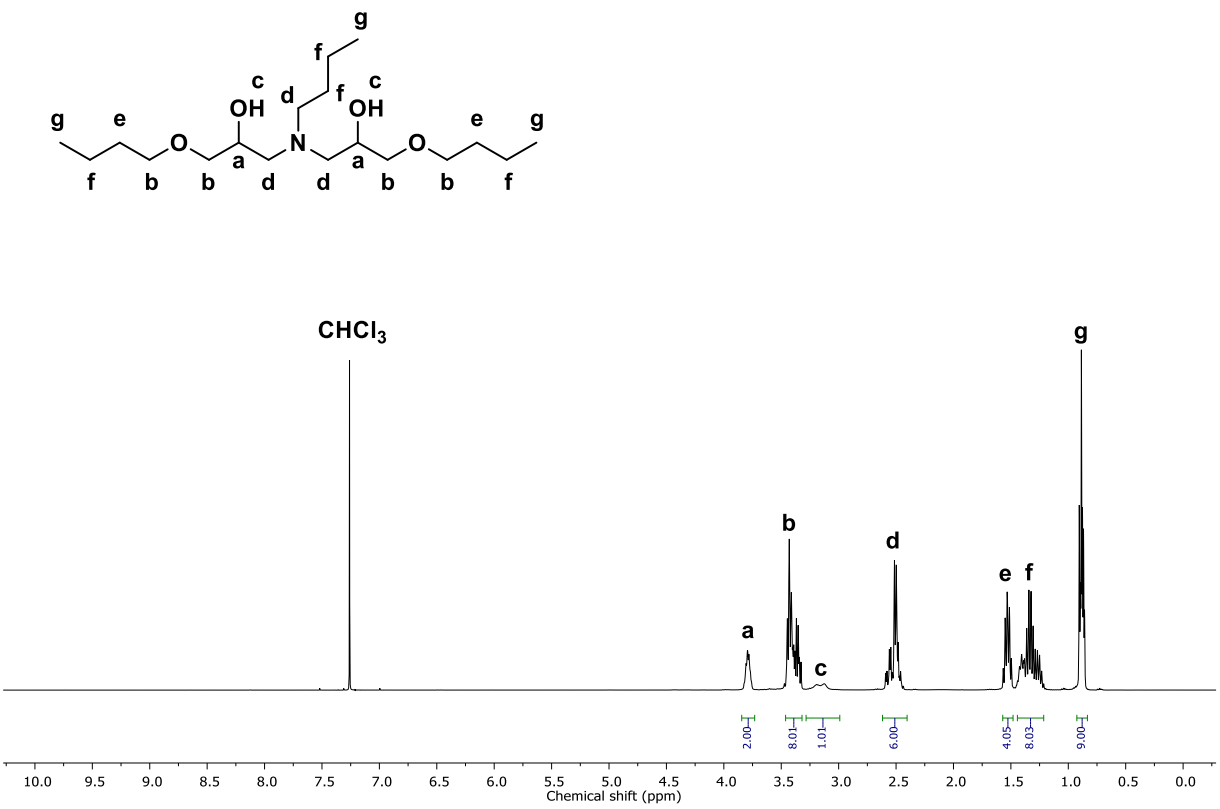


Figure S23. *^{1}H NMR of aminodiol compound (**2**) in $CDCl_3$.*

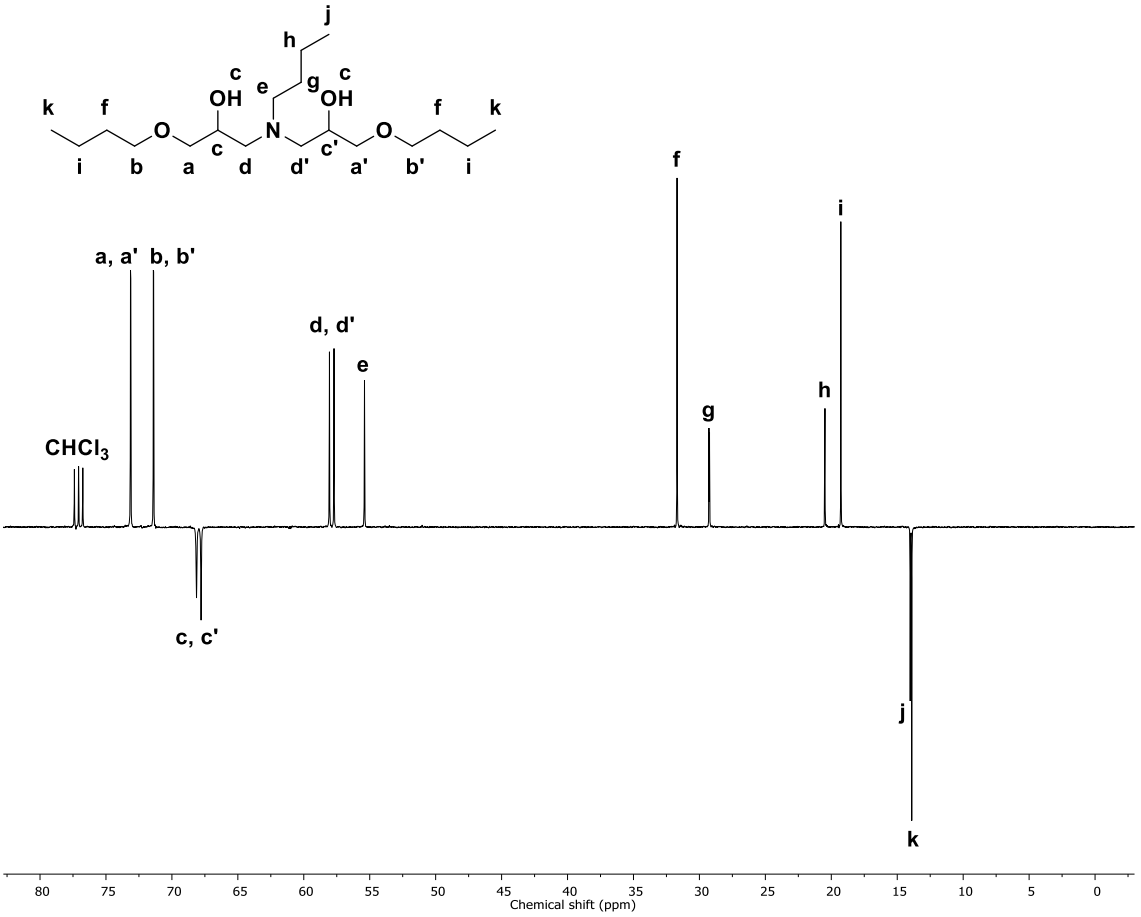


Figure S24. *^{13}C NMR of aminodiol compound (**2**) in $CDCl_3$.*

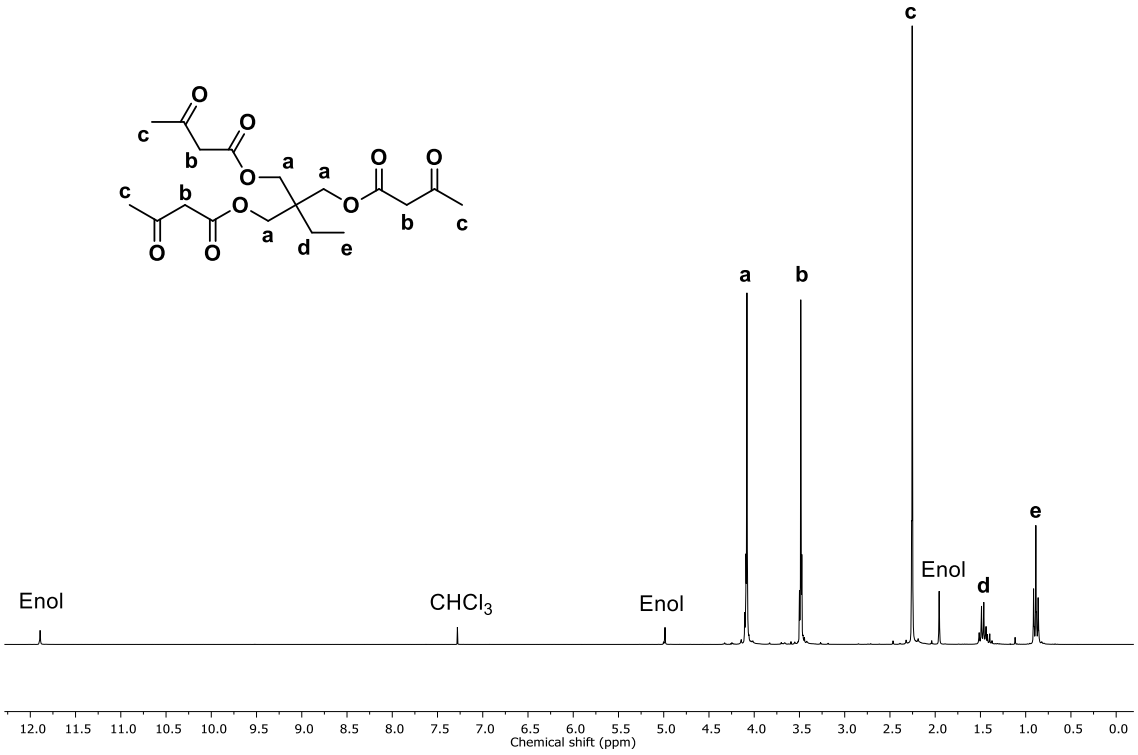


Figure S25. *^{1}H NMR of 1,1,1-trimethyl-propane trisacetoacetate in $CDCl_3$.*

Python code (example for N-1%)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import progressbar
import os

# molar masses:
Mw_connector = 885.19
Mw_bis_a = 118.17
Mw_bis_b = 333.51
Mw_tris = 134.17
# (molar) equivalents
Eq_connector = 1
Eq_bis_a = 0.396
Eq_bis_b = 0.004
Eq_tris = 0.4
# functionality
# caution! This is actually hardcoded in the simulation, for example to
calculate the Mw of the strands, for future reference only
F_connector = 2
F_bis_a = 2
F_bis_b = 2
F_tris = 3
# Resulting fraction (in number) of reactive groups in each species
# This is effectively the probability to encounter a specific reactive
group in the soup of equivalent groups
# so this takes the equivalents and functionality into account
prob_connector = Eq_connector * F_connector / (Eq_connector * F_connector)
prop_bis_a = Eq_bis_a * F_bis_a / (Eq_bis_a * F_bis_a + Eq_bis_b * F_bis_b
+ Eq_tris * F_tris)
```

```
prop_bis_b = Eq_bis_b * F_bis_b / (Eq_bis_a * F_bis_a + Eq_bis_b * F_bis_b
+ Eq_tris * F_tris)
prop_tris = Eq_tris * F_tris / (Eq_bis_a * F_bis_a + Eq_bis_b * F_bis_b +
Eq_tris * F_tris)

nbchain = 20000
numblocs = 400  # maximum number of blocks in a strand

step_size = 0.02

# Time (s) at which to determine the plateau modulus in the experimental
stress relaxation
exp_GN0_time = 1

def simulation(Mw_connector, Mw_bis_a, Mw_bis_b, Mw_tris, prop_bis_a,
prop_bis_b, prop_tris, p_attach_a, p_attach_b, nbchain, numblocs):
    # %probabilities to attach a block to the connector:
    p_tris = p_attach_a * prop_tris
    p_bis_a = p_attach_a * prop_bis_a
    p_bis_b = p_attach_b * prop_bis_b
    p_end = 1 - p_tris - p_bis_a - p_bis_b

    rand1 = np.random.rand(nbchain, numblocs)
    rand2 = np.random.rand(nbchain, numblocs)
    side_1 = np.where((rand1 < p_tris) & (rand2 < p_attach_a), 1, 0) + \
             np.where((p_tris <= rand1) & (rand1 < (p_tris + p_bis_a)) &
(rand2 < p_attach_a), 2, 0) + \
             np.where(((p_tris + p_bis_a) <= rand1) & (rand1 < (p_tris +
p_bis_a + p_bis_b)) & (rand2 < p_attach_b), 3, 0) + \
             np.where((p_tris + p_bis_a + p_bis_b) <= rand1, 100, 0)

    temp = side_1 == 100
    lastconnect_1 = np.where(np.count_nonzero(temp, axis=1) > 0,
np.argmax(temp, axis=1), numblocs)
    temp = side_1 == 0
    lastblock_1 = np.where(np.count_nonzero(temp, axis=1) > 0,
np.argmax(temp, axis=1), numblocs)
    last_1 = np.minimum(lastconnect_1, lastblock_1)
    temp = side_1 == 1
    first_tris_1 = np.where(np.count_nonzero(temp, axis=1) > 0,
np.argmax(temp, axis=1), numblocs)

    rand1 = np.random.rand(nbchain, numblocs)
    rand2 = np.random.rand(nbchain, numblocs)
    side_2 = np.where((rand1 < p_tris) & (rand2 < p_attach_a), 1, 0) + \
             np.where((p_tris <= rand1) & (rand1 < (p_tris + p_bis_a)) &
(rand2 < p_attach_a), 2, 0) + \
             np.where(((p_tris + p_bis_a) <= rand1) & (rand1 < (p_tris +
p_bis_a + p_bis_b)) & (rand2 < p_attach_b), 3, 0) + \
             np.where((p_tris + p_bis_a + p_bis_b) <= rand1, 100, 0)

    temp = side_2 == 100
    lastconnect_2 = np.where(np.count_nonzero(temp, axis=1) > 0,
np.argmax(temp, axis=1), numblocs)
    temp = side_2 == 0
    lastblock_2 = np.where(np.count_nonzero(temp, axis=1) > 0,
np.argmax(temp, axis=1), numblocs)
    last_2 = np.minimum(lastconnect_2, lastblock_2)
    temp = side_2 == 1
    first_tris_2 = np.where(np.count_nonzero(temp, axis=1) > 0,
np.argmax(temp, axis=1), numblocs)
```

```
    # %molar mass and strand specification:
    #     %branching strand:

    temp = np.broadcast_to(np.arange(numblocs), (nbchain, numblocs))
    branching_strands_1 = np.where(temp <= np.broadcast_to(first_tris_1,
(numblocs, len(first_tris_1))).transpose(),
                                   side_1,
                                   -1)  # [(first_tris_1 < last_1) *
(first_tris_2 < last_2)]
    branching_strands_2 = np.where(temp <= np.broadcast_to(first_tris_2,
(numblocs, len(first_tris_2))).transpose(),
                                   side_2,
                                   -1)  # [(first_tris_1 < last_1) *
(first_tris_2 < last_2)]
    dangling_strands_1 = np.where(temp <= np.broadcast_to(last_1,
(numblocs, len(last_1))).transpose(),
                                  side_1,
                                  -1)
    dangling_strands_2 = np.where(temp <= np.broadcast_to(last_2,
(numblocs, len(last_2))).transpose(),
                                  side_2,
                                  -1)
    temp = (first_tris_1 < last_1) & (first_tris_2 < last_2)
    branching_strands = np.concatenate((branching_strands_1[temp],
branching_strands_2[temp]), axis=1)
    temp = np.sum(branching_strands == 3, axis=1)
    mass_strands = Mw_connector + 2 * Mw_tris / 3 +
np.sum(branching_strands == 2, axis=1) * (Mw_bis_a + Mw_connector) + temp *
(Mw_bis_b + Mw_connector)
    mass_strands_strong = mass_strands[temp == 0]
    mass_strands_weak = mass_strands[temp > 0]

    temp = (first_tris_1 >= last_1) & (first_tris_2 >= last_2)
    free_chains = np.concatenate((dangling_strands_1[temp],
dangling_strands_2[temp]), axis=1)
    mass_free_chains = Mw_connector + np.sum(free_chains == 2, axis=1) *
(Mw_bis_a + Mw_connector) + np.sum(free_chains == 3, axis=1) * (Mw_bis_b +
Mw_connector) + np.sum(free_chains == 0, axis=1) * (0.5 * Mw_bis_a + 0.5 *
Mw_bis_b)

    temp = [(first_tris_1 < last_1) & (first_tris_2 >= last_2),
(first_tris_1 >= last_1) & (first_tris_2 < last_2)]
    dangling_chains =
np.concatenate((np.concatenate((branching_strands_1[temp[0]],
dangling_strands_2[temp[0]]), axis=1),

np.concatenate((dangling_strands_1[temp[1]], branching_strands_2[temp[1]]),
axis=1)))
    mass_dangling_chains = Mw_connector + np.sum(dangling_chains == 2,
axis=1) * (Mw_bis_a + Mw_connector) + np.sum(dangling_chains == 3, axis=1)
* (Mw_bis_b + Mw_connector) + np.sum(dangling_chains == 0, axis=1) * (0.5 *
Mw_bis_a + 0.5 * Mw_bis_b) + np.sum(dangling_chains == 1, axis=1) * Mw_tris
/ 3

    mass_total = mass_strands.sum() + mass_free_chains.sum() +
mass_dangling_chains.sum()

    Mn_strands = np.average(mass_strands) if len(mass_strands) else None
    Mw_strands = np.average(mass_strands, weights=mass_strands) if
mass_strands.sum() else Mn_strands
```

```
    Mfrac_strands = mass_strands.sum() / mass_total
    Mn_strands_strong = np.average(mass_strands_strong) if
len(mass_strands_strong) else None
    Mw_strands_strong = np.average(mass_strands_strong,
weights=mass_strands_strong) if mass_strands_strong.sum() else
Mn_strands_strong
    Mfrac_strands_strong = mass_strands_strong.sum() / mass_total
    Mn_strands_weak = np.average(mass_strands_weak) if
len(mass_strands_weak) else None
    Mw_strands_weak = np.average(mass_strands_weak,
weights=mass_strands_weak) if mass_strands_weak.sum() else Mn_strands_weak
    Mfrac_strands_weak = mass_strands_weak.sum() / mass_total
    Mn_free_chains = np.average(mass_free_chains) if len(mass_free_chains)
else None
    Mw_free_chains = np.average(mass_free_chains, weights=mass_free_chains)
if mass_free_chains.sum() else Mn_free_chains
    Mfrac_free_chains = mass_free_chains.sum() / mass_total
    Mn_dangling_chains = np.average(mass_dangling_chains) if
len(mass_dangling_chains) else None
    Mw_dangling_chains = np.average(mass_dangling_chains,
weights=mass_dangling_chains) if mass_dangling_chains.sum() else
Mn_dangling_chains
    Mfrac_dangling_chains = mass_dangling_chains.sum() / mass_total

    roRT = 16051319359 (calculated from experimental values)

    GN0 = Mfrac_strands * roRT / Mw_strands if Mw_strands else None
    GN0_strong = Mfrac_strands_strong * roRT / Mw_strands if Mw_strands
else None
    GN0_weak = Mfrac_strands_weak * roRT / Mw_strands if Mw_strands else
None
    # GN0 = Mfrac_strands * roRT / Mn_strands if Mn_strands else None
    # GN0_strong = Mfrac_strands_strong * roRT / Mn_strands if Mn_strands
else None
    # GN0_weak = Mfrac_strands_weak * roRT / Mn_strands if Mn_strands else
None

    return [(Mfrac_strands, Mn_strands, GN0),
            (Mfrac_strands_strong, Mn_strands_strong, GN0_strong),
            (Mfrac_free_chains, Mn_free_chains),
            (Mfrac_dangling_chains, Mn_dangling_chains)]


p_attach = np.arange(0, 1 + step_size, step_size)
p_attach_2D = np.array(np.meshgrid(p_attach, p_attach)).T.reshape(-1, 2)
strands = []
strands_strong = []
free_chains = []
dangling_chains = []

for p in progressbar.progressbar(p_attach, 0, len(p_attach)):
    a, b, c, d = simulation(Mw_connector, Mw_bis_a, Mw_bis_b, Mw_tris,
prop_bis_a, prop_bis_b, prop_tris, p, p, nbchain, numblocs)
    strands.append(a)
    strands_strong.append(b)
    free_chains.append(c)
    dangling_chains.append(d)

data = pd.concat([
    pd.DataFrame(p_attach, columns=['p_ass-A']),
    pd.DataFrame(p_attach, columns=['p_ass-B']),
```

```
    pd.DataFrame(strands, columns=['Mfrac_strands', 'Mn_strands', 'GN0']),
    pd.DataFrame(strands_strong, columns=['Mfrac_strands_strong',
'Mn_strands_strong', 'GN0_strong']),
    pd.DataFrame(free_chains, columns=['Mfrac_free_chains',
'Mn_free_chains']),
    pd.DataFrame(dangling_chains, columns=['Mfrac_dangling_chains',
'Mn_dangling_chains'])
], axis=1)

data.to_excel(os.path.splitext(os.path.basename(__file__))[0] + '.xlsx')

# strands = np.array(strands)
# strands_strong = np.array(strands_strong)
# free_chains = np.array(free_chains)
# dangling_chains = np.array(dangling_chains)

fig1 = plt.figure()
ax = fig1.add_subplot(111)
ax.set_xlabel(r'$\mathrm{p}_\mathrm{ass-A}$')
ax.set_ylabel(r'$\varphi$')
ax.plot(data['p_ass-A'], data['Mfrac_strands'], label='Trapped segments')
ax.plot(data['p_ass-A'], data['Mfrac_strands_strong'], label='Trapped
segments (no B)')
ax.plot(data['p_ass-A'], data['Mfrac_dangling_chains'], label='Dangling
chains')
ax.plot(data['p_ass-A'], data['Mfrac_free_chains'], label='Free chains')
ax.legend()
plt.show()

fig2 = plt.figure()
ax = fig2.add_subplot(111)
ax.set_xlabel(r'$\mathrm{p}_\mathrm{ass-A}$')
ax.set_ylabel(r'$\mathrm{M}_\mathrm{n}$')
ax.plot(data['p_ass-A'], data['Mn_strands'], label='Trapped segments')
ax.plot(data['p_ass-A'], data['Mn_strands_strong'], label='Trapped segments
(no B)')
ax.plot(data['p_ass-A'], data['Mn_dangling_chains'], label='Dangling
chains')
ax.plot(data['p_ass-A'], data['Mn_free_chains'], label='Free chains')
ax.set_yscale('log')
ax.legend()
plt.show()

fig3 = plt.figure()
ax = fig3.add_subplot(111)
ax.set_xlabel(r'$\mathrm{p}_\mathrm{ass-A}$')
ax.set_ylabel(r'$\mathrm{G}^0_\mathrm{N}$')
ax.plot(data['p_ass-A'], data['GN0'], label='All')
ax.plot(data['p_ass-A'], data['GN0_strong'], label='no B')
ax.legend()
plt.show()

if os.path.exists(os.path.splitext(os.path.basename(__file__))[0] +
'.csv'):
    print("Opening experimental stress relaxation
({}.csv)".format(os.path.splitext(os.path.basename(__file__))[0]))
    import aprheology
    from operator import itemgetter
    experimental = aprheology.StressRelaxation(
        os.path.splitext(os.path.basename(__file__))[0] + '.csv',
        normalise_relax_mod=False
```

```
    )
    print("Stress relaxation loaded...")
    pT = []
    for curve in experimental.curves:
        GN0 = experimental.get_tau_intersect(curve['data']['Relaxation
Modulus'].to_numpy(),

curve['data']['Time'].to_numpy(),
                                             exp_GN0_time)
        p = experimental.get_tau_intersect(data['p_ass-A'].to_numpy(),
                                           data['GN0'].to_numpy(),
                                           GN0)
        pT.append([curve['T'], GN0, p])
    pT.sort(key=itemgetter(0))
    pd.DataFrame(pT, columns=['T', 'GN0 (exp)', 'p_ass-
A']).to_excel(os.path.splitext(os.path.basename(__file__))[0] + '_pT.xlsx')
else:
    print("No experimental stress relaxation data found
({}.csv)".format(os.path.splitext(os.path.basename(__file__))[0]))
```

For python package regarding the automated fitting of relaxation data:https://pypi.org/project/aprheology/ (aprheology 0.3.2)

References

(1) Höhne, G. W. H. From DSC Curve to Thermodynamic Potential Function. *Thermochim. Acta* **1991**, *187* (C), 283–292.